Old and New Nearly Optimal Polynomial Root-Finders

Victor Y. Pan[1,2(✉)]

[1] Department of Computer Science, Lehman College of the City University of New York, Bronx, NY 10468, USA
victor.pan@lehman.cuny.edu
[2] Ph.D. Programs in Mathematics and Computer Science, The Graduate Center of the City University of New York, NewYork, NY 10036, USA
http://comet.lehman.cuny.edu/vpan/

Abstract. Univariate polynomial root-finding has been studied for four millennia and still remains the subject of intensive research. Hundreds if not thousands of efficient algorithms for this task have been proposed and analyzed. Two nearly optimal solution algorithms have been devised in 1995 and 2016, based on recursive factorization of a polynomial and subdivision iterations, respectively, but both of them are superseded in practice by Ehrlich's functional iterations. By combining factorization techniques with Ehrlich's and subdivision iterations we devise a variety of new root-finders. They match or supersede the known algorithms in terms of their estimated complexity for root-finding on the complex plane, in a disc, and in a line segment and promise to be practically competitive.

Keywords: Polynomial root-finding · Deflation · Polynomial factorization · Functional iterations · Subdivision · Real root-finding

2000 Math. Subject Classification: 65H05· 26C10· 30C15

1 Introduction

1. The Problem and Three Known Efficient Algorithms. Univariate polynomial root-finding has been the central problem of mathematics since Sumerian times (see [1,2,33,34]) and still remains the subject of intensive research due to applications to signal processing, control, financial mathematics, geometric modeling, and computer algebra (see the books [28,30], a survey [19], the recent papers [11,12,24,40,45,48], and the bibliography therein).

Hundreds if not thousands of efficient polynomial root-finders have been proposed. The algorithm of [32] and [37], extending the previous progress in

M. England et al. (Eds.): CASC 2019, LNCS 11661, pp. 393–411, 2019.
https://doi.org/10.1007/978-3-030-26831-2_26

[15,31,50], first computes numerical factorization of a polynomial into the product of its linear factors and then approximates the roots; it solves both tasks in nearly optimal Boolean time – almost as fast as one can access the input coefficients with the precision required for these tasks.[1]

Since 2000 the root-finder of the user's choice has been the package MPSolve,[2] implementing Ehrlich's iterations of [18], also known from their rediscovery by Aberth in 1973. In 2016, a distinct nearly optimal polynomial root-finder appeared in [11,12], based on subdivision iterations. That algorithm promises to compete with MPSolve for root-finding in a disc on the complex plain [24], but less likely for the approximation of all roots of a polynomial unless our innovations raise practical efficiency of subdivision to a much higher level.

2. New Hybrid Algorithms. We propose new hybrid root-finders seeking *synergistic combination* of the known techniques.

We first recall that [32] and [37] factorize a polynomial by splitting it into the product of two factors of comparable degrees and then recursively splitting the factors in a similar way as long as their degree exceeds 1. This advanced recursive construction turned out to be hard to implement, but its basic deflation algorithm developed by Schönhage in [50], traced back to Delves and Lyness [15], and hereafter referred to as *DLS algorithm* or *DLS deflation* can be handled quite readily. Presently we devise new hybrid root-finders by incorporating the DLS algorithm into Ehrlich's and subdivision iterations. In both cases deflation enables us to apply root-finding to smaller degree factors, possibly having fewer root clusters rather than to the input polynomial.

We also enhance the efficiency of subdivision iterations by incorporating a very fast and robust sub-algorithm of DLS deflation that computes the number of roots in a disc as the sum of the 0th powers of all roots in that disc. This is *dramatic improvement of root counting* in the papers [11,12], which listed root counting algorithm as their main novelty compared to their predecessors.

We also propose an additional simplification of real root-finding based on fast estimation of the distances of real roots from the origin.

Our hybrid root-finders are nearly optimal and can become the user's choice. Their implementation, testing and refinement are major challenges.

3. Some Extensions. Our hybrid algorithms can be readily extended to various functional iterations such as Newton's and Weierstrass's that approximate all roots of a polynomial, but we also extend our approach to nearly optimal root-finding in a disc and a line interval. In both cases non-costly removal of

[1] Required precision and Boolean time are smaller by a factor of d, the degree of the input polynomial, at the stage of numerical polynomial factorization, which has various important applications to modern computations, besides root-finding, e.g., to time series analysis, Wiener filtering, noise variance estimation, co-variance matrix computation, and the study of multi-channel systems (see Wilson [59], Box and Jenkins [7], Barnett [3], Demeure and Mullis [16] and [17], Van Dooren [56]).

[2] Some competition came in 2001 from the package EigenSolve of [20], but the latest version of MPSolve of [10] has combined the benefits of both packages.

the external roots by means of deflation promises substantial advantage over the customary solution of these tasks by means of application of MPSolve and subdivision to the original polynomial of higher degree.

We hope that our work will motivate further efforts towards synergistic combination of some efficient techniques well- and less- known for polynomial root-finding (see, e.g., the little explored methods of [40] and [48]).

Devising practical and nearly optimal factorization algorithms is still a research challenge because for that task both Ehrlich's and subdivision iterations are slower by at least a factor of d than [32] and [37].

4. Variations of Deflation. With Ehrlich's iterations we can combine the DLS deflation, but also two simpler algorithms (see Sect. 4.3). One of them only involves shift and scaling of the variable and forward and inverse FFTs and allows us to represent an input polynomial with a black box for its evaluation rather than with coefficients. In [39] and [25] we alternatively combine Ehrlich's iterations with implicit deflation, which ensures preserving sparseness of an input and avoiding coefficient growth.

With subdivision iterations we combine the DLS deflation, which is highly efficient and relatively simple but has been too little (if at all) used by researchers since [37]. It has been hidden in the long paper [50], within a realm of intricate and advanced techniques for the theoretical estimation of asymptotic Boolean complexity where extremely accurate polynomial factorization is required, but some of these techniques can help enhance performance of the most popular root-finders.

Now suppose that the root set of a factor is a strongly isolated cluster of w roots of p having a small diameter. Such a cluster may appear in subdivision process and then can be readily detected. In this special case we can perform deflation at a low cost by means of shifting the origin into the cluster and then reducing the resulting polynomial $q(x)$ modulo x^{w+1} (see Sect. 5.4).

5. Organization of the Paper. We state four variations of the main root-finding problem in Sect. 2 and deduce lower bounds on their Boolean complexity in Sect. 3. We cover computation of a factor with root set in a fixed disc in Sect. 5 and our hybrids of deflation with functional iterations in Sect. 4 and with subdivision in Sect. 6. In Sect. 7 we devise a new nearly optimal polynomial root-finder on a line segment. We refer the reader to [42, Appendix] for *concise exposition* of factorization algorithms of [50] by Schönhage and [26] by Kirrinnis (67 pages) and for some other auxiliary and complementary algorithms and techniques for polynomial root-finding.

2 Four Fundamental Computational Problems

Problem 1. *Univariate Polynomial Root-finding.* Given a positive b and the coefficients $p_0, p_1, \dots, p_d$ of a univariate polynomial $p(x)$,

$$p(x) = \sum_{i=0}^{d} p_i x^i = p_d \prod_{j=1}^{d} (x - x_j), \ p_d \neq 0. \tag{1}$$

approximate all d roots[3] $x_1, \dots, x_d$ within the error bound $1/2^b$ provided that $\max_{j=0}^{d} |x_j| \le 1$. We can ensure the latter customary assumption at a dominated computational cost by first approximating the root radius $r_1 = \max_{j=1}^{d} |x_j|$ and then scaling the variable x (cf., e.g., [33]).

Before proceeding any further we recall some **Basic Definitions.**

- Hereafter we freely denote polynomials $p(x)$, $t(x) = \sum_i t_i x^i$, $u(x) = \sum_i u_i x^i$ etc. by p, t, u etc. unless this can cause confusion.
- We use the norm $|u| = \sum_i |u_i|$ for $u = \sum_i u_i x^i$.
- $d_u := \deg(u)$ denotes the degree of a polynomial u; in particular $d_p = d$.
- ϵ-*cluster* of roots of p is a root set lying in a disc of radius ϵ; in particular a 0-cluster of m roots of p is its root of multiplicity m.

Problem 2. *Approximate Factorization of a Polynomial.* Given a positive b' and the coefficients $p_0, p_1, \dots, p_d$ of a polynomial $p = p(x)$ of (1), compute $2d$ complex numbers u_j, v_j for $j = 1, \dots, d$ such that

$$|p - \prod_{j=1}^{d} (u_j x - v_j)| \le 2^{-b'} |p|. \tag{2}$$

Problem 3. *Polynomial root-finding in a disc.* This is Problem 1 restricted to root-finding in a disc on the complex plain for a polynomial p that has no roots lying outside the disc but close to it.

Problem 4. *Polynomial root-finding in a line segment.* This is Problem 1 restricted to root-finding in a line segment for a polynomial p that has no roots lying outside the segment but close to it.

The above concept "close" is quantified in Definition 1 in the case of Problem 3 and is extended to Problem 4 via its reduction to Problem 3 in Sect. 1.

Remark 1. It is not easy to optimize *working precision* for the solution of Problems 1–4 a priori, but we nearly optimize it *by action*, that is, by applying the solution algorithms with recursively doubled or halved precision and monitoring the results (see Sect. 4.2 and recall similar policies in [5,10,12,45]).

Remark 2. It is customary to reduce Problems 3 and 4 to root-finding in the unit disc

$$D(0,1) := \{x : \ |x| < 1\}$$

and unit segment

$$S[-1,1] := \{x : \ -1 \le x \le 1\}$$

by means of shifting and scaling the variable. Then the working precision and Boolean cost grow but within the nearly optimal bounds.

[3] We count m times a root of multiplicity m.

3 Boolean Complexity: Lower Estimates and Nearly Optimal Upper Bounds

Proposition 1. *The solution of Problem 2 involves at least db' bits of memory and at least as many Boolean (bit-wise) operations.*

Proof. The solution of Problem 2 is given by the $2d$ coefficients u_j and v_j of the d linear factors $u_j x - v_j$ of p for $j = 1, \dots, d$. Let $u_j = 1$ and $1/2 < |v_j| < 1$ for all j. Then each v_j must be represented with b' bits and hence all v_j must be represented with db' bits in order to satisfy (2). A Boolean operation outputs a single bit, and so we need at least db' operations in order to output db' bits.

Next we bound from below the Boolean complexity of Problems 1, 3 and 4.

Lemma 1. *Let $p(x) = (x - x_1)^m f(x)$ for a polynomial $f(x)$ and a positive integer m. Fix a positive b. Then the polynomial $p_j(x) = p(x) + 2^{(j-m)b}(x - x_1)^j f(x)$ has $m - j$ roots $x_1 + \omega_{m-j}^i 2^{-b}$ for $i = 0, \dots, m - j - 1$ and $\omega_{m-j} = \exp(2\pi\mathbf{i}/(m-j))$ denoting a primitive $(m-j)$th root of unity, such that $\omega_{m-j}^{m-j} = 1$, $\omega_{m-j}^i \neq 1$ for $0 < i < m - j$.*

Proof. Observe that $p_j(x) = ((x - x_1)^{m-j} + 2^{(j-m)b})(x - x_1)^j f(x)$ and consider the roots of the factor $(x - x_1)^{m-j} + 2^{(j-m)b}$.

Corollary 1. *Under the assumption of Lemma 1 write $f := \lceil \log_2 |f| \rceil$ and $g := \sum_{j=1}^{m-1} \lceil \log_2 |g_j| \rceil$. Then one must process at least*

$$\mathcal{B}_p = \Big(d - m + 1 + \frac{m-1}{2}\Big)mb - f - g \tag{3}$$

bits of the coefficients of p and must perform at least $\mathcal{B}_p/2$ Boolean operations in order to approximate the m-multiple root x_1 of p within $1/2^b$.

Proof. By virtue of Lemma 1 the perturbation of the coefficients $p_0, \dots, p_{d-m}$ of $p(x)$ by $|f|/2^{mb}$ turns the $(m-j)$-multiple root x_1 of the factor $(x - x_1)^{m-j}$ of $p(x)$ into $m - j$ simple roots $p_j(x)$, all lying at the distance $1/2^b$ from x_1. Therefore, one must access at least $(d - m + 1)mb - f$ bits of the coefficients $p_0, \dots, p_{d-m}$ of p in order to approximate the root x_1 within $1/2^b$.

Now represent the same polynomial $p(x)$ as $(x-x_1)^{m-j} g_j(x)$ for $g_j(x) = (x - x_1)^j f(x)$ and $j = 1, \dots, m-1$. Apply Lemma 1 for m replaced by $m - j$ and for $f(x)$ replaced by $g_j(x)$ and deduce that a perturbation of the coefficient p_{d-m+j} of p by $|g_j|/2^{(m-j)b}$ turns the j-multiple root x_1 of $g_j(x) = (x - x_1)^j f(x)$ into j simple roots, all lying at the distance $1/2^b$ from x_1. Therefore, one must access at least $\sum_{j=1}^{m-1}((m-j)b - g = \frac{m-1}{2}mb - g$ bits of the coefficients $p_{d-m+1}, \dots, p_{d-1}$ in order to approximate the root x_1 within $1/2^b$ Sum the bounds $(d-m+1)mb - f$ and $\frac{m-1}{2}mb - g$ and arrive at the bound (3) on the overall number $\mathcal{B}_p$ bits; at least $\mathcal{B}_p/2$ Boolean operations must be used in order to access these bits – at least one operation per each pair of bits.

Let us specify bound (3) in two cases. (i) If $m = d$, $f(x) = 1$, and $|x_1| \le 0.5/d$, then $f = 0$, $|g_j| \le 2$ for all j, $g \le d-1$, and

$$\mathcal{B}_p \ge (d+1)db/2 - d + 1. \tag{4}$$

(ii) If x_1 is a simple root, well-isolated from the other roots of p,then substitute $m = 1$ and $g = 0$ into Eq. (3), thus turning it into $\mathcal{B}_p = db - f$ for $f(x) = p(x)/(x - x_1)$ such that $|f| \le d|p|$. This implies that

$$\mathcal{B}_p \ge (b - |p|)d.$$

Remark 3. Corollary 1 defines lower bounds on the Boolean complexity of Problems 1, 3, and 4 as long as an input polynomial p has an m-multiple root in the complex plain, a disc, and a segment, respectively. One can extend all these bounds to the case where a polynomial has an ϵ-cluster of m roots for a sufficiently small positive ϵ rather than an m-multiple root.

The algorithm of [32] and [37] solves Problem 2 by using $\tilde{O}(db')$ bits and Boolean operations.[4] This Boolean cost bound is within a poly-logarithmic factor from the information-theoretic lower bound db' of Proposition 1. Based on [52, Theorem 2.7] one can extend this estimate to the solution of Problems 1, 3 and 4 at a Boolean cost in $\tilde{O}(d^2 b)$, which is also nearly optimal by virtue of (4), and to nearly optimal solution of the problem of polynomial root isolation (see [42, Corollaries D.1 and D.2]).

4 Ehrlich's Iterations and Deflation

4.1 Ehrlich's Iterations and Their Super-Linear Convergence

The papers [5] and [10] present two distinct versions of MPSolve based on two distinct implementations of Ehrlich's functional iterations.

[5] applies original Ehrlich's iterations by updating current approximations z_i to all or selected roots x_i as follows:

$$z_i \leftarrow z_i - E_{p,i}(z_i),\ i = 1, \dots, d, \tag{5}$$

$$E_{p,i}(x) = 0 \text{ if } p(x) = 0;\ \frac{1}{E_{p,i}(x)} = \frac{1}{N_p(x)} - \sum_{j=1, j\ne i}^{d} \frac{1}{x - z_j} \text{ otherwise,} \tag{6}$$

$$N_p(x) = p(x)/p'(x). \tag{7}$$

[4] Here and hereafter we write $\tilde{O}(s)$ for $O(s)$ defined up to a poly-logarithmic factor in s.

[10] modifies these iterations by replacing polynomial equation $p(x) = 0$ by an equivalent rational *secular equation*[5]

$$S(x) := \sum_{j=1}^{d} \frac{v_j}{x - z_j} - 1 = 0 \tag{8}$$

where $z_j \approx x_j$ and $v_j = \frac{p(z_j)l(x)}{x-z_j}$ for $l(x) = \prod_{j=1}^{d}(x - z_j)$ and $j = 1, \dots, d$.

Cubic convergence of these iterations simultaneously to all roots of a polynomial has been proved locally, near the roots, but under some standard choices of initial approximations very fast global convergence to all roots, right from the start, has been consistently observed in all decades–long applications of the iterations worldwide.

4.2 Precision Management

The condition number of a root defines computational precision sufficient in order to ensure approximation within a fixed relative output error bound (see the relevant estimates in [5] and [10]). The value of the condition is not known a priori, however, and MPSolve adopts the following policy: at first apply Ehrlich's iterations with a fixed low precision (e.g., the IEEE double precision of 53 bits) and then recursively double it until all roots are approximated within a selected error tolerance. More precisely, MPSolve updates approximations only until they are close enough in order to satisfy a fixed stopping criterion, verified at a low computational cost. We call such roots *tame* and the remaining roots *wild*. MPSolve stops applying Ehrlich's iterations to a root when it is tamed but keep applying them to the wilde roots – with recursive doubling of the working precision. When a root is tamed this precision is optimal up to at most a factor of two. Recall from [50] and [52, Section 2.7] that working precision does not need to exceed the output precision b by more than a factor of d, and so at most $O(\log(db))$ steps of doubling precision are sufficient. This natural policy has been proposed and elaborated upon in [5] and [10], greatly improving the efficiency of the previous implementations of functional iterations for polynomial root-finding.

4.3 Ehrlich's Iterations with Deflation

Suppose that MPsolve seeks w wild roots $x_1, \dots, x_w$ of p and perform ITER_p Ehrlich's iterations with a working precision $\bar{b}$.This involves $O(dw\ \mathrm{ITER}_p)$ arithmetic operations performed at the Boolean cost $O(dw\mu(\bar{b})\ \mathrm{ITER}_p)$.

We propose a modification where we first (a) *deflate* p by computing its factor $f(x) = \prod_{j=1}^{w}(x - x_j)$ and then (b) apply to this factor ITER_f Ehrlich's iterations, at both stages (a) and (b) using the same working precision $\bar{b}$.

[5] The paper [10] elaborates upon expression of Ehrlich's iterations via secular equation, shows significant numerical benefits of root-finding by using this expression, and traces the previous study of this approach back to [6].

At stage (b) the Boolean cost bound $O(dw\mu(\bar{b}))$ decreases by a factor of d/w at the price of adding the cost of deflation at stage (a).[6]

Each of the three algorithms of the next subsection performs deflation at a Boolean cost in $O(dw\mu(\bar{b}))$, favorably compared to the above cost bound $O(dw\mu(\bar{b})\ \mathrm{ITER}_p)$. This comparison suggests that using deflation is competitive in terms of the Boolean cost and becomes more favorable as ITER_p grows.

Subsequent Ehrlich's iterations may in turn tame part of the roots of the polynomial $f(x)$, and then we can deflate $f(x)$. We can do this recursively, computing a sequence of factors $f_i(x)$ for $i = 1, 2, \dots, t$ where $f_1 = f$ and f_{i+1} is a smaller degree factor of f_i for all i.[7] The algorithm would stay nearly optimal overall if we perform deflation poly-logarithmic number of times t, e.g., if we delay deflation of the current factor until $\deg(f_{i+1}) \le \beta \deg(f_i)$ for a fixed constant[8] $\beta < 1$ and for all i. In the following example computational cost becomes too high if we perform deflation $d-1$ times but stays nearly optimal if we properly delay deflation.

Example 1. Let $p = \prod_{j=1}^{d}(x - 1 + 1/2^j)$ for a large integer d. In this case the roots $1 - 1/2^j$ are stronger isolated and better conditioned for smaller j, and so Ehrlich's iterations may peel out one such root of p at a time. Then we would $d-1$ times approximate polynomials of the form $p_i := f_i(x)\prod_{j=1}^{i}(x-1+1/2^j)$ for some polynomials $f_i(x)$, $i = 1, \dots, d-1$. For each i the polynomial $f_i(x)$ is a factor of p of degree $d-i$ sharing a cluster of at least $d-i$ roots with p, and so approximation of such a factor involves at least $b'_i d/2$ bits and at least $b'_i d/4$ Boolean operations. Now consider just the polynomials f_i for $i = 1, \dots, \lfloor d/2 \rfloor$. Each of them shares a cluster of at least $d/2$ roots with the polynomial p, and so we must choose $b'_i \ge bd/2$ (see Corollary 1). Hence approximation of all these factors requires at least order of bd^3 bits and Boolean operations, but we can decrease this large bound to nearly optimal level if we skip deflation at the ith step unless $\deg(f_i(x))/\deg(f_{i+1}(x)) \ge \gamma$ for a fixed $\gamma > 1$, e.g., $\gamma = 2$.

4.4 Deflation Algorithms for Ehrlich's Iterations

Recipe 1. Fix $\rho \ge 2\max_{j=1}^{w}|x_j|$ and an integer q such that $2^{q-1} \le w < 2^q$, write $z_j = \rho\exp(2\pi\mathbf{i}/2^q)$ for $j = 0, 1, \dots, 2^q - 1$, and compute (i) $p(z_j)$ for all j, (ii) $f(z_j) = p(z_j) - \prod_{g=w+1}^{d}(z_j - x_g)$ for all j, and (iii) the coefficients of $f(x)$.

[6] Furthermore we may have $\mathrm{ITER}_f < \mathrm{ITER}_p$ because of the decrease of the maximal distance between a pair of roots and of the number and sizes of root clusters in the transition from p to the polynomial $f(x)$ of a smaller degree w.

[7] [50] supplies estimates for the working precision in such a recursive process, which ensure the bound $1/2^b$ on the errors of the output approximations to the roots of p.

[8] The wild roots are much less numerous than the tame roots in a typical partition of a root set observed in Ehrlich's, Weierstrass's and other functional iterations that simultaneously approximate all roots of p as well as in Newton's iteration in [54]. Consequently the coefficient growth and the loss of sparseness are not dramatic in the transition to the factors defined by the wild roots.

Besides scaling the variable x, we perform $(d-w)2^q$ arithmetic operations at stage (ii), $\lceil d/w \rceil$ FFTs on 2^q points at stage (i), and a single inverse FFT on 2^q points; overall we need $O(dw\mu(\bar{b}))$ Boolean operations [26,46]. This cost bound can be verified for the following two recipes as well.

Recipe 2. Compute at first the values of the polynomial $f(x)$ at scaled roots of unity (as in Recipe 1), then the power sums of its roots, and finally its coefficients, (cf. [50, Section 12] or [42, Appendices A and B]).

Recipe 3 [53]**.** Compute the power sums $s_k = \sum_{j=1}^{d} x_j^k$, $k = 0, 1, \ldots$, of the roots of p by applying Newton's identities (cf., e.g., [36, Equations (2.5.4) and (2.5.5)]). Then by subtracting the powers of all tame roots compute the power sums of the roots of the polynomial $f(x)$. Finally recover its coefficients by applying Newton's identities.

Recipe 3 involves the coefficients of p, while Recipes 1 and 2 as well as Ehrlich's, Weierstrass's and Newton's iterations can be applied to a polynomial p given just by a subroutine for its evaluation, which is an advantage when, say, the polynomial is presented in a compressed form or in Bernstein basis.

4.5 Extension to Other Functional Iterations

The recipes of doubling the working precision and consequently of partitioning the roots into tame and wild ones and our recipes for deflation and its analysis can be extended to Weierstrass's [57], Werner's [60], various other functional iterations for simultaneous approximation of all roots of p [28, Chap. 4], and Newton's iterations applied to the approximation of all roots of p. E.g., Schleicher and Stoll in [54] apply Newton's iterations to the approximation of all roots of a polynomial of degree $d = 2^{17}$ and arrive at $w \approx d/1000$ wild roots.

4.6 Boolean Complexity of Problems 1–4 with and Without MPSolve

Let us compare Boolean complexity of the solution of Problems 1–4 by using MPSolve versus the algorithms of [11,32,37], and [12].

Empirically Ehrlich's iterations in MPSolve have simpler structure and smaller overhead in comparison with the algorithms of the latter papers, but unlike them have only empirical support for being nearly optimal. Moreover super-linear convergence of Ehrlich's iterations has only been observed for simultaneous approximation of all roots, and so these iterations solve Problems 3 and 4 of root-finding in a disc and on a line interval about as fast and as slow as Problem 1 of root-finding on the complex plain, while the nearly optimal cost of root-finding by the algorithms of [11,32,37], and [12] decreases at least proportionally to the number of roots in the input domain. As we have already said in the introduction, MPSolve, [11] and [12] can solve Problem 2 of factorization of p within the same Boolean cost bound as Problem 1, whereas the algorithm of [32] and [37] solves Problem 2 faster by a factor of d, reaching a nearly optimal Boolean cost bound.

5 Approximation of a Factor of a Polynomial

In Sect. 4.4 we approximated a factor $f(x)$ by first computing its values on a fixed circle or the power sums of its roots. These computations were inexpensive in applications to Ehrlich's iterations and were readily combined with them. In applications to subdivision iterations and to root-finding in the line interval the power sum techniques of the DLS deflation also work but require more elaboration. We begin that elaboration in this section, complete it in [42, Appendices A and B], and apply it in Sects. 6 and 7.

5.1 Isolation of a Domain and Its Boundary

The following definition covers quite a general class of domains on the complex plain, although in this paper we only apply it to the unit disc $D(0,1)$.

Definition 1. Isolation of a domain and its boundary. *Let a domain D on the complex plain allow its dilation from a fixed center. Then this domain has an* isolation ratio *at least θ and is θ-*isolated *for a polynomial p and real $\theta > 1$ if the root set of p in the domain D is invariant in the θ-dilation of D. The boundary of such a domain D has an* isolation ratio *at least θ and is θ-isolated if the root set of p in D stays invariant in both θ- and $1/\theta$-dilation of the domain.*

5.2 Approximation of a Factor with Root Set in an Isolated Disc

Problem 5: *Approximation of a factor with the root set in an isolated disc.*

INPUT: a polynomial p of (1), a complex number c, a positive number r, and $\theta = 1 + g/\log^h(d)$, a positive constant g and a real constant h such that the disc $D(c,r) = \{z : \ |z-c| < r\}$ on the complex plain is θ-isolated.

OUTPUT: the number w of roots of p in the disc $D(c,r)$ and a monic factor f of p whose root set is precisely the set of the roots of p that lie in that disc.

Dual **Problem 5a** of the approximation of a factor p/f whose root set lies outside an isolated disc D is equivalent to Problem 5 of the approximation of the factor of the reverse polynomial $p_{\rm rev}(x) := x^d p(1/x) = \sum_{i=0}^{d} p_{d-i}x^i$ with root set in D. [Notice that $p_{\rm rev}(x) = p_0 \prod_{j=1}^{d}(x - 1/x_j)$ if $p_0 \neq 0$.] Accordingly we can re-use the algorithms for Problem 5 in order to solve Problem 5a.

If we are given a factor f of p, we can also solve Problem 5a by means of any of the algorithms of [4,43,44,51, Section 3], and [26] for approximate polynomial division.

Alternatively (cf. Recipe 3 in Section 4.4) we can first compute (i) the sums of the ith powers of the roots of p and f for $i = 0, 1, \dots, d-w$, by applying Newton's identities or the algorithm of [50, Section 12], then (ii) the sums of the ith powers of the roots of the polynomial p/f for $i = 0, 1, \dots, d-w$, by subtracting the $d-w+1$ power sums of the roots of f from those of p, and finally (iii) the coefficients of p/f, by applying either Newton's identities or the algorithm [50, Section 12].

5.3 DLS Deflation: Outline and Complexity

Here is a very brief outline of the DLS algorithm, elaborated upon in [42, Appendices A and B] or [50, Section 12]. Given a polynomial p of degree d the DLS algorithm approximates its factor having root set in a unit disc θ-isolated from the other roots of p. The algorithm first computes $2q$ values of the factor at the $2q$th roots of unity, ensuring error bound roughly d/θ^q, then the power sums of its roots, and finally its coefficients.

The algorithm solves Problem 5 within the same asymptotic Boolean cost bound, $O(d\log(d)\mu(\bar{b}'))$, as in the special case of Sect. 4.4. [42, Corollary A.2] enables extension of the solution and its complexity estimates to the approximation of a factor of p with a root set on the θ-isolated unit circle $C(0,1)$ for $\theta = 1 + g/\log^h(d)$, a positive constant g and a real constant h. At the Boolean cost bound in $O(d\log(d)\mu(\bar{b}'))$ this reduces Problem 3 for a polynomial p of degree d to Problem 1 for a polynomial f of a degree $d_f < d$.

5.4 Deflation in the Case of an Isolated Cluster of a Small Number of Roots

Suppose that all d roots of p lie in the unit disc $D(0,1)$ and that in a subdivision iteration we observe that w roots of p form a cluster strongly isolated from the other roots of p. Let a small disc $D(c,r)$ cover the cluster,1 let $q(x) = p(x-c) = \sum_{j=0}^{d} q_j x^j$, let

$$\tilde{f}(x) = \sum_{j=0}^{w} q_j x^j = q(x) \mod x^{w+1} \tag{9}$$

be the sum of the $w+1$ trailing terms of the polynomial $q(x)$, and consider this sum an approximation of the factor $f(x)$ of $q(x)$ whose root set is made up of the roots of p lying in the cluster, as this was proposed in [27, Section 3.2] for real root-finding. Such disc $D(c,r)$ can be found as a cost-free by-product of subdivision iterations; then we obtain $\tilde{f}(x)$ by means of shifting the variable x.

Let $w \ll d$, that is, let the cluster size w be small. Let is be also well-isolated and estimate the norm $|\tilde{f} - f|$. Write $f := \sum_{i=0}^{w} f_i x^i$, $g := q/f = \sum_{i=0}^{d-w} g_i x^i$; let $f_w = g_0 = 1$.Then

$$\tilde{f}(x) - f(x) = f(x) \sum_{j=1}^{w} g_j x^j \mod x^{w+1}. \tag{10}$$

Clearly $\tilde{f}(x) = f(x)$ if $c = 0$ and $f(x) = x^w$. Consider the special case where all roots of f lie in a small r-neighborhood of 0 such that $rw \le 1/2$. Then we readily verify that $|f(x) - x^w| \le |(x-r)^w - x^w| \le 2rw$ and $|\tilde{f}(x) - x^w| \le 2(d-w)^w rw$, and so $|\tilde{f} - f| \le 2((d-w)^w + 1)rw$. This bound can be sufficient in some applications where w is a small positive integer, although the bound is by far not as strong as what we can obtain by applying the algorithms of [42, Part I of the Appendix].

A root-squaring iteration [21] lifts the isolation of the cluster or equivalently squares the radius of the disc. We can apply such lifting recursively, although limited number of times, within $O(\log(\log(d)))$, because of numerical problems. Having approximated the lifted roots in the cluster, we can recover the associated roots of p by applying the descending process of [32] and [37], at dominated overall Boolean cost at both lifting and descending stages.

6 Subdivision Iterations with Deflation

1. Background on Subdivision. Subdivision iterations extend the classical bisection iterations from root-finding on a line to polynomial root-finding in the complex plain. Under the name of *Quad-tree Construction* these iterative algorithms have been studied in [22,23,49], and [35] and extensively used in Computational Geometry. The algorithms have been introduced by Herman Weyl in [58] and advanced in [22,23,49], and [35]; under the name of subdivision Becker et al. modified them in [11] and [12].[9] Let us briefly recall subdivision algorithms for Problem 1; they are similar for Problem 3.

At the beginning of a subdivision (quad-tree) iteration all the d roots of p are covered by at most $4d$ congruent *suspect squares* on the complex plain that have horizontal and vertical edges, all of the same length. The iteration outputs a similar cover of all d roots of p with a new set of at most $4d$ suspect squares whose edge length is halved. Hence the centers of the suspect squares approximate the root set with an error bound linearly converging to 0.

At every iteration suspect squares form connected components. Given $s > 1$ components, embed them into the minimal discs $D_i = D(c_i, R_i)$, $i = 1, \dots, s$; they become well-isolated from each other in $O(\log(d))$ iterations. For $i = 1, \dots, s$ cover the root sets of p in the discs D_i by the minimal discs $D'_i = D(c'_i, R'_i)$. For some i the distances $|c_i - c'_i|$ may greatly exceed R'_i, and then linear convergence of subdivision iterations to roots lying in the disc D'_i can be too slow in order to support root-finding in nearly optimal Boolean time.

The algorithms of [11,35,49] and [12] yield super-linear convergence at those stages. [49] and [35] apply Newton's iterations, whose convergence to a disc D'_i is quadratic right from the start if they begin in its θ-dilation $D(c'_i, \theta R'_i)$ and if the disc D'_i is θ-isolated for a sufficiently large θ. Tilli in [55] proves that it is sufficient if $\theta \geq 3d - 3$, which improves the earlier estimate $\theta \geq 5d^2$ of [49]. [11] and [12] achieve super-linear convergence to the roots by applying Pellet's theorem for root-counting in a disc. This was the main algorithmic innovation of [11] and [12] versus [49] and [35]; the papers [11] and [12] have also extended to the complex plain the QIR iterations, proposed by Abbott for a line segment, and then laboriously estimated the Boolean cost of the resulting algorithm.

2. Our Alternatives: Outline. We deviate from the algorithms of [11] and [12] in two ways.

[9] The algorithms of [11] and [12] are quite similar to one another.

(i) We replace the counting algorithm of [11] and [12] with a distinct method, which is faster, more robust, and can be applied where a polynomial p is defined by a black box subroutine for its evaluation rather than by its coefficients.

(ii) If the algorithm of [11] and [12] encounters an isolated component containing d_f roots of p with sufficiently small d_f (we either monitor this explicitly or detect by action), we cover that component with an isolated disc containing these d_f roots and then approximate them by solving Problem 3.

3. Our Alternative Counting. We fix a sufficiently large integer q, let ω denote a primitive qth root of unity, and approximate the number s of the roots of p in the θ-isolated unit disc $D(0,1) = \{x : \ |x| = 1\}$ as follows:

$$s \approx s^* = \frac{1}{q} \sum_{g=0}^{q-1} \omega^g \ \frac{p'(\omega^g)}{p(\omega^g)}, \ \omega = \exp(2\pi\sqrt{-1}/q). \tag{11}$$

By virtue of [42, Theorem 12], extracted from [50], $|s-s^*| < 1/2$ if $2d+1 < \theta^q$. For example, if $\theta = 2$, then choosing any $q \geq 11$ is sufficient where $d = 1,000$ and choosing any $q \geq 21$ is sufficient where $d = 1,000,000$.

We obtain s^* of (11) by evaluating both $p(x)$ and $p'(x)$ at the qth roots of unity (which means performing discrete Fourier transform at q points twice) and in addition performing discrete Fourier transform at q points once again. We can perform Fourier transforms by applying FFT if we choose q being the power of 2.

We can extend our recipe to any sufficiently well isolated disc D on complex plain by means of shifting and scaling the variable x (see Remark 2), but alternatively we can just evaluate $p(x)$ and $p'(x)$ at q equally spaced points on the boundary circle of the disc D. Instead of two FFTs we can apply the so called Horner's algorithm $2q$ times or the algorithms of [29,38], or [41] for fast multipoint polynomial evaluation.

4. Root-Finding in an Isolated dDisc (Problem 3). As soon as subdivision defines a well-isolated disc D_i' containing a positive but reasonably small number d_f of the roots of a polynomial p we approximate its factor $f = f(x)$ having degree d_f and having all its roots in that disc (cf. Sect. 4.3); then we approximate all the d_f roots of the factor by applying the subdivision iterations or MPSolve.

By applying a subdivision algorithm to f rather than p, we approximate the roots of f at the cost that decreases by at least the factor of d/w, because of the decrease of the degree, but possibly more than that if we get rid of some root clusters in the transition from p to f.

Empirically we may additionally benefit from shifting to MPSolve rather than continuing subdivision iterations unless our simplification of subdivision makes it competitive with or even superior to MPSolve.

By applying the DLS algorithm we approximate the factor f within the same asymptotic estimates for the Boolean cost as we deduced for deflation algorithms of Sect. 4.3.[10]

Then again we can apply deflation of p repeatedly for a number of discs D_i' and can recursively extend it to deflation of the computed factors $f(x)$, together with the policy of delaying the deflation until we decrease the number of remaining roots below a fixed bound.[11] The estimates of [42, Appendices C.2 and D] for the working precision in this deflation ensure the upper bound of $1/2^b$ on the errors of the output approximation to the roots.

7 Root-Finding on a Line Segment with Deflation

The algorithms of [11,27,32,37,45] and [12] solve Problem 4 in nearly optimal Boolean time. Next we reduce Problem 4 on the unit segment $S[-1,1] = \{x : \ -1 \le x \le 1\}$ to the special case where all roots of p lie in that segment. In this case the algorithms of [8,13], and [14] are also nearly optimal, but application of MPSolve or [27], complemented with the policy of recursive doubling of working precision, may have even better chances to become the method of user's choice, particularly if its efficiency is enhanced by incorporating the initial approximation of the real roots by means of the simple algorithm of [47], which we recall at the end of this section.

Next we approximate the factor of $p(x)$ whose root set is precisely the set of the roots of p restricted to the segment $S[-1,1]$. The algorithm of [42, Appendices A and B] can be applied for the deflation over any convex domain of the complex plain (see [42, Remark 15]), but in the case of a disc its output approximation to the factor is much closer than for general convex domain (at the same computational cost). Thus we obtain better output approximation by means of reducing Problem 4 to Problem 3.

Towards this goal we first recall the two-to-one Zhukovsky function $z = J(x)$, which maps the unit circle $C(0,1)$ onto the unit segment $S[-1,1]$, and its one-to-two inverse:

$$z = J(x) := \frac{1}{2}\Big(x + \frac{1}{x}\Big); \ x = J^{-1}(z) := z \pm \sqrt{z^2 - 1}. \tag{12}$$

Here x and z are complex variables. Now perform the following steps:

1. Compute the polynomial $s(z) := x^d p(x) p(1/x)$ of degree $2d$ by applying [9, Algorithm 2.1], based on the evaluation of the polynomials $p(x)$ and $x^d p(1/x)$

[10] The DLS algorithms (cf. [42, Appendices A and B]) approximates a factor f at a nearly optimal Boolean cost if the disc D_i' is θ-isolated for isolation ratio $\theta = 1 + g/\log^h(d)$, a positive constant g and a real constant h. Such an isolation ratio is smaller than those required in [11,35,49] and [12] and thus can be ensured by means of performing fewer subdivision steps.

[11] Then again with such a delay we bound the overall cost of all deflation steps (cf. Example 1), avoid coefficient growth and do not lose sparseness.

at Chebyshev points and the interpolation to $s(z)$ at roots of unity. Recall that the set of the roots of $p(x)$ lying in the segment $S[-1,1]$ is well-isolated and observe (see Remark 5) that it is mapped in one-to-two mapping (12) into a well-isolated set of the roots of $s(z)$ lying on the unit circle $C(0,1)$.

2. Let $g(z)$ denote the monic factor of the polynomial $s(z)$ with the root set made up of the roots of $s(z)$ lying on the unit circle $C(0,1)$ and such that $\deg(g(z)) = \deg(f)$. By applying the algorithm of [50, Section 12] (cf. [42, Corollary A.2]) approximate the power sums of the roots of the polynomial $g(z)$.
3. By applying Newton's identities of the algorithm of [50, Section 13] (cf. [42, Solution 2 of Appendix B]) approximate the coefficients of $g(z)$.
4. Compute the polynomial $h(x) := x^{2a} g(\frac{1}{2}(x + \frac{1}{x}))$ of degree $4\deg(f)$ in x. Its root set is made up of the roots of the polynomial p lying in the segment $S[-1,1]$ and of their reciprocals; in the transition to $h(x)$ the multiplicity of the roots of p either grows 4-fold (for the roots 1 and -1 if they are the roots of p) or is doubled, for all other roots.
5. By applying the algorithm of [14] approximate all roots of the polynomial $h(x)$.
6. Among them identify and output $\deg(f)$ roots that lie in the segment $S[-1,1]$; they are precisely the roots of $p(x)$.

Remark 4. We can simplify stage 5 by replacing the polynomial $h(x)$ with its half-degree square root $j(x) := x^a f(x) f(1/x)$ at stage 5, but further study is needed to find out whether and how much this could decrease the overall computational cost.

Remark 5. Represent complex numbers as $z := u + \mathbf{i}v$. Then Zhukovsky's map transforms a circle $C(0,\rho)$ for $\rho \neq 1$ into the ellipse $E(0,\rho)$ whose points (u,v) satisfy the following equation,

$$\frac{u^2}{s^2} + \frac{v^2}{t^2} = 1 \text{ for } s = \frac{1}{2}\Big(\rho + \frac{1}{\rho}\Big), \quad t = \frac{1}{2}\Big(\rho - \frac{1}{\rho}\Big).$$

Consequently it transforms the annulus $A(0, 1/\theta, \theta)$ into the domain bounded by the ellipses $E(0,1/\theta)$ and $E(0,\theta)$, so the circle $C(0,1)$ is θ-isolated if and only if no roots of p lie in the latter domain.

We conclude this section with recalling an efficient algorithm of [47] for computing crude initial approximations to real roots. Extensive tests in [47] showed particular efficiency of this algorithm for the approximation of the real roots of p that are sufficiently well-isolated from the other roots.

Theorem 1. *See [50, Corollary 14.3]. Assume that we are given a polynomial $p = p(x)$ of (1) and a pair of real constants $c > 0$ and h. Write $\theta = 1 + c/d^h$ and $r_j := |x_j|$ for $j = 1, \dots, d$. (r_j are said to be the root radii for p.) Then, within the Boolean cost bound $O_B(d^2 \log^2(d))$, one can compute approximations $\tilde{r}_j$ to all root radii r_j such that $1/\theta \le \tilde{r}_j / r_j \le \theta$ for $j = 1, \dots, d$, provided that $\lg(\frac{1}{\theta - 1}) = O(\lg(d))$, that is, $|\tilde{r}_j / r_j - 1| \le c/d^h$.*

Apply the algorithm supporting this theorem and compute d narrow annuli covering all roots of p. Their intersection with real line defines at most $2d$ small segments that contain all real roots of p. Then we weed out the extraneous empty segments containing no roots of p and obtain close approximations to all real roots, in particular to those lying in the segment $S[-1, 1]$. See [47] for further details.

Acknowledgements. This research has been supported by NSF Grants CCF–1563942 and CCF–1733834 and PSC CUNY Award 69813 00 48.

References

1. Bell, E.T.: The Development of Mathematics. McGraw-Hill, New York (1940)
2. Boyer, C.A.: A History of Mathematics. Wiley, New York (1968)
3. Barnett, S.: Polynomial and Linear Control Systems. Marcel Dekker, New York (1983)
4. Bini, D.A.: Parallel solution of certain Toeplitz linear systems. SIAM J. Comput. **13**(2), 268–279 (1984)
5. Bini, D.A., Fiorentino, G.: Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder. Numer. Algorithms **23**, 127–173 (2000)
6. Bini, D.A., Gemignani, L., Pan, V.Y.: Inverse power and Durand/Kerner iteration for univariate polynomial root-finding. Comput. Math. Appl. **47**(2/3), 447–459 (2004)
7. Box, G.E.P., Jenkins, G.M.: Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco (1976)
8. Bini, D., Pan, V.Y.: Computing matrix eigenvalues and polynomial zeros where the output is real. SIAM J. Comput. 27(4), 1099–1115 (1998). Proc. version. In: SODA 1991, pp. 384–393. ACM Press, NY, and SIAM Publ., Philadelphia (1991)
9. Bini, D., Pan, V.Y.: Graeffe's, Chebyshev, and Cardinal's processes for splitting a polynomial into factors. J. Complex. **12**, 492–511 (1996)
10. Bini, D.A., Robol, L.: Solving secular and polynomial equations: a multiprecision algorithm. J. Comput. Appl. Math. **272**, 276–292 (2014)
11. Becker, R., Sagraloff, M., Sharma, V., Xu, J., Yap, C.: Complexity analysis of root clustering for a complex polynomial. In: International Symposium on Symbolic and Algebraic Computation (ISSAC 2016), pp. 71–78. ACM Press, New York (2016)
12. Becker, R., Sagraloff, M., Sharma, V., Yap, C.: A near-optimal subdivision algorithm for complex root isolation based on the Pellet test and Newton iteration. J. Symb. Comput. **86**, 51–96 (2018)
13. Ben-Or, M., Tiwari, P.: Simple algorithms for approximating all roots of a polynomial with real roots. J. Complex. **6**(4), 417–442 (1990)
14. Du, Q., Jin, M., Li, T.Y., Zeng, Z.: The quasi-Laguerre iteration. Math. Comput. **66**(217), 345–361 (1997)
15. Delves, L.M., Lyness, J.N.: A numerical method for locating the zeros of an analytic function. Math. Comput. **21**, 543–560 (1967)
16. Demeure, C.J., Mullis, C.T.: The Euclid algorithm and the fast computation of cross-covariance and autocovariance sequences. IEEE Trans. Acoust. Speech Signal Process. **37**, 545–552 (1989)

17. Demeure, C.J., Mullis, C.T.: A Newton-Raphson method for moving-average spectral factorization using the Euclid algorithm. IEEE Trans. Acoust. Speech Signal Process. **38**, 1697–1709 (1990)
18. Ehrlich, L.W.: A modified Newton method for polynomials. Commun. ACM **10**, 107–108 (1967)
19. Emiris, I.Z., Pan, V.Y., Tsigaridas, E.: Algebraic algorithms. In: Tucker, A.B., Gonzales, T., Diaz-Herrera, J.L. (eds.) Computing Handbook. Computer Science and Software Engineering, 3rd edn., vol. I, Chap. 10, pp. 10-1–10-40. Taylor and Francis Group (2014)
20. Fortune, S.: J. Symbol. Comput. **33**(5), 627–646 (2002). Proc. version in Proc. Intern. Symp. on Symbolic and Algebraic Computation An Iterated Eigenvalue Algorithm for Approximating Roots of Univariate Polynomials, (ISSAC 2001), 121–128, ACM Press, New York (2001)
21. Householder, A.S.: Dandelin, Lobachevskii, or Graeffe? Amer. Math. Mon. **66**, 464–466 (1959)
22. Henrici, P.: Applied and computational complex analysis. In: Power Series, Integration, Conformal Mapping, Location of Zeros, vol. 1. Wiley, New York (1974)
23. Henrici, P., Gargantini, I.: Uniformly convergent algorithms for the simultaneous approximation of all zeros of a polynomial. In: Dejon, B., Henrici, P. (eds.) Constructive Aspects of the Fundamental Theorem of Algebra. Wiley, New York (1969)
24. Imbach, R., Pan, V.Y., Yap, C.: Implementation of a near-optimal complex root clustering algorithm. In: Davenport, J.H., Kauers, M., Labahn, G., Urban, J. (eds.) ICMS 2018. LNCS, vol. 10931, pp. 235–244. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96418-8_28
25. Inbach, R., Pan, V.Y., Yap, C., Kotsireas, I.S., Zaderman, V.: Root-finding with implicit deflation. In: Proceedings CASC 2019. arxiv:1606.01396. Accepted 21 May 2019
26. Kirrinnis, P.: Polynomial factorization and partial fraction decomposition by simultaneous Newton's iteration. J. Complex. **14**, 378–444 (1998)
27. Kobel, A., Rouillier, F., Sagraloff, M.: Computing real roots of real polynomials ... and now for real! In: The International Symposium on Symbolic and Algebraic Computation (ISSAC 2016), pp. 301–310. ACM Press, New York (2016)
28. McNamee, J.M.: Numerical Methods for Roots of Polynomials, Part I, p. XIX+354. Elsevier, Amsterdam (2007)
29. Moenck, R., Borodin, A.: Fast modular transforms via division., In: Proceedings of 13th Annual Symposium on Switching and Automata Theory (SWAT 1972), pp. 90–96. IEEE Computer Society Press (1972)
30. McNamee, J.M., Pan, V.Y.: Numerical Methods for Roots of Polynomials, Part II, p. XXI+728. Elsevier, Amsterdam (2013)
31. Neff, C.A., Reif, J.H.: An $o(n^{1+\epsilon})$ algorithm for the complex root problem. In: Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS 1994), pp. 540–547. IEEE Computer Society Press (1994)
32. Pan, V.Y.: Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros. In: Proceedings of 27th Annual ACM Symposium on Theory of Computing (STOC 1995), pp. 741–750. ACM Press, New York (1995)
33. Pan, V.Y.: Solving a polynomial equation: some history and recent progress. SIAM Rev. **39**(2), 187–220 (1997)
34. Pan, V.Y.: Solving polynomials with computers. Am. Sci. **86**, 62–69 (1998)

35. Pan, V.Y.: Approximation of complex polynomial zeros: modified quadtree (Weyl's) construction and improved Newton's iteration. J. Complex. **16**(1), 213–264 (2000)
36. Pan, V.Y.: Structured Matrices and Polynomials: Unified Superfast Algorithms. Birkhäuser/Springer, Boston/New York (2001). https://doi.org/10.1007/978-1-4612-0129-8
37. Pan, V.Y.: Univariate polynomials: nearly optimal algorithms for factorization and rootfinding. J. Symb. Comput. **33**(5), 701–733 (2002)
38. Pan, V.Y.: Transformations of matrix structures work again. Linear Algebra Appl. **465**, 1–32 (2015)
39. Pan, V.Y.: Root-finding with Implicit Deflation. arXiv:1606.01396, Accepted 4 June 2016
40. Pan, V.Y.: Simple and nearly optimal polynomial root-finding by means of root radii approximation. In: Kotsireas, I., Martinez-Moro, E. (eds.) ACA 2015. SPMS, vol. 198, pp. 329–340. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56932-1_23
41. Pan, V.Y.: Fast approximate computations with Cauchy matrices and polynomials. Math. Comput. **86**, 2799–2826 (2017)
42. Pan, V.Y.: Old and new nearly optimal polynomial root-finders, In: Proceedings of CASC (2019). Also arxiv: 1805.12042 May 2019
43. Pan, V.Y., Sadikou, A., Landowne, E.: Univariate polynomial division with a remainder by means of evaluation and interpolation. In: Proceedings of 3rd IEEE Symposium on Parallel and Distributed Processing, pp. 212–217. IEEE Computer Society Press, Los Alamitos (1991)
44. Pan, V.Y., Sadikou, A., Landowne, E.: Polynomial division with a remainder by means of evaluation and interpolation. Inform. Process. Lett. **44**, 149–153 (1992)
45. Pan, V.Y., Tsigaridas, E.P.: Nearly optimal refinement of real roots of a univariate polynomial. J. Symb. Comput. **74**, 181–204 (2016). Proceedings version. In: Kauers, M. (ed.) Proc. ISSAC 2013, pp. 299–306. ACM Press, New York (2013)
46. Pan, V.Y., Tsigaridas, E.P.: Nearly optimal computations with structured matrices. Theor. Comput. Sci. **681**, 117–137 (2017)
47. Pan, V.Y., Zhao, L.: Real root isolation by means of root radii approximation. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) CASC 2015. LNCS, vol. 9301, pp. 349–360. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-319-24021-3_26
48. Pan, V.Y., Zhao, L.: Real polynomial root-finding by means of matrix and polynomial iterations. Theor. Comput. Sci. **681**, 101–116 (2017)
49. Renegar, J.: On the worst-case arithmetic complexity of approximating zeros of polynomials. J. Complex. **3**(2), 90–113 (1987)
50. Schönhage, A.: The Fundamental Theorem of Algebra in Terms of Computational Complexity. Department of Mathematics. University of Tübingen, Tübingen, Germany (1982)
51. Schönhage, A.: Asymptotically fast algorithms for the numerical muitiplication and division of polynomials with complex coefficients. In: Calmet, J. (ed.) EUROCAM 1982. LNCS, vol. 144, pp. 3–15. Springer, Heidelberg (1982). https://doi.org/10.1007/3-540-11607-9_1
52. Schönhage, A.: Quasi GCD computations. J. Complex. **1**, 118–137 (1985)
53. Schleicher, D.: Private communication (2018)
54. Schleicher, D., Stoll, R.: Newton's method in practice: finding all roots of polynomials of degree one million efficiently. Theor. Comput. Sci. **681**, 146–166 (2017)

55. Tilli, P.: Convergence conditions of some methods for the simultaneous computation of polynomial zeros. Calcolo **35**, 3–15 (1998)
56. Van Dooren, P.: Some numerical challenges in control theory. In: Van Dooren, P., Wyman, B. (eds.) Linear Algebra for Control Theory. The IMA Volumes in Mathematics and its Applications, vol. 62. Springer, New York (1994). https://doi.org/10.1007/978-1-4613-8419-9_12
57. Weierstrass, K.: Neuer Beweis des Fundamentalsatzes der Algebra. Mathematische Werke, Bd, vol. III, pp. 251–269. Mayer und Mueller, Berlin (1903)
58. Weyl, H.: Randbemerkungen zu Hauptproblemen der Mathematik. II. Fundamentalsatz der Algebra und Grundlagen der Mathematik. Math. Z. **20**, 131–151 (1924)
59. Wilson, G.T.: Factorization of the covariance generating function of a pure moving-average. SIAM J. Numer. Anal. **6**, 1–7 (1969)
60. Werner, W.: Some improvements of classical iterative methods for the solution of nonlinear equations. In: Allgower, E.L., Glashoff, K., Peitgen, H.-O. (eds.) Numerical Solution of Nonlinear Equations. LNM, vol. 878, pp. 426–440. Springer, Heidelberg (1981). https://doi.org/10.1007/BFb0090691