Pergamon

Computers Math. Applic. Vol. 31, No. 12, pp. 97–138, 1996

0898-1221/96 $15.00 + 0.00

S0898-1221(96)00080-6

Optimal and Nearly Optimal Algorithms for Approximating Polynomial Zeros

V. Y. PAN
Department of Mathematics and Computer Science
Lehman College, City University of New York, Bronx, NY 10468, U.S.A.
VPAN@LCVAX.BITNET

(Received November 1994; accepted January 1995)

Abstract—We substantially improve the known algorithms for approximating all the complex zeros of an n^{th} degree polynomial $p(x)$. Our new algorithms save both Boolean and arithmetic sequential time, versus the previous best algorithms of Schönhage [1], Pan [2], and Neff and Reif [3]. In parallel (NC) implementation, we dramatically decrease the number of processors, versus the parallel algorithm of Neff [4], which was the only NC algorithm known for this problem so far. Specifically, under the simple normalization assumption that the variable x has been scaled so as to confine the zeros of $p(x)$ to the unit disc $\{x : |x| \leq 1\}$, our algorithms (which promise to be practically effective) approximate all the zeros of $p(x)$ within the absolute error bound 2^{-b}, by using order of n arithmetic operations and order of $(b+n)n^2$ Boolean (bitwise) operations (in both cases up to within polylogarithmic factors). The algorithms allow their optimal (work preserving) NC parallelization, so that they can be implemented by using polylogarithmic time and the orders of n arithmetic processors or $(b+n)n^2$ Boolean processors. All the cited bounds on the computational complexity are *within polylogarithmic factors from the optimum* (in terms of n and b) under both arithmetic and Boolean models of computation (in the Boolean case, under the additional (realistic) assumption that $n = O(b)$).

Keywords—Complex polynomial zeros, Approximation, Polynomial factorization, Parallel algorithms, Computational complexity, Sequential algorithms.

1. INTRODUCTION

1.1. The Subject, Some History, and a Summary of Our Results

The problem of solving a polynomial equation $p(x) = 0$ substantially motivated the development of mathematics throughout the centuries. As particular examples of this influence, one may recall the origin of complex numbers from the solution formulae for quadratic equations (these formulae have been known already in ancient Greece), the fundamental theorem of algebra, which states the existence of a complex solution to $p(x) = 0$ (the first celebrated proof of this theorem, given by Gauss in 1799, contained a substantial flaw, corrected by Ostrowski in 1920), and the Galois theory of 1832, which extended the earlier theorem of Ruffini 1813 and Abel 1826 on nonexistence of solution formulae in radicals for a polynomial equation of a degree n if $n \geq 5$ (such formulae have been known, since the 16^{th} century, for $n = 3$ [del Ferro, Tartaglia, Cardano] and $n = 4$ [Ferrari]). In the absence of explicit solution formulae, numerous algorithms for approximating polynomial zeros have been proposed, and they are still appearing in great number,

The author is grateful to D. Bini, P. Kirrinnis, and A. Neff, for (p)reprints of [3–7], and to A. Sadikou, for helpful comments.
The results of this paper are to be presented at the 27^{th} Annual ACM Symposium on the Theory of Computing, 1995 (see [8]); the author is grateful to the ACM for the permission to reuse them.
Supported by NSF Grant CCR 9020690 and PSC CUNY Awards Nos. 664334 and 665301.

motivated by the importance of approximating polynomial zeros to many areas of algebraic and numerical computing. The designers of these algorithms have introduced various techniques of independent interest. As a single major example, we recall Weyl's paper [9]. Together with [10], this paper presented one of the two historically first algorithms, both of 1924, that converged to the zeros of any input polynomial $p(x)$, thus giving the two first *algorithmic proofs* of the fundamental theorem of algebra. Weyl's remarkable ideas are still practically important for approximating polynomial zeros; moreover, historically his algorithm was the first application of his quadtree technique, now widely used in various areas of computing (see, for instance, [11] on some older applications and [12,13] on more recent ones, to template matching and to the unsymmetric eigenvalue problem). On some further historical and technical background, we refer the reader to [1,9,10,14–20].

In the present paper, we consider algorithms that solve the problem in the general case, by approximating all the n zeros of any input polynomial $p(x)$ of degree n, with no restriction on the disposition of the zeros on the complex plane (in particular, these zeros may form various clusters, which occurs, for instance, in numerical treatment of polynomials with multiple zeros), and we will estimate the worst case computational complexity of performing these algorithms, assuming no initial approximations to the zeros of $p(x)$ available (compare [1; 19, pp. 497–499]). (Here and hereafter, we count every polynomial zero of multiplicity m as m zeros.)

In some sense, our present paper completes this line of study. Namely, our algorithms not only substantially improve the previous ones but are also *asymptotically optimal* (up to within polylogarithmic factors), both in their sequential and parallel implementations and under both Boolean and arithmetic models of measurement. Furthermore, *the presented new algorithms are machine independent, can be implemented on various real computers, and promise to be practically effective.*

1.2. Models of Computation

We will estimate the computational cost of the solution algorithms under the customary arithmetic and Boolean computational models of (sequential) RAM [22] and EREW PRAM [23]. In the latter (parallel) case, each arithmetic or Boolean processor is allowed to perform at most one arithmetic or Boolean operation, respectively, in each time-step, and we will assume a variant of Brent's scheduling principle, according to which a single processor may simulate the work of s processors in $O(s)$ time. (We only need to use this principle in order to decrease some of our processor bounds by the factor $\log n$.) We will express the computational complexity (computational cost) estimates as $O_A(t,p)$, under the arithmetic computational models, and as $O_B(t,p)$, under the Boolean computational models. This way will unify the bound $O(tp)$ on the sequential time, under the RAM models, and the simultaneous bounds $O(t)$ on parallel time and $O(p)$ on the number of processors, under the EREW PRAM models. We will state the complexity estimates in terms of n and b, assuming, for convenience, that all the zeros of $p(x)$ have magnitudes at most 1 and are sought within the absolute error bound 2^{-b}.

1.3. Previous Results

Among several effective algorithms that at the time of their publication supported record estimates for the worst case computational complexity of approximating the n complex polynomial zeros [1–4,9,10,24–26], the latest achievements are due to [3,4]. In [4], the only known NC-solution has been presented, for which the computational cost bound $O_B((\log n)^2 \log(bn), (b+n)n^8b^2)$ has been proved. In [3] the record sequential computational cost bounds have been claimed, that is, $O_A(n^{1+\tilde{\epsilon}} \log b, 1)$ and $O_B((b+n)n^{2+\tilde{\epsilon}}(\log b)^2 \log\log b, 1)$, for any fixed positive $\tilde{\epsilon}$. The latter bounds are substantially superior to the previous records, $O_B(n^3b, 1)$ of [1] and $O_A(n^2, 1)$ of [2], both of which we recall up to polylogarithmic factors.

Technically, the elegant algorithm of [3] relies on

(a) some correlations between the zeros of $p(x)$ and one of its higher order derivatives [27],
(b) the algorithms of [1,28] for splitting a polynomial into factors over a complex disc D provided that this disc is sufficiently well isolated from the zeros of $p(x)$ that lie outside D (the isolation is quantitatively measured by an isolation ratio; see our Definition 2.2, taken from [2]), and
(c) an algorithm for simultaneous approximation of the distances from the origin to all the zeros of $p(x)$ [1,2; 19, pp. 458–462; 29].

More specifically, the latter algorithm and the results of [27] were used in [3] in order to compute a disc D, with no zeros of $p(x)$ on or near its boundary circle and with comparable numbers of the zeros of $p(x)$ (that is, with the same number of them up to within a fixed constant factor) in its interior and in its exterior. Then the results of [1,28] were applied in order to split $p(x)$ numerically into two factors (of comparable degrees) having all their zeros in or, respectively, all outside the disc D, and the same process was recursively applied to each of the factors. This solved the problem in $O(\log n)$ recursive steps, since every splitting was balanced so as to decrease the degree of its input polynomial by a fixed constant factor. The result was a new surprising extension of the earlier pioneering versions of balanced splitting techniques of [30,31], applied in [30,31] to a simpler (real) case. (Compare [4,28,32,33] on the other known techniques for achieving balanced splitting.)

1.4. Some Problems Left Open by the Previous Research

The algorithm of [3] was a major step towards optimizing polynomial rootfinding, but it has also raised some new questions. In particular, approximating the zeros of $p(x)$ according to this algorithm involved approximations of all the zeros of some higher order derivative of $p(x)$ and, recursively, of the factors of this derivative and of the higher order derivatives of the factors. This complication has not allowed one to run the algorithm in NC (that is, by using parallel time $(\log(bn))^{O(1)}$ and $(bn)^{O(1)}$ arithmetic or Boolean processors). Avoiding this computation should have enabled us to decrease the overall asymptotic (both arithmetic and Boolean) cost bounds by the factor $c(\tilde{\epsilon})n^{\tilde{\epsilon}}$, where $c(\tilde{\epsilon}) \to \infty$ as $\tilde{\epsilon} \to 0$.

On the other hand, the algorithm of [3] computes a disc for splitting $p(x)$, which is isolated from the zeros of $p(x)$ lying outside this disc, but this isolation is not as strong as necessary in order to support the desired upper estimates for the arithmetic and Boolean time involved. Either the algorithm has to be improved or the claimed upper estimates must be increased by at least the factor $n^{1/3}$ (compare our Remarks 4.1, 4.2, 5.1, 6.1, and 9.2).

Besides, the algorithm of [3] required some further nontrivial elaboration in order to avoid a dramatic blow-up of its computational cost in its application to some special but important class of input polynomials $p(x)$. Namely, in the form in which this algorithm was presented, it runs into problems for the input polynomials (such as $p(x) = x^n + \sum_{i=0}^{n-1} p_i x^i$, where all the $|p_i|$ are very small) all of or most of whose zeros form a *massive cluster* having a small diameter σ. In order to compute a balanced splitting of such a polynomial $p(x)$, one has to separate some of its zeros in the cluster from each other. This is a numerically hard problem whose solution requires computations with a bit-precision exceeding $\log_2(1/\sigma)$, so that the Boolean cost of the solution is unbounded as $\sigma \to 0$. In this case, approximation of the zeros must be worked out without computation of a balanced splitting of $p(x)$, in order to ensure any reasonable bound on the overall computational complexity. (As we have already mentioned, various clusters of polynomial zeros frequently arise in numerical treatment of polynomials with multiple zeros.)

1.5. Our Techniques and the Main Theorem

In the present paper, we address all the three problems cited above and, as a result, substantially improve the construction and the main result of [3], with a respective impact on various

computational tasks, whose solution requires approximating polynomial zeros. Our progress relies on introducing new geometric, analytic, and algebraic techniques for

(a) recursive screening and discarding the zeros of the higher order derivatives (without approximating these zeros),
(b) recursive contraction of an area of search for a splitting disc,
(c) computing an unbalanced splitting into factors of a polynomial that has a massive set of clustered zeros,
(d) descending from Graeffe's iteration, and
(e) perturbation of Padé approximations.

Our resulting algorithms run in polylogarithmic parallel time and support new record bounds of the orders n (arithmetic) and $(b+n)n^2$ (Boolean) on both sequential time and the number of processors (ignoring polylogarithmic factors). (As we will show in Section 1.7, these bounds are asymptotically optimum (up to within polylogarithmic factors) unless $b = o(n)$.) Specifically, we arrive at the following estimates (to be deduced in Section 8 and improved slightly in Appendix C).

THEOREM 1.1. *Let all the n unknown zeros, $z_1, \dots, z_n$, of a given monic polynomial $p(x)$ of degree n have magnitudes at most 1, that is, $|z_i| \le 1$, for all i. For a fixed positive b, write*

$$\bar{b} = (b+2)n + \log n + 2. \tag{1.1}$$

Then, approximations z_i^* satisfying

$$|z_i - z_i^*| < 2^{-b}, \qquad \text{for } i = 1, \dots, n, \tag{1.2}$$

can be computed at a cost bounded by any of the following four expressions:

(a) $PBC_Z(b,n) = O_B((\log n)^3(\log \bar{b})^2, (M(n^3 + \bar{b}n \log \bar{b}))/((\log \bar{b})^2(\log n)^2))$,

(b) $PAC_Z(b,n) = O_A((\log n)^3 \log \bar{b}, n^2/((\log \bar{b})(\log n)^2))$,

(c) $PRAC_Z(b,n) = O_A((\log n)^3 t_{3,1}(\bar{b},n), n/\log n)$,

(d) $SAC_Z(b,n) = O_A((\log n)^2 t_{2,1}(\bar{b},n)n, 1)$.

Here and hereafter, A, B, C, P, R, S and Z of PBC_Z, PAC_Z, $PRAC_Z$ and SAC_Z abbreviate the words "arithmetic," "Boolean," "complexity," "parallel," "randomized," "sequential," and "zeros," respectively, and we write

$$M(d) = (d \log d) \log \log d, \tag{1.3}$$

$$t_{i,j}(\bar{b},n) = (\log n)^i + (\log \bar{b})^j, \qquad i = 2,3,4; \quad j = 1,2. \tag{1.4}$$

The estimates of part (c) are randomized (of the Las Vegas type, that is, the estimates include the cost of verification if the computed solution is correct); the estimates of parts (a), (b) and (d) are deterministic.

1.6. Comments to the Main Theorem

REMARK 1.1. The estimates of Theorem 1.1 show efficacy of our algorithms (supporting these estimates) in the case where b and n are large, as this occurs, for instance, in the major application to approximate solution of systems of polynomial equations, via the elimination techniques. Actually, our algorithms are effective already for moderate b and n.

REMARK 1.2. The normalization assumption is that $|z_i| \leq 1$ for all i can be ensured by means of estimating $r_{\max} = \max_i |z_i|$, within the factor $2n$, at a cost $O_A(1,n)$ [19,34], or within the factor 1.01 (say), at a cost $O_A(\log n, n)$ (see Fact 2.2(b)), followed by scaling the variable x, at a cost $O_A(\log n, n/\log n)$; furthermore, we may then make $p(x)$ monic by dividing all its coefficients by the leading coefficient. Bringing the zeros of a polynomial inside the unit disc can be facilitated by other means such as shifts of the variable x (see Fact 2.1) (so as to bring the origin into the readily available center of gravity of the zeros, $\sum_{i=1}^{n} z_i/n$, before estimating $r_{\max}$) and the transition to the reverse polynomial $x^n p(1/x) = \prod_i (1 - xz_i)$. Moreover, Theorem 1.1 can be restated assuming other normalizations of the input.

Parts (a) and (b) of Theorem 1.1 are supported by the same algorithm. More specifically (but ignoring polylogarithmic factors), one of the stages of this algorithm uses $O(n^2)$ arithmetic operations and $O(n)$-bit precision of computing, versus $O(n)$ operations and $O(bn+n)$-bit precision used at all other stages. These arithmetic estimates immediately lead to part (b). Part (a) follows when we combine the cited arithmetic and bit-precision bounds, for each stage of the algorithm, with the known estimates

$$\mu_B(d) = O_B(\log d, d \log\log d), \tag{1.5}$$

for the Boolean complexity of an arithmetic operation with bounded d-bit numbers. (The sequential Boolean complexity represents the number of Boolean operations, also called bit-operations, involved.) Each arithmetic operation is ultimately reduced to adding and/or multiplying two integers modulo $2^d - 1$ at a cost bounded by $O_B(\log d, d/\log d)$ and/or by (1.5), respectively, by means of Ofman's and/or Schönhage-Strassen's algorithms, respectively (see [22,35–39]). Furthermore, one may improve the Boolean complexity bound slightly, by means of the techniques of binary segmentation, when one implements the algorithm that supports part (a) (see Remark 8.1 in Section 8). If the algorithms of this paper rely on using $\mu_B^*(d)$ Boolean operations (bit-operations) per an arithmetic operation over two integers modulo $2^d - 1$ (recall that $\mu_B^*(d)$ has a bound of the order d^2 based on the straightforward algorithms and the order $d^{\log_2 3}$, $\log_2 3 = 1.5849\ldots$, based on the algorithms of [40]), then the resulting estimate for the sequential Boolean complexity of approximating the n zeros of $p(x)$ turns into $O(n\mu_B^*(bn))$, up to a polylogarithmic factor, and the parallel Boolean complexity estimates of Theorem 1.1 change similarly.

The arithmetic sequential time and processor bounds of part (b) are deceptively large: we decrease them roughly by the factor n and turn them into the bounds of parts (c) and (d), by modifying one of the stages of the algorithm supporting part (b) so as to use fewer arithmetic operations but a higher precision of computation at this stage. In Sections 12 and 13, we show how to control the precision of computing by this modified algorithm so as to arrive at essentially the same Boolean cost bounds as we obtain in part (a). In Appendix C we review some techniques for a further decrease of the arithmetic cost bounds, though these techniques abandon control over the precision and Boolean cost of the computations, allowing their potential blow-up. This, of course, makes such techniques unrealistic and limits their value.

Since the Boolean cost bounds (reflecting the precision required in the computations) are more informative for the users, it is important that our Boolean sequential time bound of part (a) (that is, $O(n^3 + \bar{b}n)$, within polylogarithmic factors) substantially improves the previous (long standing) record, $O(n^3 + \bar{b}n^2)$, of [1, Section 19], as well as the cited Boolean cost bound claimed in [3]. (Note that in practical computation of polynomial zeros it is common to have b and $\bar{b}$ of the order at least n and n^2, respectively, and the terms $\bar{b}n$ and $\bar{b}n^2$ dominate the above cost estimates.)

1.7. Comparison of Upper and Lower Bounds

According to the customary definition of [23], parallelization of our algorithms is optimal since they run in polylogarithmic time and support the Boolean and arithmetic work bounds tp

(that is, time * processors bounds) that match the record sequential time bounds for the same computational problem.

Let us show that we also reach (up to within polylogarithmic factors) the optimum bounds on the sequential time and work. Indeed, the upper bound of part (d) of Theorem 1.1 is quite close to the known lower bounds of the orders n (obvious) and $\log(b/n)$ (see [26]), both lower bounds applied already to the complexity of approximating a single zero of $p(x)$. Furthermore, the involvement of the precision of the order of bn bits is required already in the case of polynomials such as $p(x) = (x-5/7)^n + p_0$ for small positive p_0, whose zeros jump from $(5/7)+|p_0|^{1/n}\exp(2\pi\sqrt{-1}\,h/n)$ to $5/7$, for $h = 0, 1, \ldots, n-1$, in the result of the shift from $(x-5/7)^n + p_0$ to $(x-5/7)^n$. Similar jumps of the zeros are caused by a small perturbation of any of, say, $n/4$ trailing coefficients of $p(x) = (x-5/7)^n$, that is, by transition to $p(x) = (x-5/7)^n + p_k x^k$ for small positive p_k and for $0 < k < n/4$. This implies that the input values of the $n/4$ trailing coefficients of $p(x)$ must involve the order of bn^2 bits, to ensure the worst case output approximation of even a single zero of $p(x)$ within the absolute error bound 2^{-b}, and we arrive at the following lower bounds.

FACT 1.1. Let $O_B(t,p)$ denote the Boolean complexity of approximating (within 2^{-b}) a zero z_i of a monic polynomial $p(x)$ of degree n, all of whose zeros lie in the unit disc $\{x : |x| \le 1\}$. Then $tp = \Omega(bn^2)$; that is, asymptotically in n and b, the product tp of the time and processors bounds has an order of at least bn^2.

Fact 1.1 implies that the upper bounds of part (a) of Theorem 1.1 are also tight (up to within polylogarithmic factors) provided that $n = O(b)$.

The presented argument that supports Fact 1.1 also implies the lower bound $\Omega(n\mu_B^*(nb))$ on the Boolean complexity of approximating the polynomial zeros by any algorithm that consists of arithmetic operations, each involving $\mu_B^*(d)$ bit-operations, where the two operands are represented by a pair of d-bit strings. The upper bounds based on our algorithms supporting Theorem 1.1 meet this bound up to polylogarithmic factors.

1.8. Numerical Factorization of a Polynomial in the Complex Field

Our algorithms (like ones of [1,3,4,28,30,31,33]) proceed by numerically splitting $p(x)$ into a pair of factors and, then, by recursively splitting each factor. The recursive process stops when it computes a factorization of $p(x)$ into linear factors satisfying the bound

$$\left\| p(x) - \prod_{i=1}^{n}(x - z_i^*) \right\| < \epsilon \, \|p(x)\|, \qquad \epsilon = 2^{-\hat{b}}. \tag{1.6}$$

Here and hereafter, we write

$$\|u(x)\| = \sum_i |u_i|, \qquad \text{for } u(x) = \sum_i u_i x^i. \tag{1.7}$$

Computation of $z_1^*, \ldots, z_n^*$ satisfying (1.6) can be called *numerical factorization of a polynomial in the complex field.* The bound (1.6), for $\hat{b}$ of the order bn, guarantees the bound (1.2) on the errors of all approximations by z_i^* to the zeros z_i of $p(x)$ (see Fact 2.6).

Corollary 2.3 implies the converse implication, of (1.6) by (1.2), for $b = \hat{b} + n + \log n$, which enables us to extend the estimates of Theorem 1.1 (for b replaced by $\hat{b}$) to the problem of computing a numerical factorization (1.6) for $p(x)$. On the other hand, the argument supporting Fact 1.1 does not apply to the problem of computing a numerical factorization (1.6), so that one may hope to solve this problem at a smaller computational cost.

1.9. On Some Alternative Techniques and Extensions

Although our algorithms are optimal (up to within polylogarithmic factors), further work may substantially improve their practical performance (in particular, see our Remark 8.1, on binary

segmentation, and see [7], on the techniques of splitting). Furthermore, it is quite plausible that some techniques used in our algorithms may turn out to be practically most effective in their combination with some heuristic approaches to approximating polynomial zeros. For instance, one may utilize Facts 2.1 and 2.2 of Section 2 in order to devise some heuristic algorithms for computing a basic disc D for splitting polynomial $p(x)$. Then, one may apply Graeffe's iteration and our techniques of recursive descending in order to strengthen the isolation of the zeros of $p(x)$ lying in the disc D from the other zeros of $p(x)$ and thus to facilitate splitting $p(x)$ over this disc.

On the other hand, some alternative algorithms that lead to substantially inferior upper estimates for the computational complexity of approximating polynomial zeros and even heuristic algorithms should not be discounted. In particular, for practical approximation of complex polynomial zeros, the most promising alternative to the approach of this paper probably comes from the Durand-Kerner algorithm [41,42] and its various modifications (such as Aberth's [43] and its implementation in [44]), which rely on Newton's iteration for Viète's system of polynomial equations for the zeros of $p(x)$. The absence of global convergence proofs and of any reasonably good computational complexity estimates for these iterative algorithms is partly compensated by their very good record in numerical experiments. On the other hand, these algorithms require us to use either the order of n^2 arithmetic operations per iteration (which is roughly n times as many as we use in our entire algorithms) or a much higher precision of computing (to support application of fast multipoint polynomial evaluation, which is a basic step of these algorithms).

Some other techniques known to be effective for approximating polynomial zeros may also be highly successful in their extensions and applications to other major computational tasks. In this regard, we have already cited Weyl's (quadtree) technique for approximating polynomial zeros [9] (also compare its extensions in [2; 19, pp. 517–522; 26,29,45,46]). In another example, the zerofinding techniques of [31] only apply to the special case of polynomials, all of whose zeros are real, but these techniques have effective extensions to the symmetric eigenvalue computation [32,47]. Yet another example is given by Newton's iteration. Already in its classical or slightly modified form, it rapidly approximates a single zero of $p(x)$ [20,48]. Its more advanced variation, known as the path lifting method [48,49], has an excellent univariate version of [50], according to which all the zeros of a univariate polynomial are approximated at a cost $O_A((n+\hat{b})(\log n)^2, n)$, for $\hat{b} \geq bn + n + 2$ (compare Fact 2.6 in our Section 2), but this method shows its greater power in its application to solve a system of polynomial equations [51–55].

Finally, due to increased effectiveness of the algorithms available for approximating polynomial zeros, one may reexamine their various possible extensions, including extensions to such problems as solving a system of polynomial equations and the matrix eigenvalue computation.

1.10. Organization of the Paper

In the next sections and appendices, we will describe our algorithms in some detail but will omit some tedious techniques of the error and precision analysis, already available at length in [1,2,4], and will refer the reader to [1] and to [28, Appendices A and B] on several details of splitting a polynomial into two factors over a fixed disc. The reader may find a less formal exposition of the entire subject in [21].

We will present the results in the following order. After some preliminaries in Section 2, we revisit and modify the algorithm of [3] in Sections 3 and 4. In Sections 5 and 6, we describe our techniques for recursive contraction of a disc and for recursive screening and discarding of the zeros of a higher order derivative without computing their approximations. We summarize our basic algorithm for computing an isolated disc for splitting $p(x)$ and estimate its cost in Section 7. Based on this algorithm, we prove Theorem 1.1 in Section 8. The proof uses an algorithm for splitting $p(x)$ over an isolated disc, which we briefly recall in Section 9, in the case of the unit disc, referring the reader to [1,28] for detailed presentations. We improve this

algorithm in Section 10 (by applying Graeffe's iteration for lifting the isolation ratio of the input disc and our new descending techniques) and extend it to the case of splitting over any disc in Section 11. In Sections 12 and 13, we show how to control the precision of computations required in the algorithms of Sections 9 and 10 (by means of perturbation of Padé approximation). In Appendices A and B, for the sake of completeness, we reproduce two auxiliary results from [3,27], so that our paper can be read independently of [3,27]. In Appendix C, we review some algorithms that slightly improve the arithmetic (but not Boolean!) complexity estimates for splitting $p(x)$ and, consequently, for approximating its zeros.

2. DEFINITIONS, AUXILIARY RESULTS AND TECHNICAL BACKGROUND ON RECURSIVE SPLITTING

Hereafter, log denotes $\log_2$.

DEFINITION 2.1. *$D(X,R)$ denotes the disc $\{x : |x-X| \le R\}$, with a center X and a radius R.*

DEFINITION 2.2. *Consider a monic polynomial (with zeros $z_1,\dots,z_n$),*

$$p(x) = \prod_{i=1}^{n}(x-z_i) = x^n + \sum_{i=0}^{n-1} p_i x^i, \tag{2.1}$$

$$|z_1| \le |z_2| \le \cdots \le |z_n| \le 1. \tag{2.2}$$

Then for an integer k, $0 < k < n$, and for a positive r satisfying the bound

$$|z_k| \le r < |z_{k+1}|, \tag{2.3}$$

the pair of polynomials $F_k(x) = \prod_{i=1}^{k}(x-z_i)$ and $G_{n-k}(x) = p(x)/F_k(x)$ is called the splitting of the polynomial $p(x)$ over the disc $D(0,r)$, and the disc $D(0,r)$ is called a splitting disc for $p(x)$. Any pair of monic polynomials $F_k^*(x)$ (of degree k) and $G_{n-k}^*(x)$ (of degree $n-k$) satisfying

$$\|p(x) - F_k^*(x)G_{n-k}^*(x)\| \le \epsilon\|p(x)\|, \tag{2.4}$$

for the norm of (1.7) and for any fixed positive ϵ, is called an ϵ-splitting of a polynomial $p(x)$ over the disc $D(0,r)$. Assuming (2.1)–(2.3), the disc $D(0,r)$ and the splitting of $p(x)$ over it are called (α,β)-balanced if $\alpha n \le k \le \beta n$, and the disc $D(0,r)$ is called f-isolated (or equivalently, in terms of [2], having an isolation ratio of at least f) if $1 \le f \le |z_{k+1}/z_k|$. For every pair a and f satisfying $0 < a < 1 < f$, a disc is called an (a,f)-splitting disc for a given polynomial if this disc is both $((1-a)/2,(1+a)/2)$-balanced and f-isolated. All these definitions also apply to $p(x)$ and any disc $D(X,r)$ (with any center X replacing 0) if they hold for the disc $D(0,r)$ and for the polynomial $q(y) = p(y+X)$, replacing the $p(x)$.

The following known fact (see [39]) bounds the arithmetic complexity of shifting from $p(x)$ to $q(y)$.

FACT 2.1. For a given pair of complex $t \ne 0$ and X and for a polynomial $p(x)$ of (2.1), the coefficients of the monic polynomial $\hat{q}(y) = \sum_{i=0}^{n} \hat{q}_i y^i = t^{-n}p(ty+X)$ can be computed at a cost bounded by $O_A(\log n, n)$.

There are various ways of utilizing Fact 2.1. For instance, we may choose $X = -p_{n-1}/n$ so as to cancel the term $\hat{q}_{n-1}y^{n-1}$ of $\hat{q}(y)$; this would shift the origin into the center of gravity of the n zeros of $\hat{q}(y)$. In Sections 6 and 11, we apply Fact 2.1 in order to reduce the study of various splitting discs to the case where such a disc has its center in the origin. Otherwise, in this paper, we will usually apply Fact 2.1, for $t = 1$, in order to approximate the distances from a fixed complex point X to all the zeros of a fixed polynomial. To achieve this goal, we first shift the origin into X and then apply the algorithm that supports the following fact.

FACT 2.2. (Compare [1,2; 19, pp. 458–462; 29,46,56,57].)

(a) For a pair of fixed $c > 0$ and $d \geq 0$, one may, at a cost bounded by $O_A((\log n)^2, n)$, compute the values $\underline{r}_1, \bar{r}_1, \ldots, \underline{r}_n, \bar{r}_n$ such that $\underline{r}_k \leq |z_k| \leq \bar{r}_k = (1 + c/n^d)\,\underline{r}_k$, $k = 1, \ldots, n$.

(b) The cost bound decreases to $O_A(\log n, n)$ if $d = 0$ and if $\bar{r}_k$ and $\underline{r}_k$ are sought only for $k = 1$ and/or $k = n$.

The next theorem gives us upper bounds on the complexity of computing an ϵ-splitting of $p(x)$ over a sufficiently well-isolated disc that lies inside the unit disc $D(0,1)$, the latter restriction being motivated by (2.2).

THEOREM 2.1. *Let B^*, $\hat{b}$, c, k, n, R, and X denote seven given values, where X is complex, B^*, $\hat{b}$, c, and R are positive, k and n are integers, $0 < k < n$, and*

$$2^{-B^*} \leq R \leq 1 - |X|. \tag{2.5}$$

Let $\bar{B} = B^* + \hat{b} + n$, let a polynomial $p(x)$ satisfy (2.1),(2.2), and let the disc $D(X, R)$ be f-isolated for $f = 1 + c/n$. Then, a $2^{-\hat{b}}$-splitting of polynomial $p(x)$ into two factors, $F_k^*(x)$ and $G_{n-k}^*(x)$ (defined according to Definition 2.2 and satisfying (2.4) for $\epsilon = 2^{-\hat{b}}$), can be computed at a cost bounded as follows:

(a) $PBC_S(\bar{B}, n) = O_B((\log n)(\log \bar{B})^2, (M(n^3 + \bar{B}n \log \bar{B}))/(\log \bar{B})^2)$,

(b) $PAC_S(\bar{B}, n) = O_A((\log n) \log \bar{B}, n^2/\log \bar{B})$,

(c) $PRAC_S(\bar{B}, n) = O_A((\log n)t_{3,1}(\bar{B}, n), n)$, *allowing randomization (of the Las Vegas type),*

(d) $SAC_S(\bar{B}, n) = O_A((\log n)t_{2,1}(\bar{B}, n)n, 1)$.

Here, $t_{i,j}(\bar{B}, n)$ and $M(d)$ are defined by (1.3) and (1.4), and the subscript S abbreviates the word "splitting."

The proof is given in Sections 9–11.

Clearly, the disc $D(0, \bar{r}_k)$ is f-isolated if $1 \leq f \leq \underline{r}_{k+1}/\bar{r}_k$. (Here, we use the notation of Fact 2.2.) Therefore, if $\underline{r}_{k+1}/\bar{r}_k > 1 + c/n$, we may apply Theorem 2.1 and reduce the problem of factorization of $p(x)$ to the similar problem for $F_k^*(x)$ and $G_{n-k}^*(x)$. Our goal is in continuation of this recursive process until we approximate the linear factors $x - z_i$ and, therefore, the zeros z_i, for all i. The desired upper bounds on the output errors follow from the bounds on the errors of the auxiliary approximations to the factors, due, in particular, to the following estimate from [1, Section 5].

FACT 2.3. Let

$$\begin{aligned} \|p(x) - f_1(x) \cdots f_h(x)\| &\leq \frac{\epsilon h \, \|p(x)\|}{n}, \\ \|f_1(x) - f(x)g(x)\| &\leq \epsilon_h \, \|f_1(x)\|, \end{aligned} \tag{2.6}$$

for the norm defined by (1.7), for some polynomials $f_1(x), \ldots, f_h(x)$, $f(x)$ and $g(x)$, and for

$$\epsilon_h \leq \frac{\epsilon \, \|p(x)\|}{n \prod_{i=1}^{h} \|f_i(x)\|}. \tag{2.7}$$

Then

$$\|p(x) - f(x)g(x)f_2(x) \cdots f_h(x)\| \leq (h+1)\frac{\epsilon \, \|p(x)\|}{n}. \tag{2.8}$$

Suppose that the assumptions of Fact 2.3 hold and arrive at (2.8). Then write $f_1(x) = f(x)$, $f_{h+1}(x) = g(x)$, which turns (2.8) into (2.6) for h replaced by $h+1$. Suppose that, furthermore, the assumptions of Fact 2.3 are satisfied for $h+1$ replacing h and for some $f_i(x)$ interchanging its roles with $f_1(x)$. Then we may repeat the same splitting process. Let us assume that this process has been recursively continued until we finally arrived at a product $\prod_{i=1}^{n}(z - z_i^*)$ and stopped. Then, by the virtue of Fact 2.3, the error norm of approximating $p(x)$ by this product was bounded by $\epsilon\|p(x)\|$. Furthermore, we have the following useful estimate.

FACT 2.4. (Compare [1, Section 4].) If $n > 0$, $p(x) = \prod_{i=1}^{h} f_i(x)$ has degree n, and all $f_i(x)$ are polynomials, then

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

By using Facts 2.3 and 2.4 in the above recursive splitting process, we easily deduce the following.

FACT 2.5. The inequality (2.7) holds for all h if $\epsilon_h \le \epsilon/(n2^n)$ for all h.

REMARK 2.1. Since $|z_i| \le 1$ for all i, the magnitudes of the coefficients of $p(x)$ are maximized for $p(x) = (x+1)^n$, so that $1 \le \|p(x)\| \le 2^n$. It is well known [58, III, Ch. 1, No. 31] that all the zeros of the k^{th} order derivative of $p(x)$, for any k, lie in the disc $D(0, |z_n|) \subseteq D(0,1)$ (compare (2.2)), and therefore, $p_{n-k,k} \le \|p^{(k)}(x)\| \le 2^{n-k}p_{n-k,k}$, where $p_{n-k,k} = (n!)/((n-k)!)$ is the leading coefficient of $p^{(k)}(x)$.

By combining the above estimates, we obtain the following.

COROLLARY 2.1. *It is sufficient to compute at first ϵ^*-splitting of $p(x)$ into two factors $F_k^*(x)$ and $G_{n-k}^*(x)$, for $\epsilon^* \le \epsilon/(n2^n)$, that is, for ϵ^* satisfying*

$$\log\left(\frac{1}{\epsilon^*}\right) \ge \hat{b} + n + \log n, \tag{2.9}$$

if $\epsilon = 2^{-\hat{b}}$ (see (1.6)), and then, recursively, ϵ^*-splittings of the factors, in order to compute an approximate factorization of $p(x)$ into linear factors $f_i(x) = x - z_i^*$, $i = 1, 2, \ldots, n$, satisfying (2.6) for $h = n$, which amounts to (1.6).

Ostrowski's well-known perturbation theorem [18] has the following extensions, which allow some further refinements (compare [59] and the simple bound $\|p(x)\| \le 2^n$ of our Remark 2.1).

FACT 2.6. [1, Section 19] For a polynomial $p(x)$ of (2.1),(2.2), the bound (1.6) implies the bound (1.2) if $\hat{b} \ge bn + n + 2$.

FACT 2.7. [60, Theorem 2.7] For a polynomial $p(x)$ of (2.1), let $p^*(x)$ be a monic polynomial of degree at most n satisfying $\|p^*(x) - p(x)\| \le \eta^n\|p(x)\|$, $\eta \le 1/128$. Then one may enumerate the zeros of $p^*(x)$ so that $p^*(x) = \prod_{i=1}^{n}(x - z_i^*)$, $|z_i^* - z_i| \le 9\eta$ if $|z_i| \le 1$, $|1/z_i^* - 1/z_i| \le 9\eta$ if $|z_i| \ge 1$.

Combining Fact 2.6 and Corollary 2.1 yields the following.

COROLLARY 2.2. *Under the assumptions of Corollary 2.1, the choice of $\epsilon^* = 2^{-\bar{b}}$, for $\bar{b}$ satisfying (1.1), suffices in order to compute approximations $z_1^*, \ldots, z_n^*$ to the zeros of $p(x)$ that satisfy (1.2).*

By recursively applying Fact 2.3 for $h = n$, $k = 1, \ldots, n$, $f_i(x) = x - z_i^*$, $i = 1, \ldots, k-1$, $f_i(x) = x - z_i$, $i = k, \ldots, n$, we obtain the following.

COROLLARY 2.3. *For a polynomial $p(x)$ of (2.1),(2.2), we have $\|p(x) - \prod_{i=1}^{n}(x - z_i^*)\| \le \epsilon\|p(x)\|$ if $|z_i - z_i^*| \le \epsilon/(n2^n)$ for all i.*

Due to Theorem 2.1 and Corollary 2.2, we may recursively factorize $p(x)$ and thus solve the problem of approximating polynomial zeros provided that we have an algorithm that computes well-isolated discs for splitting $p(x)$ into two factors, as well as for splitting every nonlinear factor computed in each step of the subsequent recursive splitting of these two factors. According to [1], we obtain the desired splitting disc for $p(x)$ (and similarly for its factors) by applying Facts 2.1 and 2.2. At first, we apply Fact 2.2 for the origin shifted to the center of gravity of the zeros, $-p_{n-1}/n = \sum_{i=1}^{n} z_i/n$. If $\max_i |z_i| \ge L \min_j |z_j|$ for $L > 1$, we immediately find a desired f-isolated splitting disc for $f = 1 + c/n$ for a positive c. Otherwise, we will twice apply Fact 2.1 in order to shift the origin into $X = 2\bar{r_n}$ and $X = 2\bar{r_n}\sqrt{-1}$, and after each shift we will apply the algorithm supporting Fact 2.2. It can be shown [1] that at least one of these two applications gives us a desired splitting disc. Recursive extension of the same process to the approximate factors of $p(x)$ (produced by splitting $p(x)$ over such a disc) finally outputs the desired approximations to all the zeros of $p(x)$ within the errors bounded according to (1.6) and, consequently, (1.2). The overall cost of these computations is bounded by $O_A((n \log n) \log(b \log n), n)$, in the general (worst) case (compare [2,29,46]). Furthermore, the latter cost bound decreases by the factor $n/\log n$ in the case where in all the recursive steps the splitting discs are (α, β)-balanced for a fixed pair (α, β), $0 < \alpha \le \beta < 1$. This has been achieved in [30,31,39] under the additional (strong) assumption that all the zeros of $p(x)$ are real. A similar decrease of the cost bound has been achieved in [2] for approximating a single complex polynomial zero.

Next, we will recall and improve the approach of [3] to ensure (α, β)-balanced recursive splitting for approximating all the zeros of any input polynomial.

3. CENTERED POINTS AND SPLITTING DISCS

DEFINITION 3.1. *(See Figure 1 and compare [3].) Given real s and t, $0 < t \le 1 \le s$, a set on the complex plane is called t-full for a polynomial $p(x)$ of (2.1) or simply t-full if it covers more than tn zeros of $p(x)$; a set on the complex plane is called (t, s)-centered for $p(x)$ if it has a nonempty intersection with the dilation $D(X, sr)$ of any t-full disc $D(X, r)$; a complex point Y is called a (t, s)-center for $p(x)$ or simply a (t, s)-center if $D(X, sr) \ni Y$ for every t-full disc $D(X, r)$; a set on the complex plane is called a (t, s)-cover (for $p(x)$) if it contains a (t, s)-center (for $p(x)$); such a set is called a full (t, s)-cover (for $p(x)$) if it contains all the existent (t, s)-centers (for $p(x)$).*

We immediately observe the following.

FACT 3.1. The disc $D(X, sr)$ is a full (t, s)-cover for $p(x)$ if the disc $D(X, r)$ is t-full.

DEFINITION 3.2. *(Compare [3].) The ratio $r/|X|$, for a complex X and a positive r, is called the relative radius of a disc $D(X, r)$. The ratio $R/r > 1$ of the radii, R and r, of the two boundary circles of an annulus is called its relative width.*

FACT 3.2. A disc covers the origin if and only if its relative radius is not less than 1.

FACT 3.3. [3] If $t \ge 1/2$ and if a set S is $(t, s-2)$-centered for $p(x)$, then this set contains a (t, s)-center for $p(x)$.

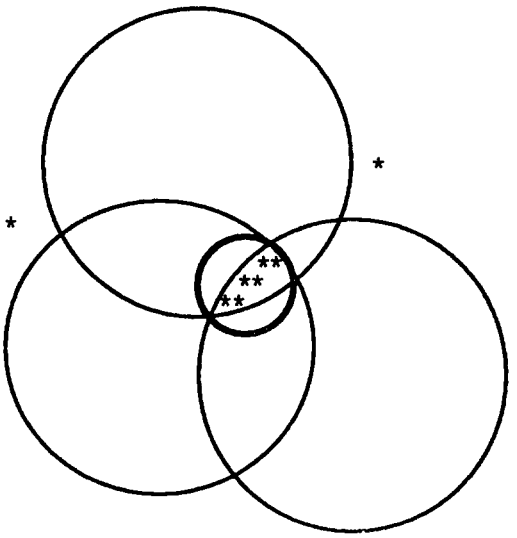


Figure 1. $p(x)$ has eight zeros marked by asterisks. All the $(3/4, 1)$-centered points for $p(x)$ lie in the smaller disc circumscribing the intersection of three larger discs.

PROOF. [3] Let $D(X, r)$ be a t-full disc for $p(x)$ of the minimum radius and let Z be a point of S lying in the disc $D = D(X, (s-2)r)$. Let $D(Y, R)$ be any other t-full disc for $p(x)$. Then, $R \geq r$, and since $t \geq 1/2$, this disc intersects $D(X, r)$. Therefore, the disc $D(Y, sR)$ covers the disc D and, consequently, the point Z, which is, therefore, a (t, s)-center for $p(x)$. ∎

To relate the above concepts to balanced splitting, write

$$f = 1 + \frac{c}{n}, \tag{3.1}$$

$$g(a) = \left\lfloor \frac{(1-a)n}{2} \right\rfloor, \qquad h(a) = g(a) + \lfloor an \rfloor + 1, \tag{3.2}$$

fix a, c, and s such that $f > 1 > a$ and both a and f are close enough to 1 (we will specify this assumption about a and f later on; in particular, one may set $a = 5/6$, $f = 1 + 1/(100n)$, according to (5.2) and (6.9)), and apply the algorithm supporting Fact 2.2. Then shift the origin into $2\bar{r}_{h(a)}$ and $2\bar{r}_{h(a)}\sqrt{-1}$ (see Fact 2.1 for $t = 1$) and after each shift apply the same algorithm again. Consider these three applications as three stages of an algorithm to be referred to as Algorithm 3.1 (see Figure 1). The computational cost of performing this algorithm is bounded by $O_A((\log n)^2, n)$, due to Facts 2.1 and 2.2. Now, we deduce the following result.

PROPOSITION 3.1. *Let Algorithm 3.1 be applied for $a > 2/3$. Let $c > 0$ and $s \geq 1$ be two fixed values, and let f be defined by (3.1). Then, at a cost bounded by $O_A(1, n)$, in addition to the cost of performing Algorithm 3.1, bounded by $O_A((\log n)^2, n)$, one can compute either*

(a) *an (a, f)-splitting disc SD for $p(x)$ (that is, a disc that is both $((1-a)/2, (1+a)/2)$-balanced and f-isolated), or otherwise*

(b) *a complex X and a positive r such that the disc $D(X, r)$ is $(3a-2)$-full for $p(x)$ and has a relative radius of at most 8δ; moreover,*

$$|X| \geq \underline{r}_{g(a)} \geq \frac{\bar{r}_{h(a)}}{(1+\delta)}, \tag{3.3}$$

$$r \leq 8\delta \underline{r}_{g(a)} \leq 8\delta |X|, \tag{3.4}$$

for $\underline{r}_{g(a)}$ and $\bar{r}_{h(a)}$ computed in Stage 1 of Algorithm 3.1 and for

$$\delta = \delta(a, f, n) = f^{3\lfloor an \rfloor + 3} - 1. \tag{3.5}$$

PROOF. (Compare the proof of Theorem 2.1 of [3].) Consider the discs $D(0, \bar{r}_k) = \{x : |x| \le \bar{r}_k\}$ for $g(a) \le k < h(a)$. Due to (3.2), all these discs are $((1-a)/2, (1+a)/2)$-balanced. First suppose that $f^{3(h(a)-g(a))}\underline{r}_{g(a)} = f^{3\lfloor an \rfloor + 3}\underline{r}_{g(a)} < \bar{r}_{h(a)}$ (see Figure 2). Then, clearly, there exists an integer k such that $g(a) \le k < h(a)$, $f^3\underline{r}_k < \bar{r}_{k+1}$. Due to Fact 2.2, we may assume that $f\underline{r}_i > \bar{r}_i$, for $i = k$ and $i = k+1$, and obtain that $f|z_k| \le f\bar{r}_k < \underline{r}_{k+1} \le |z_{k+1}|$, so that the disc $D(0, \bar{r}_k)$ is f-isolated, and we may set $SD = D(0, \bar{r}_k)$.

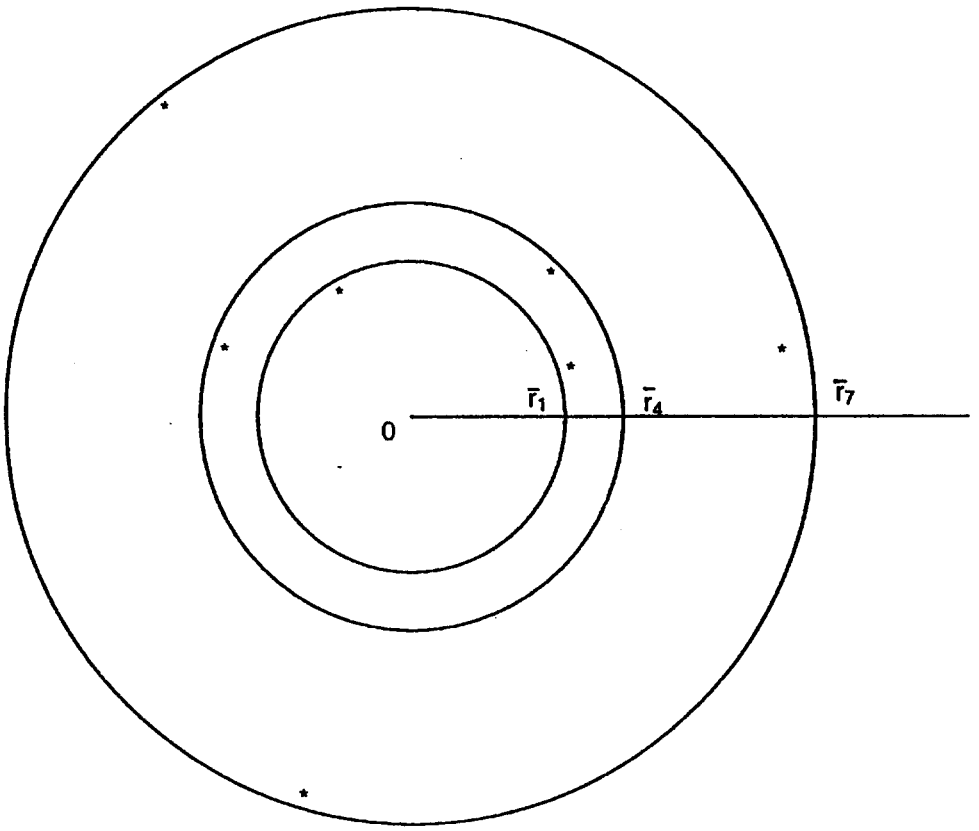


Figure 2. Case (a) of Proposition 3. $f = 1.02$, $a = 3/4$, $n = 8$, $g(a) = 1$, $h(a) = 7$, $3(h(a) - g(a)) = 18$, $\bar{r}_7/\underline{r}_1 > f^{18}$. The disc $D(0, \bar{r}_4)$ is f-isolated.

Now assume the opposite case (see Figure 3), where

$$f^{3(h(a)-g(a))} = f^{3\lfloor an \rfloor + 3} \ge \frac{\bar{r}_{h(a)}}{\underline{r}_{g(a)}}. \tag{3.6}$$

Equation (3.6) bounds the relative width, $\bar{r}_{h(a)}/\underline{r}_{g(a)}$, of the a-full annulus

$$A\left(0, \underline{r}_{g(a)}, \bar{r}_{h(a)}\right) = \left\{x : \underline{r}_{g(a)} \le |x| \le \bar{r}_{h(a)}\right\}. \tag{3.7}$$

By repeating the same argument for Stages 2 and 3 of Algorithm 3.1, we either arrive at a desired (a, f)-splitting disc SD or else at three annuli of relative widths of at most $f^{3h(a)-3g(a)}$ (compare (3.6),(3.7)), each annulus being a-full for $p(x)$. Geometric considerations show that the intersection I of these three annuli can be included into a readily computable disc $D(X, r) = \{x : |x - X| \le r\}$ with X and r satisfying (3.3)–(3.5) (see Figure 3). On the other hand, a simple argument (see Appendix A) shows that the intersection I of these three annuli and, therefore, also the disc $D(X, r) \supseteq I$ are $(3a-2)$-full for $p(x)$. ∎

Hereafter, we will cite the extension of Algorithm 3.1 supporting Proposition 3.1 as Algorithm 3.2. According to Proposition 3.1, Algorithm 3.2 is performed at a cost of $O_A((\log n)^2, n)$ and outputs either an (a, f)-splitting disc for $p(x)$ or a disc $D(X, r)$ that is $(3a-2)$-full for

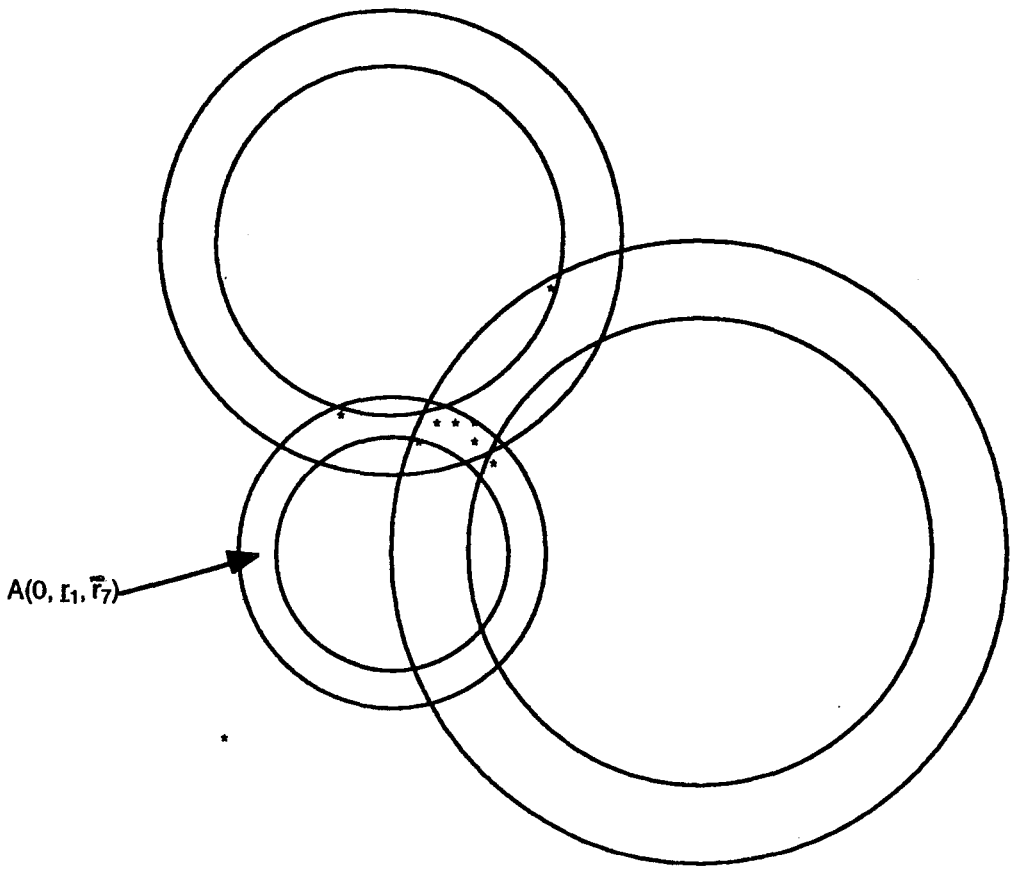


Figure 3. Case (b) of Proposition 3.1.

$p(x)$, has a small relative radius $r/|X|$, and, moreover, has a center X and a radius r satisfying (3.3)–(3.5).

REMARK 3.1. The proof of Proposition 3.1 can be modified so as to decrease the value δ of (3.5), to a level close to $f^{\lfloor an \rfloor + 1} - 1$.

REMARK 3.2. Equations (3.4) and (3.5) relate the bounds on the relative radius $r/|X|$ of the disc $D(X, r)$ and on the isolation ratio f of the disc SD (one of these two discs being output by Algorithm 3.2). Namely, (3.4) and (3.5) imply that $8|X|/r \geq 1/\delta = 1/(f^{3\lfloor an \rfloor + 3} - 1)$ and, consequently, $f^{3\lfloor an \rfloor + 3} \geq 1 + r/(8|X|)$. Therefore, $(3\lfloor an \rfloor + 3) \ln f \geq \ln(1 + r/(8|X|)) = -\sum_{i=1}^{\infty} (-r/(8|X|))^i / i$. Hence, assuming that $8|X| > r$, we obtain that

$$(3\lfloor an \rfloor + 3) \ln f \geq \frac{r}{8|X|} - \frac{r^2}{2(8|X|)^2} = \left(1 - \frac{r}{16|X|}\right) \frac{r}{8|X|},$$

$$f \geq \exp\left(\frac{1 - r/(16|X|)}{3\lfloor an \rfloor + 3} \frac{r}{8|X|}\right) \geq 1 + \frac{1 - r/(16|X|)}{3\lfloor an \rfloor + 3} \frac{r}{8|X|}.$$

Consequently,

$$\frac{1}{(f-1)} = O\left(\frac{n|X|}{r}\right), \tag{3.8}$$

as $n \to \infty$, $|X|/r \to \infty$.

4. TO THE ZEROS OF A POLYNOMIAL VIA THE ZEROS OF ITS HIGHER ORDER DERIVATIVE AND HOW TO HANDLE MASSIVE CLUSTERS OF THE ZEROS

The next result from [27] extends Rolle's well-known theorem to the complex case.

THEOREM 4.1. *[27] The set of the $n-l+1$ zeros of the $(l-1)^{st}$ order derivative $p^{(l-1)}(x)$ of $p(x)$ is an $((l-1)/n, s-2)$-centered set for $p(x)$ if*

(a) *$s \geq 2+1/\sin(\pi/(n-l+1))$ and $l \leq n-1$ (which holds if $s \geq 2+(n-l+1)/\pi$ for larger $n-l$) and, moreover, even if*

(b) *$2 \leq l \leq n-1$ and $s \geq 2+c\max\{(n-l+1)^{1/2}l^{-1/4}, (n-l+1)l^{-2/3}\}$ for some constant c (which already holds where $s = O(n^{1/3})$ as $n \to \infty$, provided that $l/n > \phi > 0$ for some fixed constant $\phi < 1$).*

Combining Fact 3.3 and Theorem 4.1 gives us the following corollary.

COROLLARY 4.1. *If $l > n/2$ and if s satisfies the assumptions of parts (a) and/or (b) of Theorem 4.1, then at least one of the $n-l+1$ zeros of the $(l-1)^{st}$ derivative of $p(x)$ is an $((l-1)/n, s)$-center for $p(x)$.*

Hereafter, we will write

$$l = \lfloor (3a-2)n \rfloor + 1, \qquad n-l = \lceil (3-3a)n \rceil - 1 \tag{4.1}$$

and will fix a in the semiopen interval

$$\frac{5}{6} \leq a < 1, \tag{4.2}$$

which means that $l > n/2$. In particular, one may choose

$$a = \frac{5}{6}, \qquad l = \left\lfloor \frac{n}{2} \right\rfloor + 1, \qquad n-l = \left\lceil \frac{n}{2} \right\rceil - 1. \tag{4.3}$$

We will also assume that s satisfies the assumptions of parts (a) and/or (b) of Theorem 4.1. (It suffices for us to use part (a) of this theorem; a simple proof of this part is recalled from [27] in our Appendix B. In fact, even weaker upper bounds on s, such as $\log s = O(\log n)$, would have sufficed for us in this paper.)

Next, examine the case where Algorithm 3.2 outputs no (a, f)-splitting disc for $p(x)$ and where

$$s < \frac{1}{(8\delta)}. \tag{4.4}$$

In this case, (3.4) and Fact 3.2 imply that the origin lies outside the disc $D(X, sr)$ (and therefore, cannot be a $(3a-2, s)$-center for $p(x)$).

Now suppose that the set $Z = Z_{l-1}$ of all the zeros of $p^{(l-1)}(x)$, for $l = 1 + \lfloor (3a-2)n \rfloor > n/2$ of (4.1) and for $a \geq 5/6$ of (4.2), is available. (According to Theorem 4.1, this set is a $(3a-2, s)$-cover for $p(x)$.) Choose f such that δ of (3.5) satisfies (4.4) and then apply Algorithm 3.2 $|Z|$ times (successively or concurrently); namely, apply it after shifting the origin into each of the $|Z| \leq n-l+1$ points of Z. Since the set Z is a $(3a-2, s)$-cover for $p(x)$ (due to Theorem 4.1), in at least one of these $|Z|$ applications of Algorithm 3.2, an (a, f)-splitting disc for $p(x)$ is output, where $1/(f-1) = O(sn)$, $f = 1 + c/(sn)$, for a constant c.

In [3] a policy of recursive shifts of the origin is proposed that enables us to apply the divide-and-conquer method to the given set Z of all the zeros of $p^{(l-1)}(x)$, thus reducing the number of required applications of Algorithm 3.2 to at most $\lceil \log |Z| \rceil$. Specifically, observe that the disc $D(X, sr)$, for X and r of part (b) of Proposition 3.2, has a relative radius less than 1. Due to Fact 3.2, such a disc does not contain the origin. Consequently, it lies entirely in at least one of the four half-planes, each bounded by the real or imaginary axis, that is, $\{x : \operatorname{Re} x > 0\}$, $\{x : \operatorname{Re} x < 0\}$, $\{x : \operatorname{Im} x > 0\}$, $\{x : \operatorname{Im} x < 0\}$. Since the disc $D(X, r)$ of Proposition 3.1 is $(3a-2)$-full for $p(x)$, the disc $D(X, sr)$ is a full $(3a-2, s)$-cover for $p(x)$, due to Fact 3.1.

Therefore, any zero of $p^{(l-1)}(x)$ lying in the complementary half-plane cannot be a $(3a-2,s)$-center for $p(x)$ and should be discarded. In [3], a shift of the origin is defined for which at least $|Z|/2$ zeros of $p^{(l-1)}(x)$ lie in such a half-plane. Specifically, according to the recipe of [3], one should first compute (at a cost $O_A(\log|Z|, |Z|/\log|Z|)$, $|Z| \le n-l+1$) a quasi-median point $\mu(Z)$ (not necessarily lying in Z), whose real and imaginary coordinates are given by the two medians of the two sets or multisets formed by all the real and all the imaginary coordinates of all points of Z, respectively. (When we define the medians, we count m times each common coordinate of exactly m points of Z.) Then one should shift the origin into $\mu(Z)$ and apply Algorithm 3.2, which will either output an (a,f)-splitting disc, where $1/(f-1) = O(sn)$ due to (3.4) and (3.8), or will enable us to discard at least $|Z|/2$ zeros of $p^{(l-1)}(x)$. Proceeding recursively, one will compute a desired (a,f)-splitting disc in at most $\lceil \log|Z| \rceil \le \lceil \log(n-l+1) \rceil$ applications of Algorithm 3.2, at an overall cost $O_A((\log n)^3, n)$.

In the next sections, we will extend the above construction of [3] so as to increase the isolation ratio f of the computed (a,f)-splitting disc from the level $1 + c/(sn)$ to or above the level $f = 1 + c/n$. Now suppose that an (a,f)-splitting disc with such an isolation ratio f has been computed, with no increase of the asymptotic complexity bounds. Then we may apply part (d) of Theorem 2.1, for appropriate B^* and $\bar{b}$, and split $p(x)$ over this disc, at a cost bounded by $O_A((\log n)t_{2,1}(\bar{B},n)n,1)$. The splitting reduces the original problem for $p(x)$ to ones for its two factors. Taking into account the computational cost of this reduction, which includes the cost of approximating the zeros of $p^{(l-1)}(x)$, we arrive at the inequality

$$A(n) \le A(n-l+1) + A(n_1) + A(n_2) + O\left(nt_{2,1}\left(\bar{B},n\right)\log n\right),$$

where $t_{2,1}(\bar{B},n) = (\log n)^2 + \log\bar{B}$, $n_1+n_2 = n$, $\max(n_1,n_2) = (1+a)n/2$, $n-l+1 = \lceil (3-3a) \rceil < n/2+1$, and $A(k)$ denotes the number of arithmetic operations required for approximating the zeros of a k^{th} degree polynomial, within an appropriate error bound (compare Corollaries 2.1 and 2.2), provided that all these zeros lie in the disc $D(0,1)$. (According to Remark 2.1, the zeros of $p^{(l-1)}(x)$ lie in the disc $D(0,|z_n|)$.) Recursive application of similar bounds on $A(h)$, for $h = n-l+1$, $h = n_1$, and $h = n_2$, implies (see [3]) approximating polynomial zeros in arithmetic time $O(n^{1+\tilde{\epsilon}}\log b)$ for any fixed positive $\tilde{\epsilon}$.

REMARK 4.1. For f of the order $1+c/(sn)$, the above inequality for $A(n)$ changes into the bound

$$A(n) \le A(n - \lceil (3a-2)n \rceil + 1) + A(n_1) + A(n_2) + O\left(nst_{2,1}\left(\bar{B},n\right)\log n\right)$$

(compare Remarks 4.2 and 9.2), whose recursive extension using a respective extension of Theorem 2.1 only gives us a larger bound, $A(n) = O(n^{1+\bar{\epsilon}}s\log b)$ for a positive $\bar{\epsilon}$.

In fact, we need to modify both of the above algorithms for the computation of a splitting disc since we actually only approximate the zeros of $p^{(l-1)}(x)$ but do not compute them. What is more serious, we also need to avoid the severe numerical problems that arise if we try to compute a *balanced* splitting of $p(x)$ in the case where all or almost all of the zeros of $p(x)$ form a *massive cluster* lying in a very small disc, $D(X,\sigma)$. In this case, in order to determine an (a,f)-splitting disc for $p(x)$, one has to separate some zeros of the clusters from each other, which requires computations with a very high precision, of the order $\log(1/\sigma)$. For smaller σ, this precision can be too high to be compatible with the complexity bounds of Theorem 2.1. To avoid such problems (not addressed in [3]), we will not seek balanced splitting whenever we can compute a sufficiently small disc containing sufficiently many zeros of $p(x)$. More specifically, we will complement the algorithms for the computation of (a,f)-splitting discs by a block that identifies and removes all the factors of $p(x)$ of the form $\prod_{i=1}^{k}(x-z_i^*)$, where $k \ge \lceil (3a-2)n \rceil$ and $|z_i^* - X| < 2^{-B}$, $i = 1,\dots,k$, for some complex X and a fixed positive B. We will use the following definition.

DEFINITION 4.1. *A disc $D(X,\rho)$ is called an (a,B,f)-splitting disc for $p(x)$ if it is both f-isolated and $(3a-2)$-full for $p(x)$ (compare Definition 3.1) and if ρ satisfies the relations*

$$2^{-B^*} = \frac{2^{-B}}{f^{2\lceil(3-3a)n\rceil-2}} \le \rho \le 2^{-B}. \tag{4.5}$$

A disc $D(X,r)$ is called an (a,B^*)-disc if it is $(3a-2)$-full and if

$$r \le 2^{-B^*}. \tag{4.6}$$

FACT 4.1. If we are given B and B^* satisfying the equation of (4.5) and if a given disc $D(X,r)$ is an (a,B^*)-disc, then there exists ρ satisfying (4.5) such that the disc $D(X,\rho)$ is an (a,B,f)-splitting disc for $p(x)$. Moreover, such a value ρ can be computed at a cost bounded by $O_A((\log n)^2,n)$.

PROOF. By assumption, the disc $D(X,r)$ is $(3a-2)$-full for $p(x)$. Therefore (due to (4.6)), the exterior of the disc $D(X,2^{-B^*})$ contains at most $n-\lfloor(3a-2)n\rfloor-1 = \lceil(3-3a)n\rceil-1$ zeros of $p(x)$. Consequently, there exist values ρ satisfying (4.5) and such that the disc $D(X,\rho)$ is f^2-isolated. We only need to compute ρ satisfying (4.5) for which the disc $D(X,\rho)$ is f-isolated. We obtain such a value ρ by applying the algorithm that supports Fact 2.2, where we require sufficiently small relative error bound; for instance, the bound $0.5(f-1)$ will suffice. This gives us a desired f-isolated disc $D(X,\rho)$, at a cost bounded by $O_A((\log n)^2,n)$. ∎

As soon as we obtain an (a,B,f)-splitting disc $D(X,\rho)$, for $f \ge 1+c/n$ and for a fixed positive c, we may recall Theorem 2.1 and compute an ϵ-splitting of $p(x)$ over this disc, for a fixed small ϵ (see (2.4)). If

$$B \ge b, \qquad \rho \le 2^{-B} \le 2^{-b}, \tag{4.7}$$

then the center X of the disc $D(X,\rho)$ approximates (within the error bound 2^{-b} of (1.2)) all the k zeros of $p(x)$ lying in this disc. The remaining $n-k$ zeros of $p(x)$ are approximated by the zeros of $G^*_{n-k}(x)$. If $n-k=o(n)$, that is, if the zeros form a massive cluster, then the splitting is unbalanced, but the entire computation is only simplified.

We may satisfy the assumption (4.7), without choosing an extremely large B. Then, for ρ bounded from below according to (4.5), we may *keep the precision of the computation reasonably well bounded from above*, as required in order to prove Theorem 1.1.

The above analysis suggests a simple extension of Algorithm 3.2, hereafter referred to as Algorithm 4.1. Namely, one should always apply the algorithm supporting Fact 4.1 and output an (a,B,f)-splitting disc $D(X,\rho)$ as soon as (in the process of performing Algorithm 3.2) one arrives at an (a,B^*)-disc $D(X,r)$ for a, B, B^*, and r satisfying (4.6) and the equation of (4.5).

Moreover, we will also modify the algorithms of [3] for computing an (a,f)-splitting disc, recalled earlier in this section. Now, we will aim either at an (a,f)-splitting disc for $p(x)$ or (in the case where the zeros of $p(x)$ form a massive cluster) at an (a,B,f)-splitting disc for $p(x)$, for a fixed B. We will achieve our goal based on the following lemma.

LEMMA 4.1. *Suppose that a $(3a-2,s)$-center for $p(x)$ lies in a disc $D(0,\rho^*)$. Suppose that application of Algorithm 4.1 does not give us an (a,f)-splitting disc SD but yields a disc $D(X,r)$ of part (b) of Proposition 3.1, which is $(3a-2)$-full for $p(x)$. Then*

$$r \le \frac{\delta^*\rho^*}{(1-\delta^* s)}, \tag{4.8}$$

where

$$\delta^* = 8\delta \tag{4.9}$$

and δ satisfies (3.5) and (4.4).

PROOF. A $(3a-2,s)$-center for $p(x)$ lies in both discs $D(X,sr)$ and $D(0,\rho^*)$. Therefore, these two discs have a nonempty intersection, and hence $|X| \le sr+\rho^*$. Combining the latter bound, (4.9), and (3.4) yields (4.8). ∎

Now, we are prepared to devise a desired algorithm for the computation of a splitting disc for $p(x)$.

ALGORITHM 4.2.

INPUT: Polynomial $p(x)$ of (2.1), a complex Y, real B^* and B, and positive a, f, and ρ^*, satisfying the relations (4.2), (4.5), $f>1$, and

$$\rho^* \le 2^{-B^*}\frac{(1-\delta^* s)}{\delta^*} \tag{4.10}$$

(for δ^* of (4.9), δ of (3.5), and s of Theorem 4.1) and such that the disc $D=D(Y,\rho^*)$ is a $(3a-2,s)$-cover for $p(x)$, that is, covers at least one $(3a-2,s)$-center for $p(x)$. (For instance, an approximation within ρ^* to at least one of the zeros of $p^{(l-1)}(x)$, for $l=\lfloor(3a-2)n\rfloor+1$, may serve as the point Y, due to Theorem 4.1.)

OUTPUT: An (a,f)-splitting disc D for $p(x)$ or an (a,B,f)-splitting disc $D(X,\rho)$ for $p(x)$.

COMPUTATIONS. Shift the origin into the point Y and apply Algorithm 4.1. If the algorithm computes an (a,f)-splitting disc D for $p(x)$ or an (a,B,f)-splitting disc $D(X,\rho)$, output this disc and stop. Otherwise, as we will show in the correctness proof below, Algorithm 4.1 outputs X and r satisfying (3.4) and such that the disc $D(X,r)$ is $(3a-2)$-full for $p(x)$, that is, contains at least $\lfloor(3a-2)n\rfloor+1$ zeros of $p(x)$. Then shift the origin into X. Apply the algorithm supporting Fact 2.2 so as to compute ρ such that the disc $D(X,\rho)$ is f-isolated and its radius ρ satisfies (4.5). Output this disc and stop.

CORRECTNESS PROOF. Due to Lemma 4.1, we have (4.8). The bounds (4.8) and (4.10) together imply (4.6). Now, correctness of the algorithm follows as in the proof of Fact 4.1. ∎

If we have approximations Y_i, within the error bound ρ^* of (4.10), to all the $n-l+1$ zeros of $p^{(l-1)}(x)$, for $l=\lfloor(3a-2)n\rfloor+1$, of (4.1), $i=1,\dots,n-l+1$, then we may compute an (a,f)-splitting disc or an (a,B,f)-splitting disc for $p(x)$ by applying Algorithm 4.2 at every point Y_i, until a desired splitting disc is output. This computation will be called Algorithm 4.3. One may improve it by incorporating the divide-and-conquer approach from [3], which we recalled earlier, so that at most $\lceil\log|S|\rceil$ calls for Algorithm 4.1 will be needed. We will cite this modification as Algorithm 4.4.

REMARK 4.2. The relative radius $r/|X|$ of the disc $D(X,r)$ of Proposition 3.1 must satisfy the bound $r/|X|<1/s$ in order to ensure that the disc $D(X,sr)$ does not cover the origin (compare Fact 3.2 and the relations (3.4) and (4.4)). On the other hand, Algorithms 4.3 and 4.4 define (a,f)-splitting or (a,B,f)-splitting discs for $f-1$ of the order $r/(|X|n)$ (compare (3.8)); that is, we should deal with the case where

$$f=1+\frac{c}{(ns)} \tag{4.11}$$

for a positive constant c (compare the relations (3.4), (3.8), and (4.4)). In the next section, we will modify this construction so as to proceed with larger relative radii $r_i/|X_i|$, satisfying $|X_i|/r_i=O(1)$. According to the equation (3.8) applied for $X=X_i$, $r=r_i$, this will enable us to increase f so as to satisfy the bound

$$\frac{1}{(1-f)}=O(n). \tag{4.12}$$

Under (4.12) we may invoke Theorem 2.1 in order to estimate the cost of splitting $p(x)$ over the computed f-isolated discs (compare (6.1), (6.4)–(6.9), Remarks 5.1, 6.1, and 9.2).

5. RECURSIVE CONTRACTION OF A REGION COVERING ALL THE $(3a-2,s)$-CENTERS FOR $p(x)$

The next recursive extension of Algorithm 4.1 will enable us to ensure a stronger isolation of the output splitting disc, so as to raise its isolation ratio f to the level (4.12), from (4.11) (compare Remarks 4.2 and 5.1).

ALGORITHM 5.1. (See Figure 4.) Fix a real a, in the interval (4.2), and a positive integer H and recursively apply Algorithm 4.1; for every i, $i=1,\dots,H$, after the i^{th} application, shift the origin into the center $X=X_i$ of the output disc $D(X,r)=D(X_i,r_i)$, for r and X satisfying (3.4). If some application of Algorithm 4.1 defines an (a,f)-splitting disc SD for $p(x)$, then output this disc and stop the computations. Otherwise, stop in H recursive applications of Algorithm 4.1 and output H, the center X_H, and the radius r_H of the disc $D(X_H,r_H)$ computed in the last application of Algorithm 4.1 (such a disc must be $(3a-2)$-full for $p(x)$). Represent the output point X_H in the original coordinates (used before the first shift of the origin at the first of the recursive applications of Algorithm 4.1) or output $\sum_{i=1}^{H}X_i$ if each X_i is defined relative to the latest shift of the origin.

PROPOSITION 5.1. *Suppose that Algorithm 5.1 has output H, X_H, and r_H (with X_H defined in the original coordinates), rather than an (a,f)-splitting disc for $p(x)$, so that the disc $D(X_H,r_H)$ is $(3a-2)$-full for $p(x)$. Then the following relations hold:*

(a) $|X_H|\ge|X_1|(1-\hat\delta/(1-\hat\delta))=|X_1|(1-2\hat\delta)/(1-\hat\delta)$,

(b) $r_H\le 8\delta\hat\delta^{H-1}|X_1|=\hat\delta^H|X_1|/(1+\delta)$,

and, consequently, the relative radius $r_H/|X_H|$ of the disc $D(X_H,r_H)$ is bounded as follows:

(c) $r_H\le 8|X_H|\delta\hat\delta^{H-1}(1-\hat\delta)/(1-2\hat\delta)=|X_H|\hat\delta^H(1-\hat\delta)/((1+\delta)(1-2\hat\delta))$,

where $\delta=\delta(a,f,n)=f^{3\lfloor an\rfloor+3}-1$, according to (3.5), and

$$\hat\delta=8(1+\delta)\delta. \tag{5.1}$$

PROOF. Proposition 5.1 follows since the radii r_i and, consequently, the relative radii $r_i/|X_i|$ of the discs $D(X_i,r_i)$ rapidly decrease as i grows (see Figure 4). It remains to specify the related estimates. Let $r_j^{(i)}$ denote the distance from X_i to a j^{th} closest zero of $p(x)$ and let $\underline{r}_j^{(i)}$ and $\bar r_j^{(i)}$ denote the lower and upper bounds of $r_j^{(i)}$, respectively, obtained by means of applying Facts 2.1 and 2.2. We first recall that

$$\underline{r}_{g(a)}^{(i)}\ge\frac{\bar r_{h(a)}^{(i)}}{(1+\delta)},\qquad i=1,\dots,H, \tag{5.2}$$

since otherwise the i^{th} application of Algorithm 4.1 would have output of a disc SD. Furthermore, we have the bounds

$$\underline{r}_{g(a)}^{(i)}\le r_{g(a)}^{(i)}\le r_i,\qquad i=1,\dots,H, \tag{5.3}$$

since the disc $D(X_i,r_i)$ is $\lfloor(3a-2)\rfloor$-full for $p(x)$ and since $g(a)=\lfloor(1-a)n/2\rfloor\le(3a-2)n$ for $a\ge 5/6$. Moreover,

$$|X_{i+1}-X_i|\le\bar r_{h(a)}^{(i)},\qquad i=1,\dots,H-1, \tag{5.4}$$

since X_{i+1} lies in the annulus that is output in the first stage of the $(i+1)^{\text{st}}$ application of Algorithm 4.1 (within Algorithm 5.1). By combining (5.2)–(5.4), we deduce that

$$|X_{i+1}-X_i|\le(1+\delta)r_i,\qquad i=1,\dots,H-1. \tag{5.5}$$

On the other hand, we extend (3.4) to the i^{th} application of Algorithm 4.1 and obtain that

$$r_i\le 8\delta|X_i-X_{i-1}|,\qquad i=1,\dots,H, \tag{5.6}$$

where $X_0 = 0$. By combining (5.5) and (5.6), we obtain that

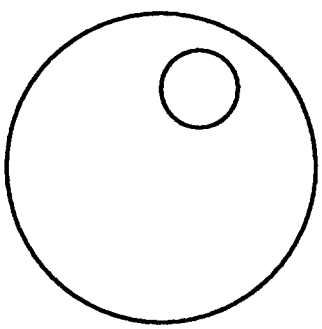
$$r_i \le \hat{\delta} r_{i-1}, \quad |X_{i+1} - X_i| \le \hat{\delta}|X_i - X_{i-1}|, \qquad i = 1, \dots, H-1,$$

for $\hat{\delta} = 8(1+\delta)\delta$ of (5.1). Then, by combining the latter bounds for $i = 1, \dots, j$ and recalling that $X_0 = 0$, we deduce that

$$r_j \le \hat{\delta}^j r_1, \quad |X_{j+1} - X_j| \le |X_1 - X_0|\hat{\delta}^j = |X_1|\hat{\delta}^j, \qquad j = 1, \dots, H-1, \tag{5.7}$$

$$|X_H - X_1| \le \sum_{j=1}^{H-1} |X_{j+1} - X_j| \le |X_1| \sum_{j=1}^{H-1} \hat{\delta}^j < |X_1| \frac{\hat{\delta}}{\left(1 - \hat{\delta}\right)}.$$

Therefore, $|X_H| > |X_1|(1-\hat{\delta}/(1-\hat{\delta}))$, which proves part (a) of Proposition 5.1. On the other hand, substitution of (5.7) (for $j = H-1$) into (5.6) (for $i = H$) proves part (b) of Proposition 5.1. ∎



* Origine (0)

Figure 4. The discs $D(X_1, r_1)$ and $D(X_2, r_2)$ of Algorithm 5.1 are represented by the two discs (larger and smaller ones) in this figure.

Due to Proposition 5.1, Algorithm 5.1 outputs either an (a, f)-splitting disc SD for $p(x)$ or a disc $D(X_H, r_H)$ that is $(3a-2)$-full for $p(x)$ and has a relative radius $r_H/|X|$ satisfying the upper bound of part (c) of Proposition 5.1. Due to Fact 3.1, the disc $D(X_H, r_H s)$ is a full $(3a-2, s)$-cover for $p(x)$.

REMARK 5.1. Due to part (c) of Proposition 5.1, we may ensure the bound $r_H/|X_H| < 1/s$ on the relative radius of the output disc $D(X_H, r_H)$ of Algorithm 5.1 already for $1/\hat{\delta} = O(1)$, $H = O(\log s)$. This will enable us to achieve (4.12) by using Algorithm 5.1, instead of relying just on Algorithm 4.1 and arriving at (4.11) (compare (6.1), (6.4)–(6.9) and Remark 6.1).

6. WE DO NOT NEED TO APPROXIMATE THE ZEROS OF HIGHER ORDER DERIVATIVES

Seeking a $(3a-2,s)$-center for $p(x)$ by means of applying Algorithms 4.3 or 4.4, we recursively split, at first the $(l-1)^{\text{st}}$ order derivative $p^{(l-1)}(x)$, for $l=\lfloor(3a-2)n\rfloor+1$, of (4.1), and then both its factors, over some available splitting discs.

We will next show how to avoid splitting one of the two factors. Let $v(x)$ denote the polynomial $p^{(l-1)}(x)$ or its factor and suppose that $v(x)$ has been split over some disc $D(0,\hat{r})$, which is f_v-isolated for $v(x)$. (Letting the origin be the center of this disc is no loss of generality, due to Fact 2.1.) Apply Algorithm 5.1. We only need to consider the case where the output disc $D(X_H,r_H)$ is $(3a-2)$-full for $p(x)$. Then the disc $D(X_H,sr_H)$ is a full $(3a-2,s)$-cover for $p(x)$, due to Fact 3.1. Therefore, in our search for a $(3a-2,s)$-center for $p(x)$, we may discard all the zeros of $v(x)$ lying outside the latter disc. Suppose that $2sr_H<(f_v-1)\hat{r}$, that is, the diameter of the disc $D(X_H,sr_H)$ is less than the width of the annulus $\{x:\hat{r}<|x|<f_v\hat{r}\}$ surrounding the disc $D(0,\hat{r})$ and free of the zeros of $v(x)$ (compare Figure 5). Then the disc $D(X_H,sr_H)$ cannot simultaneously intersect both of the disc $D(0,\hat{r})$ and the exterior of the disc $D(0,f_v r)$, so we may determine which one of the two computed factors of $v(x)$ has no zeros in $D(X_H,sr_H)$ and can be discarded.

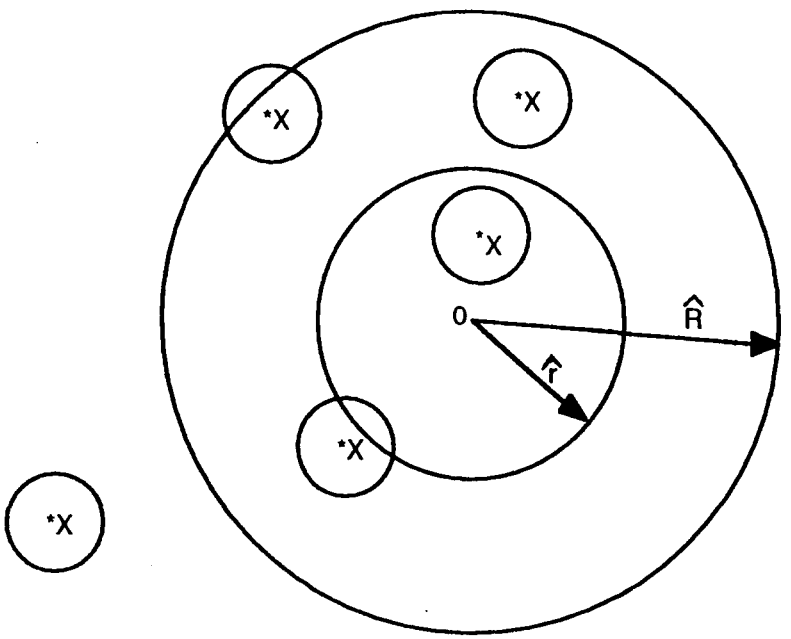


Figure 5. Five positions of a smaller disc $D(X,r)$ of Lemma 6.1 relative to the annulus $\{x:\hat{r}\le|x|\le\hat{R}\}$.

We will next formalize our argument as an algorithm and then will show that the value $2sr_H$ can be decreased below $(f_v-1)\hat{r}$, for a fixed $\hat{r}$, for $f_v=1+c/n$, for a fixed positive c, and for H of the order $\log n$ (but not for $H=1$ or even $H=O(1)$, compare Remark 6.1).

ALGORITHM 6.1.

INPUT: Polynomials $p(x)$ and $v(x)$ and five real values, that is, a of (4.2), s of part (a) or part (b) of Theorem 4.1, $f_v>1$, $f>1$, $\hat{r}>0$, such that the disc $D(0,\hat{r})$ is an f_v-isolated disc for $v(x)$, and B.

OUTPUT: Either an (a,f)-splitting disc for $p(x)$, or an (a,B,f)-splitting disc for $p(x)$, or, otherwise, a disc $D(X_H,r_H)$ and an integer, 0 or 1; 1 is output if the disc $D(0,\hat{r})$ contains no $(3a-2,s)$-centers for $p(x)$; 0 is output if the exterior of the disc $D(0,f_v\hat{r})$ contains no $(3a-2,s)$-centers for $p(x)$.

COMPUTATIONS. Apply Algorithm 5.1 with H chosen sufficiently large so that

$$(f_v+1)r_H s < (f_v-1)|X_H|. \tag{6.1}$$

Stop if an (a,f)-splitting disc for $p(x)$ is output by Algorithm 5.1. Otherwise, output the disc $D(X_H, r_H)$ and check if

$$2|X_H| \geq (f_v+1)\hat{r}. \tag{6.2}$$

If (6.2) holds, output 1 and stop; otherwise, output 0 and stop.

Correctness of Algorithm 6.1 will be proved by using the following geometric lemma.

LEMMA 6.1. *Let a complex X and positive r, $\hat{r}$, and $\hat{R}$ satisfy the inequality*

$$\left(\hat{R}+\hat{r}\right) r < |X| \left(\hat{R}-\hat{r}\right). \tag{6.3}$$

Then the disc $D(X,r)$ does not intersect the disc $D(0,\hat{r})$ if $2|X| \geq \hat{R}+\hat{r}$ and does not intersect the exterior of the disc $D(0,\hat{R})$ if $2|X| \leq \hat{R}+\hat{r}$.

PROOF. (See Figure 5.) The inequality (6.3) implies that the disc $D(X,r)$ has no overlap with $D(0,\hat{r})$ if $2|X| \geq \hat{R}+\hat{r}$. Indeed, due to (6.3), $|X|-r > |X|(1-(\hat{R}-\hat{r})/(\hat{R}+\hat{r}))$. Substitute $|X| \geq (\hat{R}+\hat{r})/2$ on the right-hand side and obtain that $|X|-r > \hat{r}$. Similarly, deduce that the disc $D(X,r)$ has no overlap with the exterior of $D(0,\hat{R})$ if $2|X| \leq \hat{R}+\hat{r}$. ∎

PROOF OF CORRECTNESS OF ALGORITHM 6.1. It suffices to consider the case where Algorithm 6.1 does not output a desired (a,f)-splitting disc for $p(x)$. Write $\hat{R} = f_v\hat{r}$, $X = X_H$, $r = sr_H$. Then (6.1) implies (6.3), and we may apply Lemma 6.1. Recall that the disc $D(X_H, r_H)$ is $(3a-2)$-full for $p(x)$. Therefore, by Fact 3.1, the disc $D(X_H, sr_H)$ is a full $(3a-2,s)$-cover for $p(x)$, that is, this disc contains all the existent $(3a-2,s)$-centers for $p(x)$, and we may discard all the zeros of $v(x)$ lying outside this disc. By the virtue of Lemma 6.1, this implies discarding all the zeros of $v(x)$ lying in $D(0,\hat{r})$, if (6.2) holds, and discarding all the other zeros (which lie in the exterior of $D(0,f_v\hat{r})$) otherwise. Now, correctness of Algorithm 6.1 follows. ∎

Let us next choose f, f_v, and H so as to satisfy (6.1). Due to (3.5) and part (c) of Proposition 5.1, the inequality (6.1) follows if

$$\frac{\left(1-\hat{\delta}\right)\hat{\delta}^H}{\left(1-2\hat{\delta}\right)} < \frac{(f_v-1)f^{3\lfloor an\rfloor+3}}{((f_v+1)s)}, \tag{6.4}$$

for $\hat{\delta}$ of (5.1) and for δ of (3.5). It is easy to verify that the latter inequality and, therefore, also (6.1) hold if, simultaneously,

$$\hat{\delta} < \frac{1}{4}, \tag{6.5}$$

$$\hat{\delta}^H < \frac{2(f_v-1)f^{3\lfloor an\rfloor+3}}{(3s(f_v+1))}. \tag{6.6}$$

LEMMA 6.2. *The bounds (6.5) and (6.6) hold if f, f_v, and H satisfy the relations*

$$f-1 = \frac{1}{(30an\eta)}, \qquad f_v \geq f, \tag{6.7}$$

$$\eta \geq 4, \qquad \eta \geq (90ans)^{1/(H-1)}, \qquad H > 1. \tag{6.8}$$

PROOF. Equation (6.7) implies that

$$f^{3an} = \left(1 + \frac{1}{30an\eta}\right)^{3an} = \exp\left(3an \ln\left(1 + \frac{1}{30an\eta}\right)\right).$$

Substitute the expansion $\ln(1+u) = u - u^2/2 + u^3/3 - \cdots$, for $u = 1/(30an\eta)$ and obtain that

$$\begin{aligned} f^{3an} &= \exp\left(3an\left(u - \frac{u^2}{2} + \frac{u^3}{3} - \cdots\right)\right) \\ &= \exp\left[\left(w - \frac{w^2}{2} + \cdots\right) + \sum_{i=1}^{\infty}\left(1 - \frac{1}{(3an)^i}\right)\frac{(-w)^{i+1}}{(i+1)}\right], \end{aligned}$$

where $w = 1/(10\eta) \le 1/40$. Now, substitute $\ln(1+w)$ for the power series $w - w^2/2 + \cdots$, observe that $\sum_{i=1}^{\infty}(1 - 1/(3an)^i)(-w)^{i+1}/(i+1) < (1 - 1/(3an))w^2/2 < w^2/2$, and deduce the bound

$$f^{3an} < (1+w)\exp\left(\frac{w^2}{2}\right) \le \left(1 + \frac{1}{(10\eta)}\right)\exp\left(\frac{1}{(200\eta^2)}\right).$$

Since $\eta \ge 4$ and since $3an \ge 3\lfloor an \rfloor$, it follows that

$$f^{3\lfloor an\rfloor+3} = \left(f^{3an}\right)^{(\lfloor an\rfloor+1)/an} \le \left(1 + \frac{1}{10\eta}\right)^{(1+1/an)} \exp\left(\left(1 + \frac{1}{an}\right)\frac{1}{200\eta^2}\right) < 1 + \frac{1}{9\eta},$$

$\delta = f^{3\lfloor an\rfloor+3} - 1 < 1/(9\eta)$, $\hat{\delta} = 8(1+\delta)\delta < 1/\eta$, which implies (6.5), since $\eta \ge 4$, due to (6.8).

Furthermore, from (6.7) we have $2f^{3\lfloor an\rfloor+3}/(f_v+1) > 1$ and $(f_v - 1)/3 \ge 1/(90an\eta)$. Multiply these two inequalities together and obtain that $2(f_v - 1)f^{3\lfloor an\rfloor+3}/(3(f_v+1)s) > 1/(90ans\eta)$. Deduce from (6.8) that $1/(90ans\eta) \ge 1/\eta^H$. Combine the two latter bounds with the bound $\hat{\delta}^H < 1/\eta^H$ and obtain (6.6). ∎

In particular, the relations (6.7),(6.8), and, therefore, also (6.5), (6.6), and (6.1) are satisfied for any $f_v \ge f$ and for any of the following three choices of H, η, and f:

$$\begin{aligned} &H = 2, && \eta = 90ans, && f - 1 = \frac{1}{2700(an)^2 s}, \\ &H = 3, && \eta = (90ans)^{1/2}, && f - 1 = \frac{1}{90(10s)^{1/2}(an)^{3/2}}, \\ &H = \lceil 0.5\log(90ans)\rceil + 1, && \eta = 4, && f - 1 = \frac{1}{120an}. \end{aligned} \tag{6.9}$$

Hereafter, we will stay with H, η, and f defined by (6.9). This will enable us to maximize f and thus to decrease the necessary precision of the computations and their Boolean complexity.

REMARK 6.1. Since we apply Algorithm 5.1, with H recursive calls for Algorithm 4.1, we shall bound $|X_H|/r_H$ according to part (c) of Proposition 5.1, instead of bounding $|X|/r$ according to (3.4). In order to simplify the expression for the bound on the relative radius $|X_H|/r_H$, we recall (5.1) and (3.5) and redefine δ^* of (4.9) as follows:

$$\delta^* = \frac{\hat{\delta}^H\left(1 - \hat{\delta}\right)}{\left(\left(1 - 2\hat{\delta}\right)f^{3\lfloor an\rfloor+3}\right)}. \tag{6.10}$$

We will use this expression for δ^* throughout, including applications of Lemma 4.1 and Algorithm 4.2; in particular, we will rewrite part (c) of Proposition 5.1 as follows:

$$r_H \le \delta^* |X_H|. \tag{6.11}$$

Combine (6.4) and (6.10), assume that $f_v < 3$, and obtain the following substitution for (4.4):

$$\delta^* < \frac{f_v - 1}{(f_v + 1)s} < \frac{f_v - 1}{2s} < \frac{1}{s}. \tag{6.12}$$

For H of (6.9), the bounds (6.6) and (6.12) hold under a mild restriction on $\hat{\delta}$, compatible with the bounds $1/\hat{\delta} = O(1)$ and (4.12). The choice of $H = 1$ would, on the contrary, have brought us back to the bounds (4.4) and (4.11). Moreover, Algorithm 6.1 is recursively applied in the next section as a block of Algorithm 7.1; if we had set $H = 1$ or even $H = O(1)$ in these applications, then we would have arrived at an (a, f)-splitting disc or at an (a, B, f)-splitting disc where $f - 1$ can be very small, say, of the order $1/n^{-cn}$ for a positive constant c; consequently, the cost of splitting $p(x)$ over such a disc can be very large (see Remark 9.2).

7. IMPROVED COMPUTATION OF A SPLITTING DISC: AN ALGORITHM AND COMPLEXITY ESTIMATES

Next, we will summarize the algorithms of the previous sections in order to improve the computation of a splitting disc.

ALGORITHM 7.1. DISC$(p(x), B)$.

INPUT: Polynomial $p(x)$ of (2.1), natural H and n_0, real a, B, f and s (provided that n_0 is a fixed constant, a satisfies (4.2), f and H satisfy (6.9), and s is defined according to parts (a) or (b) of Theorem 4.1), and two black-box subroutines, specified below and denoted DISC$(v(x), B_v)$ and FACTOR$(v(x), D)$ (for a real B_v, for a polynomial $v(x)$ of degree less than n, and for its (a, f)-splitting disc D).

OUTPUT:

(a) Either an (a, f)-splitting disc SD for $p(x)$ or
(b) an (a, B, f)-splitting disc SD_B for $p(x)$.

TWO SUBROUTINES. The subroutine DISC$(v(x), B_v)$ solves, for a polynomial $v(x)$ and scalars a, n_0, B_v, f_v, H_v, and s_v, the same problem as Algorithm 7.1 solves for $p(x)$, a, n_0, B, f, H, and s. The input values f_v, H_v, and s_v are defined (like the values f, H, and s used before) so as to satisfy the assumptions of Theorem 4.1 and the equations (6.9), except that, now, in all cases, we replace f by f_v, H by H_v, s by s_v, and n by $d_v = \deg v(x)$. (Algorithm 7.1 would correctly work also for $f_v = f$, $H_v = H$, and $s_v = s$ invariant in $\deg v(x)$, but then the computational cost would increase, slightly.) For B_v one may choose any value that satisfies the following bound, extending (4.10) and the equation of (4.5):

$$B_v \geq B + \log \left\lceil \frac{\delta_v^* f_v^{2\lceil (3a-3)d_v \rceil - 2}}{(1 - \delta_v^* s_v)} \right\rceil, \tag{7.1}$$

where δ_v^* is defined by the equations (6.10), (5.1), and (3.5) in which f is replaced by f_v, δ by δ_v, and s by s_v. (In particular, one may define B_v by setting equality in (7.1).) The subroutine FACTOR$(v(x), D)$ numerically splits $v(x)$ over the disc D, that is, computes two polynomials, $F^*(x)$ (monic and approximating the highest degree monic factor $F(x)$ of $v(x)$ that has all its zeros lying in D) and $G^*(x)$ (approximating the factor $G(x) = v(x)/F(x)$ of $v(x)$, which has no zeros lying in D), that satisfy the next bound (compare (1.1) with n replaced by $n - l + 1$, b by β, and $\bar{b}$ by $\bar{\beta}$)

$$\|v(x) - F^*(x)G^*(x)\| \leq 2^{-\bar{\beta}}, \qquad \bar{\beta} = (\beta + 3)(n - l + 1) + \log(n - l + 1) + 2, \tag{7.2}$$

where the norm is defined by (1.7) and where β satisfies the bound

$$2^{-\beta} = \Delta \le 2^{-B^*}. \tag{7.3}$$

COMPUTATIONS BY ALGORITHM 7.1.

STAGE 0. INITIALIZATION. Set $v(x) = p^{(l-1)}(x)$ for $l = \lfloor (3a-2)n \rfloor + 1$, of (4.1).

STAGE 1. If $\deg v(x) \le n_0$, first approximate the zeros of $v(x)$, then invoke one of Algorithms 4.3 or 4.4, in order to compute and to output an (a, f)-splitting disc or an (a, B, f)-splitting disc for $p(x)$; then stop. Otherwise, fix B_v according to (7.1) and apply the subroutine DISC$(v(x), B_v)$, which outputs an (a, f_v)-splitting disc or an (a, B_v, f_v)-splitting disc for $v(x)$; in both cases, such an output splitting disc is denoted $D(C_v, R_v)$. Shift the origin into C_v and go to Stage 2.

STAGE 2. Write $\hat{r} = R_v$ and $D = D(C_v, R_v)$ and apply Algorithm 6.1 for H replaced by $H+1$. If this algorithm outputs an (a, f)-splitting disc or an (a, B, f)-splitting disc for $p(x)$, then stop. Otherwise, Algorithm 6.1 outputs a disc $D(X_{H+1}, r_{H+1})$ and an integer, 0 or 1; in this case go to Stage 3.

STAGE 3. If Algorithm 6.1 outputs 0 and if D is an (a, B_v, f_v)-splitting disc for $v(x)$, then shift the origin into X_{H+1}, apply the algorithm supporting Fact 2.2, output an (a, B, f)-splitting disc for $p(x)$, denote this disc SD_B, and stop. Otherwise (that is, unless simultaneously Algorithm 6.1 outputs 0 and the disc D turns out to be an (a, B_v, f_v)-splitting disc for $v(x)$), apply the subroutine FACTOR$(v(x), D)$ and set either $v(x) = F^*(x)$, if Algorithm 6.1 outputs 0, or $v(x) = G^*(x)$, if Algorithm 6.1 outputs 1. Then go to Stage 1.

PROOF OF CORRECTNESS OF ALGORITHM 7.1. Let $J(x)$ denote the factor of $v(x)$ approximated by the output polynomial of the subroutine FACTOR$(v(x), D)$ applied at Stage 3 of Algorithm 7.1 (so that $J(x) = F(x)$ or $J(x) = G(x)$). At Stage 0 of Algorithm 7.1, some zero of $v(x)$ is a $(3a-2, s)$-center for $p(x)$, due to Corollary 4.1. Due to correctness of Algorithm 6.1, the latter property of $v(x)$ is extended to $J(x)$ and, therefore, is maintained throughout the computation by Algorithm 7.1 if we ignore the errors of the factorization of $p(x)$ approximately computed by Algorithm 7.1. Due to (4.2), $\deg J(x)$ is bounded by a fixed fraction of $\deg v(x)$; therefore, Algorithm 7.1 must terminate in $O(\log(\deg v(x)/n_0))$ passes through Stage 3 and, at the termination, must output either an (a, f)-splitting disc for $p(x)$ at Stages 1 or 2 or an (a, B, f)-splitting disc for $p(x)$ at Stages 1, 2, or 3. It remains

(a) to examine the influence of the approximation errors on correctness of application of Algorithm 6.1 as a block of Algorithm 7.1, and

(b) to show correctness of Algorithm 7.1 in the case where Algorithm 6.1 outputs 0 and where the disc $D = D(C_v, R_v)$ is an (a, B_v, f_v)-splitting disc for $v(x)$.

Towards the first goal, we recall that, on the one hand, $r_{H+1} \ge 2^{-B^*}$ unless some application of Algorithm 4.1 gives us an (a, B, f)-splitting disc for $p(x)$ and that, on the other hand, due to (7.2) and Corollary 2.2, the zeros of all the computed approximations to the factors of $p^{(l-1)}(x)$ may deviate from the respective zeros of $p^{(l-1)}(x)$ by at most $\Delta = 2^{-\beta}$. Therefore, in order to preserve correctness of Algorithm 6.1, performed as a block of Algorithm 7.1, we only need to extend the bound (6.1) as follows:

$$(f_v + 1)(r_{H+1} + 2\Delta)s < (f_v - 1)|X_{H+1}|.$$

Proposition 5.1 (for $\hat{\delta} < 1/4$ of (6.5) and for $H \ge 2$) implies that $r_H/|X_H| < 6/4^{H+1} \le 3/32$, so that $|X_{H+1}| \ge |X_H| - r_H \ge 29|X_H|/32$, and therefore, it is sufficient for us to ensure that $(f_v+1)(r_{H+1}+2\Delta)s < 29(f_v-1)|X_H|/32$. The desired extension of (6.1) to the latter inequality immediately follows from (6.1) and from the following bounds: $4r_{H+1} \le r_H$ (see (5.7),(6.5)),

$2^{-B^*} \le r_{H+1}$, and $2\Delta \le 2^{-B^*} \le 2r_{H+1}$ (see (7.2), (7.3)). Thus, the errors of approximation of the factors of $p^{(l-1)}(x)$ do not influence correctness of Algorithm 6.1.

Now, we shift to our second goal. Since D is assumed to denote an (a, B_v, f_v)-splitting disc for $v(x)$, we obtain from Definition 4.1 that $R_v \le \rho_v^*$, where $\rho_v^* = 2^{-B_v}$. Furthermore, due to correctness of Algorithm 6.1, both discs, $D = D(C_v, R_v)$ and, therefore, also $D(C_v, \rho^*)$, are $(3a-2, s)$-covers for $p(x)$; that is, both of them contain a $(3a-2, s)$-center for $p(x)$. Now, we apply Lemma 4.1, for $r = r_{H+1}$, $\rho^* = \rho_v^*$, $\delta^* = \delta_v^*$, $s = s_v$, and the origin shifted into C_v, and deduce that $r_{H+1} \le \delta_v^* \rho_v^* / (1 - \delta_v^* s_v)$. Then (7.1) implies that $r_{H+1} \le 2^{-B^*}$. Now, Fact 4.1 implies correctness of obtaining an (a, B, f)-splitting disc for $p(x)$ at Stage 3 of Algorithm 7.1. ∎

Next, we will estimate (sequential and parallel) Boolean and arithmetic cost $PBC_D(B,n)$, $PAC_D(B,n)$, $PRAC_D(B,n)$ and $SAC_D(B,n)$ of performing Algorithm 7.1, DISC$(p(x), B)$, for $n > 2$, a of (4.2) and $l = \lfloor (3a-2)n \rfloor + 1$ of (4.1); in particular, we may define a and l by (4.3). (Here, the subscript D abbreviates "DISC.")

As we have already observed, there can be at most $O(\log(d_v/n_0)) = O(\log n)$ transitions to a new $v(x)$ at Stage 1 of Algorithm 7.1, for $d_v = \deg v(x)$, and there can be as many passes through Stages 1–3 of Algorithm 7.1. It remains to estimate the computational cost of each pass. This cost is dominated by the complexity of application of the subroutine FACTOR$(v(x), D)$, for splitting the polynomial $v(x)$.

Let us assume that equalities are set in (7.1) and (7.3). Then, application of Theorem 2.1 for n replaced by $d_v = \deg v(x)$, for

$$\bar{B} = B_v^* + \bar{\beta} + d_v, \tag{7.4}$$

for B_v^* satisfying

$$2^{-B_v^*} = \frac{2^{-B_v}}{f_v^{2\lceil (3-3a)d_v \rceil - 2}} \tag{7.5}$$

(compare (4.5) and (7.1)), and for $\bar{\beta}$ defined by (7.2) and (7.3), gives us the following bounds on the computational complexity of each pass through Stages 1–3 of Algorithm 7.1 (where $d_v = \deg v(x)$ decreases in every pass):

$$PBC_S(B, d_v) = O_B\left((\log d_v)\left(\log \bar{B}\right)^2, \frac{\left(M\left(d_v^3 + \bar{B} d_v \log \bar{B}\right)\right)}{\left(\log \bar{B}\right)^2}\right),$$

$$PAC_S(B, d_v) = O_A\left((\log d_v)\log \bar{B}, \frac{d_v^2}{\log \bar{B}}\right),$$

$$PRAC_S(B, d_v) = O_A\left((\log d_v) t_{3,1}\left(\bar{B}, d_v\right), d_v\right), \qquad \text{allowing Las Vegas randomization,}$$

$$SAC_S(B, d_v) = O_A\left((\log d_v) t_{2,1}\left(\bar{B}, d_v\right) d_v, 1\right),$$

for $M(d)$ of (1.3) and $t_{i,j}(\bar{B}, d_v) = (\log d_v)^i + (\log \bar{B})^j$ of (1.4).

We summarize the above bounds, where $\bar{B}$ is defined by (7.4) and where initially d_v takes on the value $n+1-l, l = \lceil (3a-2)n \rceil$, and then, in each of the $O(\log n)$ recursive steps, decreases by at least a fixed constant factor exceeding 1. This gives us the following estimates for the overall arithmetic and Boolean cost of performing Algorithm 7.1:

$$PBC_D(B, n) = O_B\left((\log n)^2 \left(\log \bar{B}\right)^2, \frac{\left(M\left(n^3 + \bar{B} n \log \bar{B}\right)\right)}{\left(\left(\log \bar{B}\right)^2 \log n\right)}\right), \tag{7.6}$$

$$PAC_D(B, n) = O_A\left((\log n)^2 \log \bar{B}, \frac{n^2}{\left(\left(\log \bar{B}\right) \log n\right)}\right), \tag{7.7}$$

$$PRAC_D(B, n) = O_A\left((\log n)^2 t_{3,1}\left(\bar{B}, n\right), \frac{n}{\log n}\right), \qquad \text{using randomization,} \tag{7.8}$$

$$SAC_D(B, n) = O_A\left((\log n) t_{2,1}\left(\bar{B}, n\right) n, 1\right), \tag{7.9}$$

where $M(d)$ and $t_{i,j}(\bar{B}, n)$ are defined by (1.3) and (1.4).

8. SUMMARY OF THE ENTIRE RECURSIVE PROCESS AND THE OVERALL COMPLEXITY ESTIMATES

As soon as we compute an (a, f)-splitting disc for $p(x)$, we apply Theorem 2.1 and split $p(x)$ over this disc, into two factors having degrees k and $n-k$, respectively, for $(1-a)n/2 \le k \le (1+a)n/2$; in particular, $n \le 12k \le 11n$ if we choose $a = 5/6$, according to (4.3). Then Algorithm 7.1 and Theorem 2.1 are recursively applied to the factors. $O(\log n)$ such recursive steps reduce the original problem of approximating the zeros of $p(x)$ to $O(n)$ such problems for polynomials of degrees at most $n_0 = O(1)$, which we then solve at the overall arithmetic cost $O_A(\log(bn), n)$, by applying the algorithms of [2] or [29], say. If at some recursive step we compute an (a, B, f)-splitting disc $D(X, \rho)$ for $p(x)$ (rather than an (a, f)-splitting disc), then the recursive process is only simplified. Indeed, to handle this case we write $B = b$, which satisfies (4.7), and then let X approximate all the $k > \lfloor (3a-2)n \rfloor$ zeros of $p(x)$ lying in $D(X, \rho)$. It remains to deal with a single factor of $p(x)$, of degree at most $n - \lfloor (3a-2)n \rfloor - 1 \le \lceil (3-3a)n \rceil - 1$ (that is, at most $\lceil n/2 \rceil - 1$ for a of (4.3)), rather than with two factors.

The complexity bounds of Theorem 2.1 are given in terms of $\bar{B} = B^* + \hat{b} + n$, where B^* is defined by (4.5) and (4.7). Due to (4.5), it suffices if $B^* \ge B + (2\lceil (3-3a)n \rceil - 2) \log f$. Under (4.3) and (6.9), we satisfy this bound already for $B^* = B + 0.15$, which implies that

$$\bar{B} = B + \hat{b} + n + 0.15.$$

For $B = b$ and, more generally, for $B = O(b)$, we have $\bar{B} = O(\hat{b} + b + n)$. For splitting the polynomial $p^{(l-1)}(x)$ or its factors (denoted $v(x)$) over an (a, B_v, f_v)-splitting disc, we apply Theorem 2.1 for $\bar{B} = B_v^* + \bar{\beta} + d_v$ of (7.4), where we define B_v^* and $\bar{\beta}$ by (7.1)–(7.3) and (7.5), replacing the two inequalities by equalities in (7.1) and (7.3). We have $d_v < n$, $\bar{\beta} = O(\beta n)$ (compare (7.2), (7.3)), and $B_v^* = O(B)$ (compare (6.12) and (7.1)). Consequently, assuming, as before, that $B = O(b)$, we obtain that $\beta = O(b)$ and $\bar{B}_v = O(bn)$.

By taking into account the latter bounds on $\bar{B}$ and $\bar{B}_v$ and by recursively applying the bounds (7.6)–(7.9), Theorem 2.1, and the variant of Brent's principle, cited in the Introduction, we arrive at the estimates of Theorem 1.1, for approximating the n zeros of $p(x)$. ∎

REMARK 8.1. By applying some special techniques of binary segmentation (due to [61] and rediscovered and extended at first in [62] and then in [1], in [63, Section 40], and in [64]), one may further decrease the Boolean sequential time bound and the Boolean processor bound (by roughly a logarithmic factor) [1,7].

9. AN ALGORITHM AND COMPLEXITY ESTIMATES FOR SPLITTING A NORMALIZED POLYNOMIAL OVER THE UNIT ISOLATED DISC

In this section, we will briefly recall a known splitting algorithm developed, in particular, by Delves and Lyness [65], by Schröder [17, pp. 295–320; 66], and, so far probably most extensively, by Schönhage [1] and Kirrinnis [7] (compare also [28, Appendices A and B]). The algorithm splits a normalized polynomial over the unit disc $D(0, 1)$. In Section 11, we will extend this algorithm to splitting polynomial $p(x)$ of (2.1),(2.2) over any fixed disc $D(X, R)$, where X and R satisfy (2.5). Together with our technique of recursive descending, to be introduced in Section 10, this extension supports Theorem 2.1. The variant of Brent's principle (cited in the Introduction) will be routinely applied in the following sections in order to improve processor bounds (by a logarithmic factor) (compare, for instance, [39, Proposition 4.1.1]).

We will keep defining the norm by (1.7).

ALGORITHM 9.1. Splitting a normalized polynomial over the unit disc.

INPUT: The values a and f satisfying (5.1) and (6.9), two integers k and n, $0 < k < n$; positive $\bar{c}$, $\hat{c}$, c^*, and $\bar{B}$ (see Remark 9.1 on the choice of $\bar{c}$, $\hat{c}$, and c^*), and a polynomial $\bar{p}(y)$ satisfying the following relations:

$$\bar{p}(y) = \sum_{i=0}^{n} \bar{p}_i y^i = \bar{p}_n \prod_{j=1}^{n} (y - \bar{z}_j), \qquad \|\bar{p}(y)\| = 1, \quad \bar{p}_n \neq 0, \tag{9.1}$$

$$|\bar{z}_j| \leq 1, \qquad j = 1, \ldots, k, \tag{9.2}$$

$$f \leq |\bar{z}_j|, \qquad j = k+1, \ldots, n. \tag{9.3}$$

OUTPUT: Approximations $\bar{F}_k^*(y)$ and $\bar{G}_{n-k}^*(y)$ to the two factors, $\bar{F}_k(y) = \prod_{i=1}^{k}(y - \bar{z}_i)$ and $\bar{G}_{n-k}(y) = \bar{p}(y)/\bar{F}_k(y)$ (compare Definition 2.2, where $\bar{p}(y)$, $\bar{F}_k(y)$ and $\bar{G}_{n-k}(y)$ replace $p(x)$, $F_k(x)$ and $G_{n-k}(x)$, respectively), satisfying the following bound:

$$\left\|\bar{\Delta}(y)\right\| \leq 2^{-\bar{B}}, \qquad \bar{\Delta}(y) = \bar{F}_k^*(y)\bar{G}_{n-k}^*(y) - \bar{p}(y). \tag{9.4}$$

COMPUTATIONS.

1. Compute approximations s_i^* to the power sums

$$\bar{s}_i = \sum_{j=1}^{k} \bar{z}_j^i, \qquad i = 1, 2, \ldots, K, \quad K = 2^{\lceil \log k \rceil} < 2k,$$

of all the zeros of $\bar{p}(y)$ lying in the disc $D(0,1)$, so as to satisfy the bounds

$$|s_i^* - \bar{s}_i| \leq \frac{1}{2^{\bar{c}n}}, \qquad i = 1, 2, \ldots, K. \tag{9.5}$$

2. Use the values s_i^* computed at Stage 1 in order to approximate the factor $\bar{F}_k(y) = \prod_{j=1}^{k}(y - \bar{z}_j)$ of $\bar{p}(y)$ within the error norm bound $1/2^{c^*n}$, that is, to compute a monic polynomial $\hat{F}_k(y)$ satisfying

$$\left\|\bar{F}_k(y) - \hat{F}_k(y)\right\| \leq 2^{-c^*n}. \tag{9.6}$$

3. Approximate the factor $\bar{G}_{n-k}(y) = \bar{p}(y)/\bar{F}_k(y)$ by a polynomial $\hat{G}_{n-k}(y)$ so as to satisfy the inequality

$$\left\|\bar{p}(y) - \hat{F}_k(y)\hat{G}_{n-k}(y)\right\| \leq 2^{-\hat{c}n}. \tag{9.7}$$

4. Improve the approximations $\hat{F}_k(y)$ and $\hat{G}_{n-k}(y)$, computed at Stages 2 and 3, so as to compute and to output $\bar{F}_k^*(y)$ and $\bar{G}_{n-k}^*(y)$ satisfying (9.4).

Stage 1 is performed by means of numerical integration (see [1, Section 12], or [28, Appendix A]); that is, the values

$$s_i^{(k)} = \frac{1}{2\pi\sqrt{-1}} \int_{|y|=1} \left(\frac{y^i \bar{p}'(y)}{\bar{p}(y)}\right) dy$$

are approximated by the sums

$$s_i^* = \frac{1}{Q} \sum_{q=0}^{Q-1} \omega^{(i+1)q} \frac{\bar{p}'(\omega^q)}{\bar{p}(\omega^q)},$$

where $i = 1, \ldots, K$; $\omega = \exp(2\pi\sqrt{-1}/Q)$ is a primitive Q^{th} root of 1, and a natural $Q = Q(f)$ is specified later on, as a function in f.

Stage 2 is the transition from the computed approximations s_i^* of the power sums $s_i^{(k)}$ to the initial approximations of the coefficients of $\bar{F}_k(y)$. This stage is performed by means of a variant of Newton-Hensel's lifting algorithm from [1, Section 13] (compare [39, pp. 34–35]).

Stage 3 is reduced to the division of $\bar{p}(y)$ by the computed approximation $\hat{F}_k(x)$ to $\bar{F}_k(x)$ (compare [64,67]).

Stage 4 relies on a sophisticated iteration algorithm of [1, Sections 10, 11, and 13] (also compare [7; 17, pp. 295–320; 28, Appendices A and B; 66]).

REMARK 9.1. The choice of sufficiently large constants c^*,$\bar{c}$, and $\hat{c}$ (all of them independent of $\bar{B}$ and n) is specified in [1]. The constant $\hat{c}$ is chosen so as to ensure that the iteration algorithm applied at Stage 4 converges sufficiently fast (so that $\epsilon_{i+1} \le \epsilon_i^{1.5}$, where ϵ_h denotes the error norm bound in h iteration steps) provided that the initial approximations $\hat{F}_k(y)$ and $\hat{G}_{n-k}(y)$ to $\bar{F}_k(y)$ and $\bar{G}_{n-k}(y)$ satisfy (9.6) and (9.7). The constant c^* is chosen so as to ensure the bound (9.7) provided that (9.6) holds. The constant $\bar{c}$ is chosen so as to ensure (9.6) as long as the values s_i^*, computed at Stage 1, satisfy (9.5).

According to the estimates of [28, Appendices A and B] and of [39, pp. 34–35], the arithmetic cost of performing the four stages of Algorithm 9.1 is bounded as follows:

at Stage 1, by $O_A(\log Q, Q)$,

at Stage 2, by $O_A((\log n)^2, n/\log n)$,

at Stage 3, by $O_A(\log n, n)$,

at Stage 4, by $O_A((\log n)\log(\bar{B}n), n)$.

Moreover, the analysis presented in [1, Sections 9–13,16; 2,4,7] shows that the precision of $O(n)$ bits suffices at Stages 1, 2, and 3, whereas the precision of $O(\bar{B}+n)$ bits suffices at Stage 4 (compare Remark 9.1). Based on this analysis and on the known bounds (1.5) and $O_B(\log d, d/\log d)$ on the complexity of a multiplication and an addition/subtraction of two integers modulo 2^d-1, respectively, we extend the above arithmetic complexity bounds to the Boolean complexity estimates. (The known asymptotic bounds on the Boolean cost of an integer division are either the same (in the sequential case) or only slightly higher (in the parallel case) than ones for a multiplication [39,68], whereas the divisions required in Algorithm 9.1 are much less numerous than multiplications, so the overall cost of performing all the multiplications involved in Algorithm 9.1 dominates the overall Boolean cost of performing the algorithm.)

The overall arithmetic and Boolean cost of performing Algorithm 9.1 depends on the choice of $Q = Q(f)$ at Stage 1. According to the estimates of [1,28], we need to choose Q of the order $n/(f-1)$. Since f is defined by (6.9), the latter bound on Q implies the choice of $Q = O(n^2)$, and then, summarizing the above analysis gives us a splitting of $\bar{p}(y)$ over the unit disc $D(0,1)$ satisfying (9.4). The computational cost of this splitting is bounded according to parts (a) and (b) of Theorem 2.1. In Section 11, we will show how to extend the cost bounds for splitting $\bar{p}(y)$ over $D(0,1)$ to ones for splitting over a subdisc of $D(0,1)$. Now, we observe that the arithmetic cost bounds of part (b) of Theorem 2.1 are far from the optimum because the order of $n^2 \log n$ arithmetic operations are involved in Stage 1 of Algorithm 9.1 (if Q is of the order n^2). This does not contradict our final arrival at nearly optimum Boolean cost bounds of part (a) of Theorem 1.1 because the latter arithmetic operations are performed with a lower precision, of $O(n)$ bits (versus $O(\bar{B}+n)$-bit precision, generally required at Stage 4). Consequently, the Boolean cost bounds (which more realistically measure the complexity of approximating polynomial zeros than the arithmetic cost bounds do) are lower at Stage 1 than at Stage 4, at least in the case of our major interest, where $\bar{B}$ is of the order bn and $n = O(b)$ (compare (1.1) and Fact 2.6).

For theoretical purposes, however, we also wish to have an optimal or nearly optimal algorithm in terms of arithmetic complexity, and in the next section we will decrease the overall arithmetic cost of splitting given by part (b) of Theorem 2.1. We will achieve this goal by means of devising an algorithm that lifts an isolation ratio of the input (splitting) disc $D(0,1)$ of Algorithm 9.1.

Specifically, we will lift the ratio from f of (6.9) to $f \geq 4$ (say), which will decrease Q to the level $O(n)$. Furthermore, in Sections 12 and 13, we will show how to bound the precision of computing by the resulting algorithm so as to *make this algorithm supports the same (record and nearly optimal) Boolean complexity bounds* as ones obtained in parts (a) of Theorems 2.1 and 1.1 (provided that $n = O(b)$).

REMARK 9.2. Q has the order $n/(f-1) = cn^2 s$ under (4.11). (In particular, this is the order of Q achieved by the algorithm of [3].) In comparison to the case of Q of the order $O(n^2)$ under (4.12), this implies an increase, by the factor s, of both arithmetic and Boolean cost bounds of Theorems 2.1 and 1.1 on sequential time and number of processors, and we ought to choose s of an order of at least $n^{1/3}$, to satisfy the assumptions of Theorem 4.1. Furthermore, the arithmetic cost bounds have an order of at least $Q \log Q$, which means, in particular, that they stay above the level $n^2 s$ for the algorithm of [3].

10. DECREASING THE ARITHMETIC COMPLEXITY OF THE SPLITTING ALGORITHM

In this section, we will combine the known techniques (for Graeffe's recursive lifting and for splitting a polynomial into two factors) with our new techniques (for recursive descending) in order to increase the isolation ratio of a splitting disc for $\bar{p}(y)$, from the value f of (6.9) to at least 4. This will enable us to decrease the upper bound on the parameter Q (used at Stage 1 of Algorithm 9.1) to the level $O(n)$ and, thus, to decrease the upper bound on the overall arithmetic computational cost of splitting.

ALGORITHM 10.1. Recursive lifting, splitting, and recursive descending.

INPUT: As in Algorithm 9.1.

OUTPUT: A monic polynomial $F_k^*(y)$, of degree k, satisfying (9.6) for $\hat{F}_k(y) = F_k^*(y)$.

COMPUTATIONS.

1. (*recursive lifting*). Set $q_0(y) = \bar{p}(y)/\bar{p}_n$, for polynomial $\bar{p}(y)$ of (9.1), set

$$u = 1 + \left\lceil \log\left(\frac{1}{\log f}\right)\right\rceil = O(\log n), \tag{10.1}$$

and apply u iteration steps,

$$q_{j+1}(y) = (-1)^n q_j(-\sqrt{y})q_j(\sqrt{y}), \qquad j = 0, 1, \ldots, u-1. \tag{10.2}$$

(The iteration (10.2) has been successively discovered and rediscovered at first by Dandelin, then by Lobachevsky, and then by Graeffe [18].) Note that $q_j(y) = \prod_{i=1}^n (y - \bar{z}_i^{2^j})$, $j = 0, 1, \ldots, u$, so that $D(0,1)$ is an f^{2^j}-isolated disc, for the polynomial $q_j(y)$.

2. (*splitting* $q_u(y)$). Deduce from (10.1) that $f^{2^u} > 4$ and apply Algorithm 9.1 for $Q = O(n)$ in order to split the polynomial $\bar{p}_u(y) = q_u(y)/\|q_u(y)\|$ numerically, over the disc $D(0, r^{2^u})$, into two factors, $F_{k,u}^*(y)$ and $\bar{G}_{n-k,u}^*(y)$ (compare [1,28]); obtain numerical factorization of $q_u(y)$ as the product $F_{k,u}^*(y)G_{n-k,u}^*(y)$, where we write $G_{n-k,u}^*(y) = \|q_u(y)\|\bar{G}_{n-k,u}^*(y)$.

3. (*recursive descending*). Recursively recover approximations to the factors $F_{k,u-j}(y)$ (monic) and $G_{n-k,u-j}(y)$ in the splittings $q_{u-j}(y) = F_{k,u-j}(y)G_{n-k,u-j}(y)$ of the polynomials $q_{u-j}(y)$ of (10.2) over the disc $D(0,1)$, for $j = 1, 2, \ldots, u$. Output the computed approximation to the factors $F_k(y) = F_{k,0}(y)$ and $G_{n-k}(y) = G_{n-k,0}(y)$ of the polynomials $q_0(y)$ and $\bar{p}(y) = \bar{p}_n q_0(y)$. Before the j^{th} step of this recursive recovery (which we also call recursive descending), we have the polynomial $q_{u-j}(y)$, computed at Stage 1, and an approximation

to the factor $G_{n-k,u-j+1}(y)$ of $q_{u-j+1}(y)$ computed at the $(j-1)^{\text{st}}$ step (or at Stage 2 if $j=1$). At the j^{th} step, first compute approximations $\hat{F}_{k,u-j}(y)$ and $\hat{G}_{n-k,u-j}(-y)$ to the pair of the polynomials $F_{k,u-j}(y)$ and $G_{n-k,u-j}(-y)$, by observing that the latter pair of polynomials fills up the $(k,n-k)$-entry of the Padé approximation table for the analytic function

$$\frac{q_{u-j}(y)}{G_{n-k,u-j+1}(y^2)} = \frac{F_{k,u-j}(y)}{G_{n-k,u-j}(-y)}. \tag{10.3}$$

(We refer the reader to [39,69] on the definition and some basic properties of Padé tables. To substantiate (10.3), observe the following equations:

$$G_{n-k,u-j+1}(y^2) = (-1)^{n-k} G_{n-k,u-j}(y) G_{n-k,u-j}(-y),$$
$$q_{u-j}(y) = F_{k,u-j}(y) G_{n-k,u-j}(y),$$

and $\gcd(F_{k,u-j}(y), G_{n-k,u-j}(-y)) = 1$.) Ensure that

$$\left\| q_{u-j}(y) - \hat{F}_{k,u-j}(y)\hat{G}_{n-k,u-j}(y) \right\| \le \frac{1}{2^{\hat{c}n}\,\|q_{u-j}(y)\|} \tag{10.4}$$

(compare (9.7), where $\|\bar{p}(y)\| = 1$). Improve the computed approximations to $F_{k,u-j}(y)$ and $G_{n-k,u-j}(y)$ by performing Stage 4 of Algorithm 9.1, where $\bar{p}(y)$ is replaced by $q_{u-j}(y)$, $\hat{F}_k(y)$ by $\hat{F}_{k,u-j}(y)$, and $\hat{G}_{n-k}(y)$ by $\hat{G}_{n-k,u-j}(y)$. Then go to the $(j+1)^{\text{st}}$ step of the recursive recovery if $j < u$ or stop if $j = u$.

REMARK 10.1. The reader may examine two alternative versions of Stage 3 (see [8]), where one either approximates the factors $F_{k,u-j}(y) = \gcd(q_{u-j}(y), F_{k,u-j+1}(y^2))$ and $G_{n-k,u-j}(y) = \gcd(q_{u-j}(y), G_{n-k,u-j+1}(y^2))$ or approximates only $F_{k,u-j}(y)$ as the gcd and then applies Stages 3 and 4 of Algorithm 9.1 to compute some refined approximations to both factors. (Here, $\gcd(u(x), v(x))$ denotes the monic greatest common divisor of two polynomials $u(x)$ and $v(x)$.) The known algorithms for computing the gcds [39] lead to the same arithmetic complexity estimates as for computing the Padé approximations.

Next, let us estimate the arithmetic complexity of Algorithm 10.1 and of our solution of the entire splitting problem.

By using the FFT based algorithms for polynomial computations [39, Chapter 1], we perform Stage 1 of Algorithm 10.1 at the cost bounded by $O_A((\log n)^2, n)$. According to [28, Appendices A and B], the cost of performing Stage 2 is $O_A((\log n)^2, n)$ too.

At each of the u steps of Stage 3 of Algorithm 10.1, we may compute the Padé approximation of the analytic function of (10.3) at the cost $O_A((\log n)^2 n, 1)$ [39], which gives us the bound $O_A((\log n)^3 n, 1)$ for all the u steps (compare (10.1)).

To compute the Padé approximation of (10.3) in parallel, we reduce the problem to the solution of a nonsingular Toeplitz linear system of $n-k$ equations (see [39, equation (2.5.6)]), associated with the entry $(k, n-k)$ of the Padé approximation table for the analytic function $q_i(y)/G_{n-k,i+1}(y^2) = F_{k,i}(y)/G_{n-k,i}(-y)$; this entry is to be filled up with the nondegenerating pair of polynomials $(F_{k,i}(y), G_{n-k,i}(-y))$, $i = u-j$. (Nonsingularity and nondegeneration follow since the degrees of the polynomials $F_{k,i}(x)$ and $G_{n-k,i}(x)$ are known to be exactly k and $n-k$, respectively.) At this point we apply the following theorem.

THEOREM 10.1. *The exact solution of a nonsingular Toeplitz linear system of m equations with an integer coefficient matrix T or with a matrix T filled with Gaussian integers (of the form $a + b\sqrt{-1}$, for integers a and b) can be computed at a cost $O_A((\log m)^2, m^2/\log m)$, by deterministic algorithms, and at a cost $O_A((\log m)^2 \log L, m)$, by a randomized Las Vegas algorithm (using only a single random parameter), provided that $L = m \log \|T\|$ and that $\|T\|^m$ is an upper bound on $|\det T|$. Furthermore, both of these algorithms can be performed with the precision of $O(L)$ bits if the right-hand-side vector of the linear system is filled up by integers or Gaussian integers whose absolute values are less than 2^L.*

REMARK 10.2. The estimates of Theorem 10.1 have been deduced in the case of a more general class of nonsingular Toeplitz-like linear systems (see [39,70]).

The deterministic bound of Theorem 10.1 has been obtained in [70], by means of the techniques of parametrized Newton's iteration (introduced in [70] and also applicable to the solution of nonsingular Toeplitz-like linear systems of m equations over any field of constants having a characteristic 0 or greater than m). The randomized bound of Theorem 10.1 has been obtained in [39, p. 356], by means of straightforward combination of the results and techniques of [71–74]. (Specifically, the iterative algorithms of [71,72] have been originally proposed for parallel inversion of a general nonsingular matrix and relied on the combination of the variable diagonal techniques of [71,72] with the customary techniques of p-adic lifting. In the case of Toeplitz or (more generally) Toeplitz-like input matrices T, these algorithms have been made more effective in [39] (so as to support Theorem 10.1). This has been achieved by means of incorporation of the techniques of [73,74], which, in particular, include a nontrivial algorithm for cutting the length of displacement generators of the computed approximations to T^{-1}.)

Since the output error bounds of (10.4) suffice for our purpose at the recursive descending stage of Algorithm 10.4, it follows that we only need to apply Theorem 10.1 in the case where $\log L = O(\log m)$, $m = n$, and this application will give us the bounds $O_A((\log n)^3, n^2/\log n)$ (deterministic) and $O_A((\log n)^4, n)$ (Las Vegas randomized) on the parallel cost of the solution of the $u = O(\log n)$ Toeplitz or Sylvester linear systems at Stage 3 of Algorithm 10.1. (In Sections 12 and 13, we will show that, furthermore, we only need to deal with the case where $L = n^{O(n)}$.)

By summarizing the above complexity estimates for Algorithm 10.1 and by combining them with ones of Section 9, for $Q = O(n)$, we deduce the following result.

PROPOSITION 10.1. *At a cost bounded according to parts (c) and (d) of Theorem 2.1, a normalized polynomial $\bar{p}(y)$ of (9.1)–(9.3) can be split numerically, over the unit disc $D(0,1)$, so that the two computed factors $\bar{F}_k^*(y)$ and $\bar{G}_{n-k}^*(y)$ satisfy (9.4).*

11. EXTENSION TO SPLITTING A POLYNOMIAL OVER ANY DISC

In this section, we will extend the splitting Algorithms 9.1 and 10.1 from the case of a polynomial $\bar{p}(y)$, which satisfies (9.1)–(9.3), to the case of splitting any polynomial $p(x)$ over any f-isolated disc $D(X,R)$, assuming that (2.1), (2.2) and (2.5) hold. To obtain such an extension, for splitting over an f-isolated disc, we write

$$x = X + yR, \tag{11.1}$$

$$y = \frac{(x-X)}{R}, \tag{11.2}$$

$$\bar{p}(y) = \frac{\bar{q}(y)}{\|\bar{q}(y)\|}, \tag{11.3}$$

$$\bar{q}(y) = p(x) = p(X + yR). \tag{11.4}$$

Equations (11.1) and (11.2) transform the discs $D(X,R) = \{x : |x - X| \le R\}$ and $D(0,1) = \{y : |y| \le 1\}$ into each other. Equations (11.3) and (11.4) transform $\bar{p}(y)$ into $p(x)$ (and vice versa). We will also use the bounds $|X| + R \le 1$, $R \ge 2^{-B^*}$ of (2.5). Now, if $D(X,R)$ is an (a,f)-splitting disc for $p(x)$ or an (a,B,f)-splitting disc for $p(x)$, then we will apply the following algorithm.

ALGORITHM 11.1.

INPUT: Two integers, k and n, $0 \le k \le n$; a positive ϵ, a polynomial $p(x)$ of (2.1), and an f-isolated disc $D(X,R)$ containing exactly k unknown zeros of $p(x)$, enumerated as $z_1, \ldots, z_k$, so that $D(X,R) \ni z_i$; $p(z_i) = 0$, $i = 1, \ldots, k$.

OUTPUT: Approximations $F_k^*(x)$ and $G_{n-k}^*(x)$ to the factors $F_k(x) = \prod_{i=1}^k (x - z_i)$ and $G_{n-k}(x) = p(x)/F_k(x)$, respectively, satisfying (2.4).

COMPUTATIONS.

(a) Compute the polynomial $\bar{p}(y)$ of (11.3).

(b) Recall Algorithms 9.1 and 10.1 in order to compute a pair of approximate factors $\bar{F}_k^*(y)$ (monic) and $\bar{G}_{n-k}^*(y)$ of $\bar{p}(y)$ satisfying (9.4), for an appropriate $\bar{B}$.

(c) Compute and output monic approximate factors of $p(x)$ satisfying (2.4):

$$F_k^*(x) = \bar{F}_k^*(y)R^k = \bar{F}_k^*\left(\frac{(x-X)}{R}\right)R^k, \tag{11.5}$$

$$G_{n-k}^*(x) = \frac{\bar{G}_{n-k}^*(y)\|\bar{q}(y)\|}{R^k} = \frac{\bar{G}_{n-k}^*\left((x-X)/R\right)\|\bar{q}(y)\|}{R^k}. \tag{11.6}$$

The arithmetic cost of performing Stages (a) and (c) is bounded by $O_A(\log n, n)$, due to Fact 2.1, and is clearly dominated by the bounds (given in Theorem 9.1 and Proposition 10.1) on the arithmetic cost of performing Stage (b). Furthermore, shifting and scaling the variable does not require the use of precision of computations any higher than the precision of the approximation to the coefficients of the input and/or output polynomials in the splitting algorithms (compare [1]). Thus, the Boolean complexity of Algorithm 11.1 is also dominated by the Boolean cost of performing its Stage (b).

To complete the proof of Theorem 2.1, we will next show that the bound (9.4) for $\bar{B} \geq B^* + n + \log(1/\epsilon)$ implies the bound (2.4).

PROPOSITION 11.1. *Equations (9.1) and (9.4) imply (2.4) for* $\epsilon = 2^{B^*+n-\bar{B}}$.

PROOF. Recall from (9.4) that $\bar{\Delta}(y) = \bar{\Delta}((x-X)/R) = \bar{F}_k^*(y)\bar{G}_{n-k}^*(y) - \bar{p}(y)$ and write $\Delta(x) = F_k^*(x)G_{n-k}^*(x) - p(x)$. Obtain from (11.1)–(11.6) that $\Delta(x) = \bar{\Delta}(y)\|\bar{q}(y)\|$. Therefore,

$$\|\Delta(x)\| = \left\|\bar{\Delta}\left(\frac{(x-X)}{R}\right)\right\| \|\bar{q}(y)\|. \tag{11.7}$$

The equation $\bar{\Delta}(y) = \bar{\Delta}((x-X)/R)$, together with (2.5), implies that $\|\bar{\Delta}((x-X)/R)\| \leq \|\bar{\Delta}(y)\|(2/R)^n \leq \|\bar{\Delta}(y)\|2^{B^*+n}$.

On the other hand, by applying (11.4), we obtain $\|\bar{q}(y)\| = \|p(X+yR)\| = \|\sum_i p_i(X+yR)^i\| \leq \sum_i |p_i|(|X|+R)^i$. Now, recall (2.5) and deduce that the right-hand side of the latter inequality and, therefore, also $\|\bar{q}(y)\|$ cannot exceed $\sum_i |p_i| = \|p(x)\|$. Substitute the latter upper bounds on $\|\bar{\Delta}((x-X)/R)\|$ and $\|\bar{q}(y)\|$ into (11.7) and obtain Proposition 11.1. ∎

Due to Algorithm 11.1 and Proposition 11.1, the proof of Theorem 2.1 has been completed. ∎

12. REDUCTION OF THE CONTROL OVER THE PRECISION OF THE COMPUTATION OF RECURSIVE DESCENDING TO THE STUDY OF PERTURBATION OF PADÉ APPROXIMATION

Analysis along the lines of [1] shows that $O(\bar{B}+n)$-bit precision of computing suffices at all stages of Algorithms 9.1 and 10.1, except for the stages of solving the auxiliary Toeplitz linear systems of equations, at which we seek polynomials $\hat{F}_{k,u-j}(y)$ and $\hat{G}_{n-k,u-j}(y)$ satisfying the bound (10.4). Our next objective is to prove that $O(n^2 \log n)$-bit precision suffices at the latter stages. By combining such a precision estimate with the arithmetic cost bounds of Sections 9 and 10, we arrive at the same Boolean cost bounds as ones obtained in parts (a) of Theorems 1.1 and 2.1 (provided that $bn = O(\hat{b})$ and $n = O(b)$).

We will achieve our goal by analyzing the effect of perturbing the input polynomials, $P_{u-j}(y)$, of the Padé approximation problems, where

$$P_{u-j}(y) = \frac{q_{u-j}(y)}{G_{n-k,u-j+1}(y) \bmod y^{n+1}} = \frac{F_{k,u-j}(y)}{G_{n-k,u-j}(-y) \bmod y^{n+1}} \tag{12.1}$$

(compare (10.3)). Let $P_{u-j}(y) + p_{u-j}(y)$ denote the perturbed polynomials, where we assume that

$$\|p_{u-j}(y)\| \le \frac{\|P_{u-j}(y)\|}{n^{Cn}}, \tag{12.2}$$

for some fixed positive constant C. For our purpose, it suffices to show that such a perturbation of $P_{u-j}(y)$ changes the norm of the polynomial $\hat{F}_{k,u-j}(y)\hat{G}_{n-k,u-j}(y)$ of (10.4) by at most $2^{-\hat{C}n}$, where $\hat{C} = \hat{C}(C) \to \infty$ as $C \to \infty$. Indeed, in this case, we may truncate the value of each coefficient of $P_{u-j}(y)$ so as to represent it with $O(n \log n)$ bits, without violating the bound (10.4) on the error of the output approximation to $q_{u-j}(y)$. Then, by scaling the polynomial $P_{u-j}(y)$, we may make all its coefficients Gaussian integers, of the form $a + b\sqrt{-1}$, where a and b are integers, and $|a| + |b| \le n^{O(n)}$. Then Theorem 10.1 will imply that $O(n^2 \log n)$-bit precision of computing will suffice at the stages of solving the auxiliary Toeplitz linear systems.

Next, let us show that the perturbation of $P_{u-j}(y)$ satisfying (12.2) does not affect the bound (10.4). For convenience, we will scale the variable y and the polynomials $P_{u-j}(y)$, $p_{u-j}(y)$, $F_{k,u-j}(y)$, and $G_{n-k,u-j}(y)$, so as to shift from these variables and polynomials to a new variable $x = y/\varphi$, for some

$$\varphi > 1, \tag{12.3}$$

and to polynomials $Q(x) = \alpha P_{u-j}(y)$, $q(x) = \alpha p_{u-j}(y)$, $F(x) = \beta F_{k,u-j}(y)$, and $G(x) = \gamma G_{n-k,u-j}(y)$, where the scalars $\varphi = \varphi_{u-j}$, $\alpha = \alpha_{u-j}$, $\beta = \beta_{u-j}$, and $\gamma = \gamma_{u-j}$ have been chosen such that

$$F(x) = Q(x)G(x) \bmod x^{n+1}, \tag{12.4}$$

$$F(x) = \prod_{i=1}^{k}(x - \hat{z}_i), \qquad |\hat{z}_i| \le \frac{1}{\varphi}, \quad i = 1, \dots, k, \tag{12.5}$$

$$G(x) = \prod_{i=k+1}^{n}\left(x - \frac{1}{\hat{z}_i}\right), \qquad |\hat{z}_i| \ge \varphi, \quad i = k+1, \dots, n. \tag{12.6}$$

In particular, the relations (12.3)–(12.6) imply that the pair $(F(x), G(x))$ fills up the (k, n)-entry of the Padé approximation table (also called *Padé table*) for $Q(x)$ and that the disc $D(0, 1/\varphi)$ is φ^2-isolated, with respect to the polynomial $F(x)G(x)$. (The scaling $x = y/\varphi$ implies the latter property if $f = \varphi^2$ and if the disc $D(0,1)$ is f-isolated with respect to the polynomial $q_{u-j}(y)$.) Hereafter, let $(F(x) + f(x), G(x) + g(x))$ denote the pair of polynomials that fill up the $(k, n-k)$-entry of the Padé table for $Q(x) + q(x)$; that is,

$$F(x) + f(x) = (Q(x) + q(x))(G(x) + g(x)) \bmod x^{n+1}, \tag{12.7}$$

$$\deg f(x) < k, \qquad \deg g(x) \le n - k. \tag{12.8}$$

Here and hereafter, $\deg u(x)$ denotes the degree of a polynomial $u(x)$. By the virtue of Frobenius theorem (see [69, Theorem 3.1]), the equations (12.4) and (12.7), together with the bounds on the degrees of $F(x)$, $f(x)$, $G(x)$, and $g(x)$ implied by (12.5), (12.6), and (12.8), uniquely define the rational functions $F(x)/G(x)$ and $(F(x) + f(x))/(G(x) + g(x))$, for fixed $Q(x)$ and $q(x)$. It follows that the relations (12.4)–(12.6) uniquely define the polynomials $F(x)$ and $G(x)$ too, for a fixed $Q(x)$.

Now, we may state our remaining goal as the proof of the following fact, which specifies that *the Padé approximation of $Q(x)$ is well conditioned* in the classical sense (compare [39, Chapter 3]).

FACT 12.1. There exist two positive constants C_0 and C_1 such that if the relations (12.3)–(12.8) hold and if $\|q(x)\| \le (2+1/\varphi)^{-C_0 n}$, then

$$\|f(x)\| + \|g(x)\| \le \|q(x)\| \left(2 + \frac{1}{(\varphi - 1)}\right)^{C_1 n}.$$

We will prove Fact 12.1 in the next section. In this section, we will use this fact in order to obtain its extension to the case where the bound $\deg f(x) < k$, of the assumption (12.8), is replaced by the weaker bound, $\deg f(x) \le k$, provided that all other assumptions of Fact 12.1 hold and that

$$\|q(x)\| \le \eta^n n^{-C_1 n}, \qquad \eta < \min\left\{\frac{1}{128}, \frac{(1-1/\varphi)}{9}\right\}. \tag{12.9}$$

First consider a pair of polynomials $(u(x), v(x))$ filling up the $(k, n-k)$-entry of the Padé table for a fixed polynomial $P(x)$. Unless $P(x)$ is identically 0, we can make the choice of such a pair unique by requiring that $u(x)$ be monic and have only constant common factors with $v(x)$. (Uniqueness of $u(x)$ and $v(x)$ follows for this normalization since $u(x)/v(x)$ is unique, by the virtue of Frobenius theorem.)

Clearly, the pair $(F(x), G(x))$ of (12.5) and (12.6) has been normalized in the above way. Let us assume that the pair $(F(x)+f(x), G(x)+g(x))$ has also been normalized in the same way. Then, $\deg f(x) < k$ if and only if

$$\deg(F(x) + f(x)) = k. \tag{12.10}$$

It remains to prove the following fact.

FACT 12.2. The bound (12.9) implies (12.10).

PROOF. Consider the $(n+1)$-dimensional linear space, SPACE$(n+1)$, of the coefficient vectors of polynomials $\Delta(x)$ having degrees at most n. Let $(F_\Delta(x), G_\Delta(x))$ denote the normalized pair of polynomials filling up the $(k, n-k)$-entry of the Padé table for the input polynomial $Q(x)+q(x)+\Delta(x)$. Then, clearly, the coefficient vectors of the polynomials $\Delta(x)$ for which $\deg F_\Delta(x) < k$ form an algebraic variety of a lower dimension in SPACE$(n+1)$, and therefore, there exists a sequence of polynomials $\{\Delta_h(x),\ h = 1, 2, \ldots\}$ such that $\deg F_{\Delta_h}(x) = k$, for $h = 1, 2, \ldots$, and $\|\Delta_h(x)\| \to 0$ as $h \to \infty$. Since $\deg F_{\Delta_h}(x) = k$, we may apply Fact 12.1 to the polynomial $Q(x) + q(x) + \Delta_h(x)$ replacing the polynomial $Q(x) + q(x)$ and obtain that

$$\|F_{\Delta_h}(x) - F(x)\| + \|G_{\Delta_h}(x) - G(x)\| \le n^{C_1 n} \|q(x) + \Delta_h(x)\|. \tag{12.11}$$

The inequality (12.11) bounds the norms of the $(m+2)$-dimensional row vectors $(\vec{F}_{\Delta_h}, \vec{G}_{\Delta_h})$. (Here and hereafter, we use the notation $\vec{P}$ for the coefficient vector of a polynomial $P(x)$.) Therefore, the sequence of vectors $(\vec{F}_{\Delta_h}, \vec{G}_{\Delta_h})$, $h = 1, 2, \ldots$, has a subsequence, $(\vec{F}_{\Delta_{h(i)}}, \vec{G}_{\Delta_{h(i)}})$, $i = 1, 2, \ldots$, converging to some $(m+2)$-dimensional vector, $(\vec{F}^*, \vec{G}^*)$.

Let $F^*(x)$ and $G^*(x)$ be two polynomials having the coefficient vectors $\vec{F}^*$ and $\vec{G}^*$, respectively. By considering (12.11) for $h \to \infty$, we obtain that

$$\|F^*(x) - F(x)\| + \|G^*(x) - G(x)\| \le n^{C_1 n} \|q(x)\|.$$

We will next show that

$$F^*(x) = (Q(x) + q(x)) G^*(x) \bmod x^{n+1}, \tag{12.12}$$

that is, that the pair $(F^*(x), G^*(x))$ fills up the $(k, n-k)$-entry of the Padé table for the input polynomial $Q(x) + q(x)$. For this purpose, we recall that any fixed entry of the Padé table for any fixed input polynomial $P(x)$ can be obtained from a singular homogeneous linear system of

equations, L_P, whose coefficients (except for some zeros and ones at some fixed places) are the coefficients of $P(x)$ (compare [39, equation (2.5.5)]). We observe that such systems L_P defined by input polynomials $P(x) = Q(x) + q(x) + \Delta_{h(i)}(x)$ have coefficients that converge (as $i \to \infty$) to the coefficients of the linear system L_{Q+q} defined by the input polynomial $Q(x) + q(x)$. (The convergence follows since $\Delta_{h(i)}(x) \to 0$ as $i \to \infty$.) On the other hand, the system L_P for $P(x) = Q(x) + q(x) + \Delta_{h(i)}(x)$ defines the vector $(\vec{F}_{\Delta_{h(i)}}, \vec{G}_{\Delta_{h(i)}})$ that converges to the vector $(\vec{F}^*, \vec{G}^*)$ as $i \to \infty$. It follows that the latter vector must satisfy the linear system L_P for $P(x) = Q(x) + q(x) + \Delta_{h(i)}(x)$, and therefore, (12.12) holds.

Thus, we may identify $F^*(x)$ with $F(x) + f(x)$ and $G^*(x)$ with $G(x) + g(x)$ and rewrite our previous bound as $\|f(x)\| + \|g(x)\| \le n^{C_1 n}\|q(x)\|$. By combining the latter inequality with (12.5), (12.6), (12.9), and Fact 2.7 applied to $p(x) = Q(x)$ and $p^*(x) = Q(x) + q(x)$, we obtain that all the k zeros of $F(x) + f(x)$ and no zeros of $G(x) + g(x)$ lie in the unit disc $D(0,1)$. Therefore, (12.10) holds, and the two polynomials, $F(x) + f(x)$ of degree k and $G(x) + g(x)$ of degree $n - k$, have only constant common divisors, so that Fact 12.2 follows. ∎

Facts 12.1 and 12.2 give us estimates sufficient for our purpose. Indeed, we have the f-isolated splitting disc for $q_0(y)$ and for f of (6.9), and since each Graeffe's step (10.2) squares the isolation ratio of the disc $D(0,1)$, we may choose $\varphi = (1 + 1/(120an))^{2^{i-1}}$ for a of (4.2),(4.3), when we set $Q(y) = P_i(y)$ (compare (12.1)). Even for $i = 0$, where φ is minimum, we have $\varphi > 1 + 1/(250an)$. Now, application of Facts 12.1 and 12.2 immediately implies that the bound (10.4) will be preserved under any perturbation of the polynomial $P_{u-j}(y)$ of (12.1) that satisfies (12.2) for a sufficiently large C.

13. ANALYSIS OF THE PERTURBATION OF PADÉ APPROXIMATION

We will start our proof of Fact 12.1 with some auxiliary results.

LEMMA 13.1. *Let D be a disc on the complex plane, let Γ denote its boundary circle, let $f(x)$ and $F(x)$ be two polynomials such that $\deg f(x) < \deg F(x)$ and $F(x)$ has all its zeros strictly inside the disc D, so that $F(x) \neq 0$ for $x \in \Gamma$, and let $R(x)$ be a rational function having no poles in the disc D. Then, for any x, we have*

$$f(x) = \frac{1}{2\pi\sqrt{-1}} \int_\Gamma \frac{f(t)}{F(t)} \frac{F(t) - F(x)}{t - x}\, dt, \tag{13.1}$$

$$\int_\Gamma R(t) \frac{F(t) - F(x)}{t - x}\, dt = 0. \tag{13.2}$$

PROOF. (Compare [1, Proof of Lemma 10.1; 58, III, Ch. 4, No. 163; 75, Proof of Lemma 4.6].) Cauchy's integral theorem [76] immediately implies (13.2). Furthermore, Cauchy's integral formula [76] implies equation (13.1) for x being any zero of $F(x)$. Since $\deg f(x) < \deg F(x) = N$, the equation (13.1) holds for all x if $F(x)$ has N distinct zeros. Generally, for any positive ϵ, there exists α such that $0 < \alpha < \epsilon$ and $F(x) + \alpha$ has N distinct zeros, so (13.1) holds for $F(x) + \alpha$ replacing $F(x)$. For $\alpha \to 0$, we arrive at (13.1). ∎

REMARK 13.1. Lemma 13.1 can be immediately extended to the case where D is any open set on the complex plane whose boundary Γ consists of a finite number of piecewise regular Jordan curves. One may also relax some of the assumptions of Lemmas 13.2 and 13.4.

LEMMA 13.2. *Let $F(x)$, $f(x)$, $G(x)$, $g(x)$, $Q(x)$, and $q(x)$ be polynomials satisfying (12.3)–(12.8). Let*

$$w(x) = (G(x) + g(x))G(x)q(x) \bmod x^{n+1}, \qquad \deg w(x) \le n. \tag{13.3}$$

Then

$$\|f(x)\| \le M\|F'(x)\| \le Mn\|F(x)\|,$$

where

$$M = \max_{|x|=1}\left|\frac{w(x)}{F(x)G(x)}\right|.$$

PROOF. Subtract (12.4) from (12.5) and obtain $f(x) = (Q(x)+q(x))g(x)+q(x)G(x) \bmod x^{n+1}$. Multiply both sides of this equation by $G(x)$, substitute $G(x)Q(x) = F(x)$ (from (12.4)), and obtain that $G(x)f(x) = F(x)g(x)+(G(x)+g(x))G(x)q(x) \bmod x^{n+1}$. Deduce from (12.5), (12.6), and (12.8) that $\deg(G(x)f(x) - F(x)g(x)) \le n$ and rewrite the latter equation as follows:

$$G(x)f(x) = F(x)g(x) + w(x), \tag{13.4}$$

where $w(x)$ is defined by (13.3). Divide both sides of (13.4) by $F(x)G(x)$ and obtain that

$$\frac{f(x)}{F(x)} = \frac{g(x)}{G(x)} + \frac{w(x)}{F(x)G(x)}.$$

Substitute this equation into (13.1), then apply (13.2) for $R(t) = g(t)/G(t)$, and obtain that

$$f(x) = \frac{1}{2\pi\sqrt{-1}}\int_\Gamma \frac{w(t)}{F(t)G(t)}\frac{F(t)-F(x)}{t-x}\,dt.$$

Apply this identity coefficientwise and deduce Lemma 13.2. ∎

The estimate of the next lemma is immediate.

LEMMA 13.3. *Let two polynomials $F(x)$ and $G(x)$ satisfy (12.5) and (12.6) for some $\varphi > 1$. Then*

$$\min_{|x|=1}|F(x)G(x)| \ge \varphi_-^n,$$

where

$$\varphi_- = 1 - \frac{1}{\varphi}. \tag{13.5}$$

Next, we will deduce the following result.

LEMMA 13.4. *Suppose that the relations (12.3)–(12.8), (13.3), and (13.4) hold. Then*

$$\|g(x)\| \le 2^{n-1}\varphi_+^{n-k}\frac{\left(\|f(x)\| + \varphi_+^{n-k}\|q(x)\|\right)}{\left(1 - 2^{n-1}\varphi_+^{n-k}\|q(x)\|\right)},$$

where

$$\varphi_+ = 1 + \frac{1}{\varphi} < 2. \tag{13.6}$$

PROOF. Recall that $\deg(F(x)g(x)) \le n$, and deduce from Fact 2.4 that $\|F(x)\|\,\|g(x)\| \le 2^{n-1} \times \|F(x)g(x)\|$. We have $\|F(x)\| \ge 1$, due to (12.5), and consequently, $\|g(x)\| \le \|F(x)\|\,\|g(x)\| \le 2^{n-1}\|F(x)g(x)\|$. Substitute the bound $\|F(x)g(x)\| \le \|G(x)\|\,\|f(x)\| + \|w(x)\|$, implied by (13.4), and obtain that $\|g(x)\| \le 2^{n-1}(\|G(x)\|\,\|f(x)\| + \|w(x)\|)$. From (13.3), we have the inequality $\|w(x)\| \le (\|G(x)\| + \|g(x)\|)\,\|G(x)\|\,\|q(x)\|$. Combine the two latter inequalities and the bound $\|G(x)\| \le \varphi_+^{n-k}$, implied by (12.6) and (13.6), and obtain that

$$\|w(x)\| \le \varphi_+^{n-k}\left(\varphi_+^{n-k} + \|g(x)\|\right)\|q(x)\|, \tag{13.7}$$

$\|g(x)\| \le 2^{n-1}(\varphi_+^{n-k}\|f(x)\| + \varphi_+^{n-k}(\varphi_+^{n-k} + \|g(x)\|)\,\|q(x)\|) = 2^{n-1}\varphi_+^{n-k}(\|f(x)\| + \varphi_+^{n-k}\|q(x)\| + \|g(x)\|\,\|q(x)\|)$. It follows that $\|g(x)\|\,(1 - 2^{n-1}\varphi_+^{n-k}\|q(x)\|) \le 2^{n-1}\varphi_+^{n-k}(\|f(x)\| + \varphi_+^{n-k}\|q(x)\|)$, and we arrive at Lemma 13.4. ∎

COROLLARY 13.1. *Assume the relations (12.3)–(12.8), (13.5), and (13.6) and let* $2^n\|q(x)\| \le 1/\varphi_+^{n-k}$, $n2^{n+1}\varphi_+^{n-k}(\varphi_+/\varphi_-)^n\|q(x)\| \le 1$. *Then*

(a) $\|f(x)\| \le 4n\varphi_+^{n-k}(\varphi_+/\varphi_-)^n\|q(x)\|$,

(b) $\|g(x)\| \le 2^n\varphi_+^{2n-2k}(1+4n(\varphi_+/\varphi_-)^n)\|q(x)\|$.

PROOF. Combining the first of the two assumed upper bounds on $\|q(x)\|$ with Lemma 13.4 implies that

$$\|g(x)\| \le 2^n\varphi_+^{n-k}\left(\|f(x)\| + \varphi_+^{n-k}\|q(x)\|\right). \tag{13.8}$$

On the other hand, from Lemmas 13.2 and 13.3, we have

$$\|f(x)\| \le n\|F(x)\|\varphi_-^{-n}\max_{|x|=1}|w(x)|.$$

Now, we deduce that $\|F(x)\| \le \varphi_+^k$ (see (12.5) and (13.6)) and $\max_{|x|=1}|w(x)| \le \|w(x)\| \le \varphi_+^{n-k}(\varphi_+^{n-k} + \|g(x)\|)\,\|q(x)\|$ (compare (13.7)). It follows that

$$\|f(x)\| \le n\left(\frac{\varphi_+}{\varphi_-}\right)^n\left(\varphi_+^{n-k} + \|g(x)\|\right)\|q(x)\|.$$

Combine this bound with (13.8) and obtain that

$$\begin{aligned}\|f(x)\| &\le n\left(\frac{\varphi_+}{\varphi_-}\right)^n\left(\varphi_+^{n-k} + 2^n\varphi_+^{n-k}\left(\|f(x)\| + \varphi_+^{n-k}\|q(x)\|\right)\right)\|q(x)\| \\ &= n\varphi_+^{n-k}\left(\frac{\varphi_+}{\varphi_-}\right)^n\left(1 + 2^n\left(\|f(x)\| + \varphi_+^{n-k}\|q(x)\|\right)\right)\|q(x)\|.\end{aligned}$$

Therefore,

$$\left(1 - n2^n\varphi_+^{n-k}\left(\frac{\varphi_+}{\varphi_-}\right)^n\|q(x)\|\right)\|f(x)\| \le n\varphi_+^{n-k}\left(\frac{\varphi_+}{\varphi_-}\right)^n\left(1 + 2^n\varphi_+^{n-k}\|q(x)\|\right)\|q(x)\|.$$

Recall the second upper bound on $\|q(x)\|$ assumed in Corollary 13.1 and deduce that $\|f(x)\| \le 2n\varphi_+^{n-k}(\varphi_+/\varphi_-)^n(1 + 2^n\varphi_+^{n-k}\|q(x)\|)\,\|q(x)\|$. Simplify this expression by using the first upper bound on $\|q(x)\|$ assumed in Corollary 13.1 and arrive at part (a) of Corollary 13.1. Combine the bounds of part (a) and (13.8) and obtain part (b) of Corollary 13.1. ∎

Fact 12.1 immediately follows from Corollary 13.1.

APPENDIX A
A CORRELATION BETWEEN THE CARDINALITIES OF INTERSECTION AND UNION

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

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

PROOF. We only need this result for $h = 3$ and will prove it for this h by following [3].

Let s_i and s_{ij} denote the set cardinalities, $s_i = |S_i - (S_j \cup S_k)|$, $s_{ij} = |(S_i \cap S_j) - I|$, where i, j, k are distinct integers chosen among 1, 2, and 3 (in any order). Then, clearly,

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

By subtracting twice the latter equation from the sum of the preceding three equations, we obtain that

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

which implies Proposition A.1 for $h = 3$, since $s_i \geq 0$, $i = 1, 2, 3$. ∎

In Section 3, we use Proposition A.1, in the case where S_1, S_2, and S_3 denote the three sets of the zeros of $p(x)$ lying in three fixed annuli.

APPENDIX B
EXTENDING ROLLE'S THEOREM TO THE COMPLEX CASE

We will follow [27] and will prove part (a) of Theorem 4.1. We will start with recalling a little known but simple lemma.

LEMMA B.1. *[27] Let $v_1, \ldots, v_k$ denote the vertices of a simplex σ in the $(k-1)$-dimensional real space $\mathbb{R}^{k-1}$. Let $c_1, \ldots, c_k$ be k complex points in C and let $\alpha : \mathbb{R}^{k-1} \to C$ be the real affine map taking v_i to c_i. Let f be an analytic function on the image of σ. Let $[c_1, c_2, \ldots, c_k]$ f denote the divided difference operator applied to f and let $v(\vec{t}\,)$ be the standard volume form on $\mathbb{R}^{k-1}$. Then*

$$[c_1, c_2, \ldots, c_k]f = \int_{\sigma} f^{(k-1)}\left(\alpha\left(\vec{t}\,\right)\right) dv\left(\vec{t}\,\right). \tag{B.1}$$

PROOF OF THEOREM 4.1, PART (a). Apply Lemma B.1 where $k = l$, $f(x) = p(x)$, and $c_1, \ldots, c_k$ are the zeros of $p(x)$. Then the left-hand side of (B.1) vanishes. Therefore, so does the right-hand side too. This means that its integrand must vary by at least π, and this implies the condition on the zeros of $p^{(k-1)}(x)$ of part (a) of Theorem 4.1 for $k = l$. ∎

APPENDIX C
IMPROVING AN ARITHMETIC TIME BOUND

Let us decrease the arithmetic time bounds of Theorems 1.1, 1.2, and 2.1, by performing Stage 3 of Algorithm 10.1 as follows.

SUBALGORITHM C.1. (Compare Remark 10.1.) Denote $r_u(y) = F_{k,u}(y)$, then recursively compute

$$r_{u-i}(y) = r_{u-i+1}(y^2) \bmod q_{u-i}(y), \qquad \text{for } i = 1, 2, \ldots, u-1, \tag{C.1}$$

and finally compute and output

$$F_k(x) = \gcd\left(q_0(x), r_1\left(x^2\right)\right). \tag{C.2}$$

To prove *correctness* of this algorithm, first obtain from (10.2) the following extension of (10.3):

$$F_k(x) = F_{k,0}(x) = \gcd\left(q_0(x), F_{k,u}\left(x^{2^u}\right)\right). \tag{C.3}$$

On the other hand, (10.2) implies that $q_i(x)$ divides $q_{i+1}(x^2)$ for every i, so that

$$\gcd\left(q_i(x), F_{k,u}\left(x^{2^u}\right)\right) = \gcd\left(q_i(x), q_{i+1}\left(x^2\right), F_{k,u}\left(x^{2^u}\right)\right), \qquad \text{for } i = 0, 1, \ldots, u-1.$$

Therefore, recursive modular reduction of $F_{k,u}(x^{2^u})$, performed according to (C.1) and (C.2), defines the desired gcd of (C.3). ∎

REMARK C.1. Subalgorithm C.1 can be modified if we redefine $r_u(y)$ as $G_{n-k,u}(y)$. Then application of (C.1) will enable us to output $G_{n-k}(x) = \gcd(q_0(x), r_1(x^2))$, since $G_{n-k}(x) = \gcd(q_0(x), G_{n-k,u}(x^{2^u}))$.

Performing Stage 3 of Algorithm 10.1 by means of Subalgorithm C.1, we replace $u-1$ (out of u) computations of the gcds by polynomial divisions, which enables us to improve the bounds of parts (d) of Theorems 1.1 and 2.1 as follows:

$$SAC_Z(b,n) = O_A\left((\log n)^2\left(\log \bar{b}\right)n, 1\right),$$
$$SAC_Z^*\left(\hat{b},n\right) = O_A\left((\log n)^2\left(\log \hat{b}\right)n, 1\right),$$
$$SAC_S\left(\bar{B},n\right) = O_A\left((n\log n)\log \bar{B}, 1\right).$$

Similar minor improvements follow for parts (b) and (c) of Theorems 1.1 and 2.1.

Alternatively, exactly the same minor asymptotic improvements of the estimates of parts (b)–(d) of Theorems 1.1 and 2.1 can be obtained by replacing Algorithm 10.1 and Subalgorithm C.1 by Algorithm 1 of [5], by Algorithm 3.1 of [6] (adjusted to the problem of splitting $p(x)$ into two factors, rather than to approximating all the zeros of $p(x)$), or by the algorithm of [77]. ∎

REMARK C.2. Subalgorithm C.1 involves recursive polynomial divisions, which requires us to increase the precision of the computations and their overall Boolean cost. The algorithms of [5,6,77] have the same feature. As a result, application of all these algorithms would only support Boolean complexity bounds that are much inferior to ones of parts (a) of Theorems 1.1 and 2.1.

REFERENCES

1. A. Schönhage, The fundamental theorem of algebra in terms of computational complexity, (manuscript), Math. Dept., University of Tübingen, Tübingen, Germany, (1982).
2. V.Y. Pan, Sequential and parallel complexity of approximate evaluation of polynomial zeros, *Computers Math. Applic.* **14** (8), 591–622 (1987).
3. C.A. Neff and J.H. Reif, An $O(n^{1+\epsilon}\log b)$ algorithm for the complex root problem, In *Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 540–547, IEEE Computer Society Press, (1994).
4. C.A. Neff, Specified precision polynomial root isolation is in NC, *Journal of Computer and System Sciences* **48**, 429–463 (1994).
5. D. Bini, Complexity of parallel polynomial computations, In *Proc. Parallel Computing: Methods, Algorithms, Applications*, (Edited by J. Evans and C. Nodari), pp. 115–126, Adam Hilger, Bristol, (1989).
6. D. Bini and L. Gemignani, On the complexity of polynomial zeros, *SIAM J. on Sci. Stat. Computing* **8** (21), 781–799 (1992).
7. P. Kirrinnis, Polynomial factorization and partial fraction decomposition by simultaneous Newton's iteration, (extended abstract), (1994).
8. V.Y. Pan, Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros, In *Proc. 27th Annual ACM Symposium on Theory of Computing*, pp. 741–750, ACM Press, New York, (1995).
9. H. Weyl, Randbemerkungen zu Hauptproblemen der Mathematik, II, Fundamentalsatz der Algebra and Grundlagen der der Mathematik, *Math. Z.* **20**, 131–151 (1924).
10. L.E.J. Brouwer and B. de Loer, Intuitionisher Beweis des Fundamentalsatzes der Algebra, *Amsterdam Konigl. Acad. Van Wetenschapen, Proc.* **27**, 186–188 (1924); also in *Coll. Works*, (Edited by L.E.J. Brouwer), North-Holland, Amsterdam, (1975).
11. H. Samet, The quadtree and related hierarchical data structures, *Computing Surveys* **16** (2), 187–260 (1984).
12. H. Senoussi, A quadtree algorithm for template matching on pyramid computer, *Theor. Comp. Science* **136**, 387–417 (1994).
13. V.Y. Pan, Weyl's quadtree algorithm for the unsymmetric eigenvalue problem, *Appl. Math. Lett.* **8** (5), 87–88 (1995).
14. C.F. Gauss, Demonstratio nova theorematics omnen functionem algebraicam rationalem integram unius variablis in factores reales primi vel secondi gradus resolvi posse, In *Gesammelte Werke*, III, (1799); also in C.F. Gauss, *Werke*, Band X, Georg Olms Verlag, New York, (1973).
15. E.T. Bell, *The Development of Mathematics*, McGraw-Hill, New York, (1940).
16. C.A. Boyer, *A History of Mathematics*, Wiley, New York, (1968).
17. B. Dejon and P. Henrici, Editors, *Constructive Aspects of the Fundamental Theorem of Algebra*, Wiley, London, (1969).

18. A.S. Householder, *The Numerical Treatment of a Single Nonlinear Equation*, McGraw-Hill, New York, (1970).
19. P. Henrici, *Applied and Computational Complex Analysis*, Vol. 1, Wiley, New York, (1974).
20. S. Smale, The fundamental theorem of algebra and complexity theory, *Bull. Amer. Math. Soc.* **4**, 1–36 (1981).
21. V.Y. Pan, Solving a polynomial equation: Some history and recent progress, (preprint), (1995).
22. A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, (1976).
23. R. Karp and V. Ramachandran, A survey of parallel algorithms for shared memory machines, In *Handbook for Theoretical Computer Science*, (Edited by J. van Leeuwen), pp. 869–941, North-Holland, Amsterdam, (1990).
24. P. Turan, On the approximate solution of algebraic functions (in Hungarian), *Comm. Math. Phys. Class Hung. Acad., XVIII*, 223–236 (1968).
25. P. Henrici and I. Gargantini, Uniformly convergent algorithms for the simultaneous approximation of all zeros of a polynomial, In *Constructive Aspects of the Fundamental Theorem of Algebra*, (Edited by B. Dejon and P. Henrici), Wiley, London, (1969).
26. J. Renegar, On the worst-case arithmetic complexity of approximating zeros of polynomials, *J. of Complexity* **3** (2), 90–113 (1987).
27. D. Coppersmith and C.A. Neff, Roots of a polynomial and its derivatives, In *Proc. 5^{th} Ann. ACM-SIAM Symp. on Discrete Algorithms*, pp. 271–279, ACM Press, New York, and SIAM Publications, Philadelphia, (1994).
28. V.Y. Pan, Deterministic improvement of complex polynomial factorization based on the properties of the associated resultant, *Computers Math. Applic.* **30** (2), 71–94 (1995).
29. V.Y. Pan, New techniques for approximating complex polynomial zeros, In *Proc. 5^{th} Ann. ACM-SIAM Symp. on Discrete Algorithms*, pp. 260–270, ACM Press, New York, and SIAM Publications, Philadelphia, (1994).
30. 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. on Comput.* **17**, 1081–1092 (1989).
31. M. Ben-Or and P. Tiwari, Simple algorithm for approximating all roots of a polynomial with real roots, *J. of Complexity* **6**, 417–442 (1990).
32. D. Bini and V.Y. Pan, Parallel complexity of tridiagonal symmetric eigenvalue problem, In *Proc. 2^{nd} Ann. ACM-SIAM Symp. on Discrete Algorithms*, pp. 384–393, ACM Press, New York, and SIAM, Philadelphia, (1991).
33. V.Y. Pan, New resultant inequalities and complex polynomial factorization, *Journal on Computing* **23** (5), 934–950 (1994).
34. M. Marden, *Geometry of Polynomials*, Amer. Math. Soc., Providence, RI, (1966).
35. Y.P. Ofman, On the algebraic complexity of discrete functions, *Dokl. Acad. Nauk SSSR* **145** (1), 48–51 (1962); English translation in *Soviet Physics-Dokl.* **7** (7), 589–591, (1963).
36. A. Schönhage and V. Strassen, Schnelle Multiplikation grosse Zahlen (in German), *Computing* **7**, 281–292 (1971).
37. J.E. Savage, *The Complexity of Computing*, Wiley and Sons, New York, (1976).
38. D.E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, Vol. 2, Addison-Wesley, Reading, MA, (1981).
39. D. Bini and V.Y. Pan, *Polynomial and Matrix Computations, Vol. 1: Fundamental Algorithms*, Birkhäuser, Boston, (1994).
40. A. Karatsuba and Yu. Ofman, Multiplication of multidigit numbers on automata, *Soviet Physics Dokl.* **7**, 595–596 (1963).
41. E. Durand, *Solutions Numériques des Equations Algébriques, Tome I: Equations du Type $F(x) = 0$: Racines d'un Polynome*, Masson, Paris, (1960).
42. I.O. Kerner, Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen, *Numer. Math.* **8**, 290–294 (1966).
43. O. Aberth, Iteration methods for finding all zeros of a polynomial simultaneously, *Math. Comput.* **27** (122), 339–344 (1973).
44. D. Bini, Numerical computation of polynomial zeros by Aberth's method, (manuscript), Dept. of Math., University of Pisa, (1994).
45. H.S. Wilf, A global bisection algorithm for computing the zeros of polynomials in the complex plane, *J. ACM* **25**, 415–420 (1978).
46. V.Y. Pan, On approximating complex polynomial zeros: Modified quadtree (Weyl's) construction and improved Newton's iteration, (preprint), (1994).
47. V.Y. Pan and E. Linzer, A new approach to bisection acceleration for the symmetric eigenvalue problem, (preprint), (1995).
48. S. Smale, Algorithms for solving equations, In *Proc. International Congress of Mathematicians,*, Berkeley, CA, 1986, pp. 172–195, American Math. Society, Providence, RI, (1987).
49. S. Smale, On the efficiency of algorithms of analysis, *Bull. Amer. Math. Soc.* **4**, 1–36 (1986).
50. M.-H. Kim and S. Sutherland, Polynomial root-finding algorithms and branched covers, *SIAM J. on Computing* **23** (2), 415–436 (1994).

51. M. Shub and S. Smale, Complexity of Bezout's Theorem I: Geometric aspects, *J. of the Amer. Math. Soc.* **6**, 459–501 (1993).
52. M. Shub and S. Smale, Complexity of Bezout's Theorem II: Volumes and probabilities, In *Computational Algebraic Geometry, Progress in Mathematics*, Vol. 109, (Edited by F. Eyssette and A. Galligo), pp. 267–285, Birkhäuser, (1993).
53. M. Shub and S. Smale, Complexity of Bezout's Theorem III: Condition number and packing, *J. of Complexity* **9**, 4–14 (1993).
54. M. Shub and S. Smale, Complexity of Bezout's Theorem IV: Probability of success, extensions, Research Report RC 18921, IBM T.J. Watson Research Center, Yorktown Heights, NY, (1993).
55. M. Shub and S. Smale, Complexity of Bezout's Theorem V: Polynomial time, Report No. 236, Centre de Recerca Matemàtica, Institut d'Estudis Catalanas, Barcelona, Spain, (1993).
56. A. Van der Sluis, Upper bounds for roots of polynomials, *Numer. Math.* **15**, 250–262 (1970).
57. P. Turan, Power sum method and approximative solution of algebraic equations, *Math. Comp.* **29**, 311–318 (1975).
58. G. Polýa and G. Szegö, *Aufgaben und Lehrsätze aus der Analysis*, Vol. 1, Dover Publications, New York, (1945).
59. R. Schätzle, Zur Störung der Nullstellen von komplexen Polynomen, Dimplomarbeit, (in German), Math. Dept., Univ. of Bonn, Bonn, Germany, (1990).
60. A. Schönhage, Quasi-GCD computations, *J. of Complexity* **1**, 118–137 (1985).
61. M.J. Fischer and M.S. Paterson, String matching and other products, *SIAM-AMS Proc.* **7**, 113–125 (1974).
62. V.Y. Pan, The bit-operation complexity of the convolution of vectors and of the DFT, Tech. Rep. 80-6, Computer Science Dept., SUNYA, Albany, NY, (1980); abstract in *Bulletin of the EATCS* **14**, 95, (1981).
63. V.Y. Pan, *How to Multiply Matrices Faster*, Lecture Notes in Computer Science, Springer, Berlin, (1984).
64. D. Bini and V.Y. Pan, Polynomial division and its computational complexity, *J. of Complexity* **2**, 179–203 (1986).
65. L.M. Delves and J.N. Lyness, A numerical method for locating zeros of an analytic function, *Math. Comp.* **21**, 543–560 (1967).
66. J. Schröder, Über das Newtonsche Verfahren, *Arch. Rat. Mech. Anal.* **1**, 154–180 (1957).
67. V.Y. Pan, E. Landowne and A. Sadikou, Polynomial division with a remainder by means of evaluation and interpolation, *Information Process. Letters* **44**, 149–153 (1992); In *Proc. 3rd IEEE Conf. on Parallel and Distributed Proc.*, pp. 212–218, IEEE Computer Society Press, (1991).
68. J.H. Reif and S.R. Tate, Optimal size division circuits, *SIAM J. on Computing* **19** (5), 912–925 (1990).
69. W.B. Gragg, The Padé table and its relation to certain algorithms of numerical analysis, *SIAM Review* **14** (1), 1–62 (1972).
70. V.Y. Pan, Parametrization of Newton's iteration for computations with structured matrices and applications, *Computers Math. Applic.* **24** (3), 61–75 (1992).
71. V.Y. Pan, Fast and efficient parallel algorithms for the exact inversion of integer matrices, In *Proc. 5th Ann. Conference on Foundation of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science, Vol. 206, pp. 504–521, Springer, Berlin, (1985).
72. V.Y. Pan, Complexity of parallel matrix computations, *Theoretical Computer Science* **54**, 65–85 (1987).
73. V.Y. Pan, Parallel solution of Toeplitz-like linear systems, *J. of Complexity* **8**, 1–21 (1992).
74. V.Y. Pan, Concurrent iterative algorithm for Toeplitz-like linear systems, *IEEE Trans. on Parallel and Distributed Systems* **4** (5), 592–600 (1993).
75. P. Kirrinnis, Fast computation of numerical partial fraction decompositions and contour integrals of rational functions, In *Proc. ACM Annual Int. Symp. on Symb. and Alg. Comput. (ISSAC92)*, (Edited by P.S. Wang), pp. 16–26, ACM Press, New York, (1992).
76. L. Ahlfors, *Complex Analysis*, McGraw-Hill, New York, (1979).
77. V.Y. Pan, Faster splitting a polynomial into factors over a fixed disc, (preprint), (1994).
78. G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, (1989).