Faster Numerical Univariate Polynomial Root-Finding by Means of Subdivision Iterations

Qi Luan[1], Victor Y. Pan[1,2(✉)], Wongeun Kim[3], and Vitaly Zaderman[1]

[1] Ph.D. Programs in Mathematics and Computer Science, The Graduate Center of the City University of New York (CUNY), New York, NY 10036, USA
qi_luan@yahoo.com, vza52@aol.com

[2] Department of Computer Science, Lehman College of CUNY, Bronx, NY 10468, USA
victor.pan@lehman.cuny.edu

[3] Department of Mathematics, The Lander College for Men, Touro College, Kew Garden, NY 11367, USA
won-geun.kim2@touro.edu
http://comet.lehman.cuny.edu/vpan/

Abstract. Root-finding for a univariate polynomial is four millennia old and still highly important for Computer Algebra and various other fields. Subdivision root-finders for a complex univariate polynomial are known to be highly efficient and practically promising. The recent one by Becker et al. [2] competes for user's choice and is nearly optimal for dense polynomials represented in monomial basis, but [18] proposes and analyzes further significant acceleration, which becomes dramatic for polynomials admitting their fast evaluation (e.g., sparse ones). Here and in the companion paper [19], we present some of these results and algorithms.

Keywords: Polynomial roots · Subdivision · Sparse polynomials · Real polynomial root-finding

1 Introduction

1.1 State of the Art

Root-finding for univariate polynomials has been the central subject of Mathematics and Computational Mathematics for four millennia since Sumerian times and until the middle of 19th century A.D and began its new life with the advent of computers. Presently this subject is highly important for Computer Algebra and many other computational areas. Since 2000, the root-finder of user's choice has been the package MPSolve (Multiprecision Polynomial Solver) [3,5], which implements Ehrlich's, aka Aberth's iterations, but recent progress in subdivision iterations has made them potentially competitive. Due to [23], advanced in [2,8,14,21], and known in Computational Geometry as Quad-tree Construction,

F. Boulier et al. (Eds.): CASC 2020, LNCS 12291, pp. 431–446, 2020.
https://doi.org/10.1007/978-3-030-60026-6_25

they extend bisection of a line segment to root-finding in the complex plane. Their advanced version of 2016–2018 by Becker et al. [2] is the second known nearly optimal root-finder[1] for a polynomial represented in monomial basis – by its coefficients:

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

The implementation in [11] has slightly outperformed MPSolve for root-finding in a region containing a small number of roots,[2] while the implementation in [12] is user's current choice for the highly important task of real polynomial root-finding, where the input and output are real.

The main and bottleneck block of the known subdivision iterations, including those of [2,14,21], is the application of an *exclusion test*, which either certifies or does not certify that a fixed disc on the complex plane contains no roots of an input polynomial p. This test is a special case of *root-counting* in the disc, which is another basic block of subdivision algorithms. According to [2] its main algorithmic novelty versus its predecessors is its root-counting by means of pairwise comparison of the absolute values of the coefficients of $p(x)$ and invoking Pellet's classical theorem.

1.2 Our Progress

A new significant acceleration in [18] relies on another novel approach to root-counting and exclusion tests. Unlike the case of [2] these blocks and the whole root-finder work for a *black box input polynomial* – defined by a black box for its evaluation at a given point. This class includes polynomials represented in Bernstein and Chebyshev bases – admitting numerically stable evaluation, as well as sparse, Mandelbrot's, and various other polynomials, which admit dramatically faster evaluation. [2] takes no advantage of this huge benefit, but [18] fully exploits it and thus dramatically accelerates [2] for the latter input class. In particular, Taylor's shift of the variable and Dandelin–Lobachevsky–Gräffe's recursive root-squaring [9], being two well-known drawbacks of the subdivision root-finder [2], are avoided in [18].

Work [18] also accelerates polynomial root-finding based on Ehrlich's, Newton's, and other functional iterations, as well as numerical multipoint polynomial evaluation, extensively involved in polynomial root-finding but also having independent importance. Because of the size limitation, however, we skip these subjects, omit many details, and leave to [18] formal support for our algorithms,

[1] The first such root-finder, of [16], is nearly optimal also for the task of numerical factorization of a polynomial into the product of its linear factors, having independent importance.

[2] Throughout the paper we count m times a root of multiplicity m and handle it as a cluster of m roots whose diameter is smaller than the tolerance to the output approximation errors.

including correctness proof and Boolean cost estimates. We occasionally estimate arithmetic complexity where we can control the precision of computing. Already an initial implementation of our algorithms in [10] demonstrates 3-fold acceleration of the previous best implementation of subdivision root-finding, even though we have not yet incorporated many promising directions for further progress specified in [18]. Our present paper and its companion [19] together cover only a fraction of the results of [18], focusing on new exclusion test and root-counting in subdivision iterations and extension to real root-finding.

1.3 Power Sums and Cauchy Sums

We adopt rather than counter the subdivision iterations of [2,14,21] but enrich them with performing their main two blocks by means of the approximation of the power sums of the roots of $p(x)$ that lie in a fixed disc on the complex plain: (i) the sum of their 0th powers is precisely the number of the roots in the disc, and (ii) such a disc contains no roots if and only if all the power sums vanish. We only approximate integers (0 or the number of roots), perform computations with a low precision, and use just order of $\log(d)$ arithmetic operations in an exclusion test and root-counting for a degree d input polynomial.

A technical point of our departure was the study of the power sums in the extensive advanced work on the Boolean complexity of polynomial root-finding by Schönhage in [22, Sects. 12 and 13]. He has approximated the power sums s_h of the roots lying in the unit disc $D(0,1) = \{x : |x| \le 1\}$ by means of *Cauchy sums* s_h^*, being discretizations of Cauchy's contour integral:[3]

$$s_h := \sum_{x_j \in D(c,\rho)} x_j^h = \int_{C(c,\rho)} \frac{p'(x)}{p(x)}\, x^h\, dx, \text{ for } h = 0, 1, \dots, \tag{2}$$

$$s_h^* := \frac{1}{q} \sum_{g=0}^{q-1} \zeta^{(h+1)g}\, \frac{p'(c+\rho\zeta^g)}{p(c+\rho\zeta^g)} \text{ for } h = 0, 1, \dots, q-1,\ \zeta := \exp\Big(\frac{2\pi\mathbf{i}}{q}\Big), \tag{3}$$

where ζ denotes a primitive qth root of unity, for a fixed $q > 1$.

1.4 Real Root-Finding

Real root-finding is highly important because in many applications, e.g., to geometric and algebraic-geometric optimization, only real roots of a polynomial are of interest and because they are typically much less numerous than all d complex roots. In particular, under a random coefficient model, a polynomial of degree d is expected to have $O(\log(d))$ real roots (cf. [6]).

Real roots of a polynomial defined numerically, with rounding errors, turn into nearly real roots, whose approximation has rarely if at all been addressed properly in the known algorithms, while we handle this issue by following [18].

[3] Schönhage was seeking a factor of $p(x)$ with root set made up of the roots of $p(x)$ lying in that disc; he only approximated the power sums s_h for positive h.

Namely [18] proposes, elaborates upon and analyzes efficient root-counting and deflation techniques for the roots of a polynomial lying on and near a circle on the complex plane, and in Sect. 4, we extend these techniques to the roots lying on and near a fixed segment of the real axis by applying Zhukovsky's function and its inverse. Given a black box polynomial $p(x)$, we deflate its factor f whose root set is precisely the root set of $p(x)$ lying on or near a fixed segment of the real axis. Since $\deg(f)$ tends to be much smaller than d, deflation of the factor f and its subsequent root-finding are performed at a low Boolean cost [18].

A preliminary version of this algorithm appeared in [17, Section 7], but presently we simplify it substantially.[4] We perform its stage 1 by means of evaluation and interpolation without involving more advanced algorithm of [4]. At its stage 3, we use more efficient [18, Algorithm 46] instead of [18, Algorithm 45], and we simplify root-finding stage 4 by first applying our new Algorithm 4, which decreases by twice the degree of the factor f of p and still keeps in its root set the images of all real roots of p.

Moreover we propose an alternative extension of our study of root-counting from the complex plain to real interval. This achieves less than deflation but at a much lower cost, and is still a major stage of real and nearly real root-finding. Our non-costly root-counter in and near a line segment provides more information than the customary ones – based on the Descartes rule of signs or Budan–Fourier theorem, involves no costly computation used in Sturm sequences and, unlike Budan–Fourier theorem, can be applied to black box polynomials. And as we said already, unlike the known real root-counters we output the overall number of roots lying in and near a fixed segment of the real axis.

[18, Section 6] and the paper [20] approximate pairwise well-isolated real roots fast by narrowing the range for their search.

1.5 Organization of the Paper

We recall some background material in the next section, cover Cauchy sum computation, root-counting and exclusion tests in Sect. 3, and devote Sect. 4 to real polynomial root-finding. The Boolean complexity of the new algorithms is estimated in [18] in some detail; we do not include this study because of size limitation.

2 Background

2.1 Definitions and Auxiliary Results

– $S(c,\rho)$, $D(c,\rho)$, $C(c,\rho)$, and $A(c,\rho_1,\rho_2)$ denote square, disc, circle (circumference), and annulus on the complex plain, respectively:

$$S(c,\rho) := \{x : |\Re(c-x)| \le \rho,\ |\Im(c-x)| \le \rho\},$$
$$D(c,\rho) := \{x : |x-c| \le \rho|\}, \tag{4}$$

[4] Otherwise [17] focuses on deflation, and only half-page [17, Section 6.3] overlaps with us.

$$C(c,\rho) := \{x : |x-c| = \rho|\},\ A(c,\rho_1,\rho_2) := \{x : \rho_1 \le |x-c| \le \rho_2|\}. \quad (5)$$

- An annulus $A(c,\rho_1,\rho_2)$ has *relative width* $\frac{\rho_2}{\rho_1}$.
- 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.
- $|u| = \sum_{i=0}^{d} |u_i|$ denotes the norm of a polynomial $u(x) = \sum_{i=0}^{d} u_i x^i$.
- IND($\mathcal{R}$), the index of a region $\mathcal{R}$ of the complex plain (e.g., a square, a disc, an annulus, or a circle), is the number of roots of p contained in it.
- A disc $D(c,\rho)$ and circle $C(c,\rho)$ have an *isolation ratio* θ or equivalently are *θ-isolated* for a polynomial p, real $\theta \ge 1$, and complex c if no roots of p lie in the open annulus $A(c,\rho/\theta,\rho\theta)$, of relative width θ^2, or equivalently if IND($D(c,\rho/\theta)$) =IND($D(c,\rho\theta)$). (See Fig. 1.)
  A disc and a circle are *well-isolated* if they are θ-isolated for $\theta - 1$ exceeding a positive constant.
- Define the reverse polynomial of $p(x)$:

$$p_{\rm rev}(x) := x^d p\Big(\frac{1}{x}\Big) = \sum_{i=0}^{d} p_i x^{d-i},\ p_{\rm rev}(x) = p_0 \prod_{j=1}^{d} \Big(x - \frac{1}{x_j}\Big) \text{ if } p_0 \ne 0. \quad (6)$$

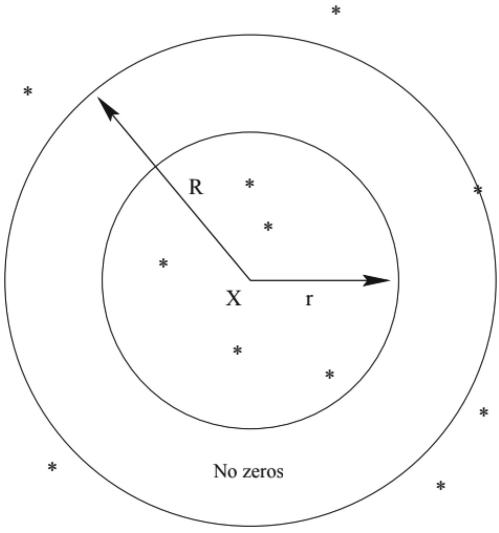

Fig. 1. The internal disc $D(X,r)$ (cf. (4)) is R/r-isolated

Equation (6) implies that the roots of $p_{\rm rev}$ are the reciprocals of the roots of p, which leads to the following results:

$$r_j(0,p)r_{d+1-j}(0,p_{\rm rev}) = 1 \text{ for } j = 1,\dots,d. \quad (7)$$

Proposition 1. *The unit disc $D(0,1)$ is θ-isolated for p if and only if it is θ-isolated for p_{rev}.*

The proof of the following theorem of [13] and [1, Theorem 2] is constructive.

Theorem 1. *An algorithm that evaluates at x_0 a black box polynomial $p(x)$ over a field $\mathcal{K}$ of constants by using A additions/subtractions, S multiplications by elements from the field $\mathcal{K}$, and M other multiplications/divisions can be extended to evaluate both $p(x_0)$ and $p'(x_0)$ at the cost $2A + M$, $2S$, and $3M$.*

2.2 Subdivision Iterations

Suppose that we seek all roots of p in a fixed square on the complex plane well-isolated from the external roots of p; call this square *suspect*. One can readily compute such a square centered at the origin and containing all roots of p (cf. [18, Section 6.2]). A subdivision iteration divides every suspect square into four congruent sub-squares and to each of them applies *an exclusion test:* a sub-square is discarded if the test proves that it contains no roots of p; otherwise the sub-square is called suspect and is processed in the next iteration (see Fig. 2).

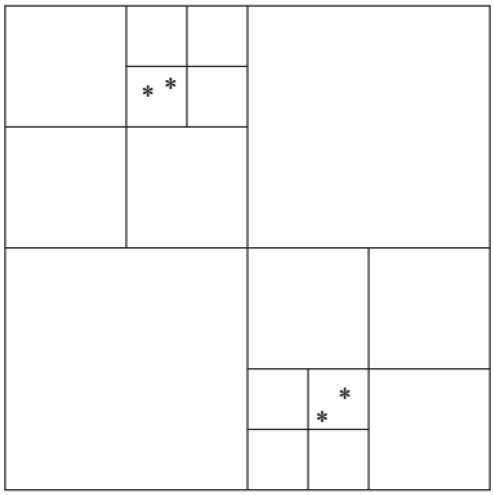

Fig. 2. Four roots of p are marked by asterisks; sub-squares that contain them are suspect; the other sub-squares are discarded

There are at most kd suspect squares at every iteration for a constant k. A root of p can make at most four squares suspect if an exclusion test enabled us to discard every square that contains no roots of p, and then we would have $k \leq 4$. Realistically the subdivision processes have been made less expensive overall by means of incorporation of *soft exclusion tests*, which keep a tested square $S(c, \rho)$

suspect if a disc $D(c, u\rho)$ contains a root of p for some u exceeding $\sqrt{2}$. Then the constant k grows above 4, but the cost of performing exclusion test and the overall cost of subdivision root-finding decrease.

A subdivision iteration begins with approximation of every root of p by the center of some suspect square with an error of at most one half of the diameter of the square and ends with decreasing this bound by twice; the papers [2,14,21] accelerate such a linear convergence to a root to quadratic based on Newton's or QIR iterations applied where one knows: (i) a component formed by suspect squares containing a root of a polynomial p and covered with a well-isolated disc and (ii) a number of the roots of p in that component.

3 Cauchy Root-Counting and Soft Exclusion Test

Algorithm 1. Cauchy sum computation.
Input: An integer $q > 1$, a disc $D(c, \rho)$, and a black box polynomial p of degree d satisfying the following inequalities (cf. Remark 1):

$$p(c + \rho\zeta^g) \neq 0 \text{ for } g = 0, 1, \dots, q-1. \tag{8}$$

Output:[5] The vector $\mathbf{s}_* = (s^*_{q-1}, s^*_0, \dots, s^*_{q-2})^T$ for $s^*_0, \dots, s^*_{q-1}$ of (3).
Computations: Successively compute the values

1. $p(c + \rho\zeta^g)$ and $p'(c + \rho\zeta^g)$ for $g = 0, 1, \dots, q-1$.
2. $r_g := \frac{p'(c+\rho\zeta^g)}{p(c+\rho\zeta^g)}$ for $g = 0, 1, \dots, q-1$,
3. $\tilde{s}_h := \sum_{g=0}^{q-1} \zeta^{(h+1)g} r_g$ for $h = q-1, 0, 1, \dots, q-2$, and
4. $s^*_h = \tilde{s}_h/q$ for $h = q-1, 0, 1, \dots, q-2$.

We evaluate the polynomials $p(x)$ and $p'(x)$ at the q points $c + \rho\zeta^g$ for $g = 0, 1, \dots, q-1$, perform q divisions at each of stages 2 and 4, and perform DFT on q points at stage 3.

Remark 1. We can ensure (8) with probability 1 if we randomly rotate an input disc $D(c, \rho)$: $p(x) \leftarrow t(x)$ for $t(x - c) = p(\alpha \cdot (x - c))$ and a random α sampled under the uniform probability distribution on $C(0, 1)$. At stage 1, we can detect if $p(c + \rho\zeta^g) \approx 0$ and then recursively reapply random rotation. The rotation can be generalized to other maps [18, Remark 8].

Hereafter **Algorithm 1a** denotes Algorithm 1 restricted to the computation of just the first Cauchy sum s^*_0 and its closest integer $\bar{s}_0$ provided that it breaks ties by assigning $\bar{s}_0 = \lceil s^*_0 \rceil$. In transition to Algorithm 1a both stages 1 (dominant) and 2 of Algorithm 1 stay unchanged, but stages 3 and 4 are simplified and involve just $2q$ arithmetic operations.

[5] With this order of its components the vector $\mathbf{s}_*$ turns into the vector of discrete Fourier transform (*DFT*) at q points (upon a reviewer request we recall its celebrated fast solution FFT in the Appendix). Here and hereafter we assume that $\mathbf{v}$ denotes a column vector, while $\mathbf{v}^T$ denotes its transpose.

In the special case where $D(c,\rho) = D(0,1)$, expressions (3) are simplified and if a polynomial p is represented in the monomial basis, then the computation at the bottleneck stage 1 of Algorithm 1a can be reduced to performing DFT at q points twice [18, Sub-algorithm 7.1].

Cauchy sum s_h^* is a weighted power sum s_h, with the weights $\frac{1}{1-x_j^q}$ (see Theorem 2); as a simple Corollary 1, we obtain the bounds of [22] on $|s_h^* - s_h|$.

Theorem 2 *[18]. For the roots x_j of $p(x)$ and all h, the Cauchy sums s_h^* (3) satisfy $s_h^* = \sum_{j=1}^{d} \frac{x_j^h}{1-x_j^q}$ unless $x_j^q = 1$ for some j.*

Corollary 1 *[22].*[6] *Let the disc $D(0,1)$ be θ-isolated and let d_{in} and $d_{\text{out}} = d - d_{\text{in}}$ denote the numbers of the roots of p lying in and outside that disc, respectively. Write $\eta := 1/\theta$. Then*

$$|s_h^* - s_h| \le \frac{d_{\text{in}}\eta^{q+h} + d_{\text{out}}\eta^{q-h}}{1-\eta^q} \text{ for } h = 0, 1, \dots, q-1. \tag{9}$$

In particular[7]

$$s_h = 0 \text{ and } |s_h^*| \le \frac{d\eta^{q+h}}{1-\eta^q} \text{ for } h = 0, 1, \dots, q-1 \text{ if } d_{\text{in}} = 0. \tag{10}$$

$$\mu := |s_0^* - s_0| \le \frac{d}{\theta^q - 1}, \text{ and so } \mu < 1/2 \text{ if } q > \frac{\log(2d+1)}{\log(\theta)}, \tag{11}$$

$$\theta \le \left(\frac{\mu + d}{\mu}\right)^{1/q}, \text{ and so } \theta \le (d+1)^{1/q} \text{ if } \mu = |s_0^* - s_0| \ge 1. \tag{12}$$

Corollary 2. *Suppose that Algorithm 1a, applied to the unit disc $D(0,1)$ for $q \ge b\log_2(2d+1)$ and $b > 0$, outputs $s_0^* > 1/2$ and consequently outputs a positive integer $\bar{s}_0$. Then the disc $D(0,\theta)$ contains a root of p for $\theta = 2^{1/b}$.*

Proof. Suppose that the disc $D(0,\theta)$ contains no roots of p. Then the unit disc $D(0,1)$ contains no roots of p as well and is θ-isolated. Apply bound (11) for $\theta = 2^{1/b}$ and obtain $|s_0^* - s_0| \le \frac{d}{2^{q/b}-1}$ and $q \ge b\log_2(2d+1)$. Conclude that $2^{q/b} \ge 2d+1$ and hence $|s_0^* - s_0| \le 1/2$, while we assumed that $s_0^* > 1/2$.

Remark 2. (i) By applying equations (3) to the reverse polynomial $p_{\text{rev}}(x) = x^d p(\frac{1}{x})$ rather than $p(x)$ extend Corollaries 1 and 2 to the approximation of the power sums of the roots of $p(x)$ lying outside the unit disc $D(0,1)$, whose isolation

[6] Unlike paper [22], this result is deduced in [18] from Theorem 2, which is also the basis for probabilistic support of correctness of Cauchy root-counter in [18].

[7] Clearly, we can only improve our approximation of the integer s_0 by the Cauchy sum s_0^* if we drop its imaginary part $\Im(s_0^*)$. The power sum s_0 of the roots in a well-isolated disc is only slightly closer to $\Re(s_0^*)$ than to s_0^* but can be dramatically closer when some or all roots lie on the boundary circle of an input disc (see [18, Section 3.7]).

ratio is invariant in the transition from p to $p_{\rm rev}$, by virtue of Proposition 1. (ii) Extend Corollaries 1 and 2 and part (i) of this remark to the case of any disc $D(c,\rho)$ by means of shifting and scaling the variable $y \leftarrow \frac{x-c}{\rho}$ and observing that this does not change the isolation ratio of the disc (see the definition of the isolation ratio and Proposition 1).

We obtain a root-counter s_0 in a disc by means of rounding the 0th Cauchy sum s_0^* if $\tau = |s_0 - s_0^*| < 0.5$, e.g., if $q > \log_\theta(2d+1)$ for $\theta > 1$ by virtue of (11), and if $\theta = 2$, then we can choose any $q \geq 21$ for $d \leq 1,000,000$.

Seeking correct output of a Cauchy root-counter or exclusion test without unnecessary increase of the parameter q, one can first apply Algorithm 1a for a small integer q and then recursively double it, reusing the results of the previous computations, until the computed values of the Cauchy sum s_0^* stabilize near an integer or just until they approximate an integer closely enough. [18, Section 5] proves that such an integer is s_0 with a high probability (hereafter *whp)* under random root models.

This result supports root-finding computations in [18, Section 6.4], but not in the subdivision processes of [2,14,21], where root-counting is applied only where an input disc is well-isolated, and then Algorithm 1a yields non-costly solution s_0 by virtue of Corollary 1. For correctness of our exclusion test, we seek stronger support because in the subdivision iterations of [2,14,21], such a test is applied to the discs for whose isolation ratios no estimates are known. By virtue of Corollary 2 Algorithm 1a applied to such a disc certifies that its controlled dilation contains a root of p unless the algorithm outputs $\bar{s}_0 = 0$. The following algorithm completes an exclusion test in the latter case.

Algorithm 2. Completion of a Cauchy soft exclusion test.
Input: A black box polynomial $p(x)$ of degree d such that Algorithm 1a, applied to the or equivalently to the disc $D(0,2)$ and[8] the polynomial $p(x)$, has output $\bar{s}_0 = 0$.
Output: Certification that (i) the disc $D(0,2)$ contains a root of p definitely if $q > d$ or whp otherwise or (ii) the unit disc $D(0,1)$ definitely contains no roots of p, where cases (i) and (ii) are compatible.
Initialization: Choose an integer q such that

$$q_0 < q \leq 2q_0 \text{ for } q_0 \geq \max\{1,\ \log_2(\frac{d}{q_0\alpha_d\sqrt{3}})\} \text{ and } \alpha_d = \sqrt{d+\sqrt{d}}. \tag{13}$$

Computations: Apply Algorithm 1 to the unit disc $D(0,1)$ for the selected q. Let

$$\mathbf{s}_* := (s^*_{q-1}, s^*_0, s^*_1, \dots, s^*_{q-2})^T \tag{14}$$

denote the vector of the values s_h^* of the Cauchy sums output by the algorithm and let $|\mathbf{s}_*|$ denote the Euclidean norm $(\sum_{h=0}^{q-1}|s_h^*|^2)^{1/2}$. If $|\mathbf{s}_*|\ q_0\ \alpha_d \geq 1$, conclude that the disc $D(0,\theta)$ definitely contains a root of p. Otherwise conclude that the disc $D(0,1)$ contains no roots of p definitely if $q > d$ or whp otherwise.

[8] One can extend the algorithm by applying Algorithm 1a to a disc $D(0,\theta)$ for smaller $\theta > 1$ and modifying bound (13) accordingly.

Correctness proof. If the disc $D(0,\theta)$ contains no roots of p, then the unit disc $D(0,1)$ is θ-isolated, and we can apply bound (10) to the Cauchy sums s_h^* output in the above application of Algorithm 1. This would imply that $|s_h^*| \le \frac{d}{(\theta^q-1)\theta^h}$, and then we would deduce that $|s_h^*|^2 \le \frac{d^2}{(\theta^q-1)^2\theta^{2h}}$, and so $|\mathbf{s}_*|^2 \le \frac{d^2}{(\theta^q-1)^2(\theta^2-1)}$. Under the assumed choice of $\theta = 2$ it follows that

$$|\mathbf{s}_*| \le \frac{d}{(2^q-1)\sqrt{3}} < \frac{d}{(2^{q_0}-1)\sqrt{3}} \quad \text{for } q_0 < q,$$

and then (13) would imply that $|\mathbf{s}_*|\ q_0\ \alpha_d \le 1$ and hence $|\mathbf{s}_*|\ q\ \alpha_d < 1$. Therefore, the disc $D(0,\theta)$ contains a root of p unless the latter bound holds, as claimed. Correctness of the algorithm in the case where $|\mathbf{s}_*|\ q\ \alpha_d < 1$ follows from [18, Corollaries 4.2 and 4.3].

Remark 3. For the computation of Cauchy sums for q of order of d we should evaluate p and p' at order of d points; by applying our reduction of multipoint polynomial evaluation (MPE) to fast multipole method (FMM) (see [18, Appendix E]) we can do this by using order of $d\log^2(d)$ arithmetic operations, performed numerically with the precision of order $\log(d)$ bits. It outputs the vector of the first q Cauchy sums $s_0,\dots,s_{q-1}$ within a relative error of order $\log(d)$. This should be sufficient in order to verify the bounds of Algorithm 2 [18, Theorem 19 and Corollaries 4.2 and 4.3], because FMM is celebrated for being very stable numerically, although further formal and experimental study is in order.

The algorithm runs faster as we decrease integer q, and already for q_0 of order of $\log(d)$ the disc $D(0,2)$ contains a root of p if $|\mathbf{s}_*|\ q_0\ \alpha_d \ge 1$, while the unit disc $D(0,1)$ contains a root of p with a probability that fast converges to 1 as the value $|\mathbf{s}_*|\ q_0\ \alpha_d$ decreases under a random coefficient model for the polynomial p, by virtue of [18, Corollary 4.3].

For $q \le d$ we have only probabilistic support of correctness of Algorithm 2 in the case where $|\mathbf{s}_*|\ q_0\ \alpha_d < 1$, but we can try to strengthen reliability of our exclusion tests by verifying additional necessary conditions for correctness of our exclusion test and root-counting:

(a) the Cauchy sums s_h^* for $h = 0,1,\dots,q-1$ still nearly vanish for the polynomials $t(x)$ obtained from $p(x)$ by means of various mappings of the variable x that keep an input disc and the power sum s_0 invariant (cf. Remark 1);
(b) an exclusion test should succeed for any disc lying in the disc $D(c,\rho)$. In particular, if the disc covers a suspect square, then exclusion tests should succeed for the four discs that cover the four congruent sub-squares obtained from sub-dividing the input square;
(c) all suspect squares of a subdivision iteration together contain precisely d roots of p.

If these additional necessary conditions hold, it is still plausible that the disc $D(c,\rho)$ contains a root of p.[9] We can, however, detect whether we have lost

[9] A polynomial p has no roots in a closed disc $D(0,1)$ if and only if $p_{\rm rev}$ has precisely d roots in the open disc $D(0,1)$; similar property holds for Cauchy sums s_0^* (see [18]).

any roots at the end of the subdivision process, when $d-w$ roots are *tamed*, that is, closely approximated, and when w roots remain at large; we call the latter roots *wild*. If $0 < w \ll d$, then at a low cost we can deflate the *wild factor* of p, whose root set is made up of the w wild roots; then we can approximate the roots of this factor at a low cost (see [18, Section 7]).

It is natural to call a point c a tame root of p if $r_d(c,p) \leq$ TOL for a fixed tolerance TOL. The algorithm of [18, Section 6.2] closely approximates $r_d(c,p)$ at a relatively low cost, but it is even less expensive to verify whether $d\,|p'(c)/p(c)| \leq$TOL and then to recall that $r_d(c,p) \leq d\,|p'(c)/p(c)|$ (see [8, Theorem 6.4g]), although his upper bound on $r_d(c,p)$ is extremely poor for a worst case input such as $p(x) = x^d - h^d$ for $h \neq 0$.

Empirical support from the initial implementation and testing of our algorithms in [10] has substantially superseded their formal support here and in [18]. In these tests, subdivision iterations with Cauchy exclusion tests by means of Algorithm 1a have consistently approximated the integer $s_0 = 0$ within $1/4$ for $q = \lceil \log(4d+1)/\log(4\theta) \rceil$. For discs containing no roots and for $q = \lceil \log(4d+1)/\log(4\theta) \rceil + 1$ Algorithm 1 has consistently approximated both s_0 and s_1 within $1/4$ (cf. [10, equation (22) in Corollary 12 for $e = 1/4$]).

4 Real Root-Finding

The algorithm of [12] specializes subdivision iterations of [2] to real univariate polynomial root-finding and is currently the user's choice algorithm, but we can readily accelerate it by narrowing the search for the real roots by means of incorporation of the techniques of [18, Section 6].

Furthermore, we can extend all our other accelerations of subdivision iterations from a disc to a line segment. E.g., our simple root-counting Algorithm 5 provides more information than the customary counting based on the Descartes rule of signs and on Budan–Fourier theorem, involves no costly computation of the Sturm sequences, and unlike Budan–Fourier theorem can be applied to black box polynomials. Our real root-counter amounts essentially to multipoint evaluation. Allowing also interpolation we deflate a factor of p whose root set is precisely the set of roots of p that lie in a fixed segment of real axis. Typically, the degree of the factor is dramatically smaller than d, even where the line segment contains all real roots of p (cf. [6]), and so root-finding on the segment is simplified accordingly.

Actually by saying "real roots" we mean both real and nearly real roots, which is appropriate where an input polynomial is studied numerically with rounding errors.

Next we extend our root-counters and deflation algorithms from the unit disc $D(0,1)$ to the unit segment $S[-1,1]$, but this actually covers the case of any disc and any segment: we can perform the relevant shift and scaling of the variable implicitly because we reduce our real root-counting essentially to multipoint evaluation and reduce real root-finder to multipoint evaluation and interpolation.

We first reduce root-counting and root-finding on the unit segment $S[-1,1]$ to that on the unit circle $C(0,1)$. We assume that the segment is reasonably well isolated from the external roots of p, and this implies that so is the unit circle (see Remark 4). Then Algorithm 5 for root-counting is readily reduced to our root-counting on the complex plane.

Root-counting and root-finding on an isolated circle are reduced to the same tasks for a pair of θ-isolated discs $D(0,\theta_-)$ and $D(0,\theta_+)$ for a constant $\theta > 1$; for root-counting it is sufficient to apply Algorithm 1a (see Algorithm 5).

By scaling the variable x we reduce the root-finding task to that in the well-isolated unit disc $D(0,1)$, and then apply highly efficient deflation algorithms of [22, Section 13].

It remains to specify back and forth transition between the segment and the circle. We apply the two-to-one Zhukovsky's function $z = J(x)$ and its one-to-two inverse, for complex variables x and z. It maps the circle $C(0,1)$ to the segment $S[-1,1]$, and vice versa:

$$x = J(z) := \frac{1}{2}\Big(z + \frac{1}{z}\Big);\ z = J^{-1}(x) := x \pm \sqrt{x^2-1}. \tag{15}$$

Algorithm 3. Root-finding on a line segment.
Input: A polynomial $p = p(x)$ of (1).
Output: The number w of its roots on the unit segment $S[-1,1]$ and approximations to all these roots.
Computations:

1. Compute the values $v_h = p(\Re(\zeta_{2d}^h))$ of the polynomial p at the Chebyshev points $\Re(\zeta_{2d}^h) = \cos(\frac{\pi h}{2d})$ for $h = 0,1,\dots,2d-1$ and ζ_{2d} of (3).
2. Interpolate to the polynomial $s(z)$ of degree $2d$ such that

$$s(z) = z^d p(x) \text{ for } x = \frac{1}{2}(z + z^{-1}) \tag{16}$$

   from its values

$$s(\zeta_{2d}^h) = (-1)^h v_h \text{ for } h = 0,1,\dots,2d-1$$

   by means of applying Inverse DFT. [Recall that $\zeta_{2d}^{dh} = (-1)^h$.]
3. Approximate a factor (e.g., the monic factor) $g = g(z)$ of p whose root set is made up of all roots of the polynomial $s(z)$ that lie on the unit circle $C(0,1)$. [By virtue of (15) $\Re(s(z_j)) = 0$ if $\Re(s(z_j^{-1})) = 0$, and so these roots appear in complex conjugate pairs; if $p(x) = 0$ for $x = 1$ and/or $x = -1$, then 1 and/or -1 are also the roots of $g(z)$ with double multiplicity.] At this stage, first compute the power sums of the roots of $g(z)$ by applying Algorithm 1 to the discs $D(0,1/\theta)$ and $D(0,\theta)$ provided that the circle $C(0,1)$ is θ^2-isolated; then recover the coefficients of $g(z)$ from the power sums by applying [22, Algorithm 46]. Output $w = 0.5\deg(g)$.
4. By applying MPSolve, subdivision iterations, or another root-finder approximate all $2w$ roots of $g(z)$. Let $z_1,\dots,z_w$ denote the first w of them in the order of increasing their arguments from 0.

5. Compute and output the w roots of p lying in the segment $S[-1,1]$ and given by the values $x_j = \frac{1}{2}(z_j + \frac{1}{z_j})$ for $j = 1, \dots, w$.

Correctness of this algorithm follows from (15) and (16).

Its Stage 2 is DFT. Its Stage 1 of the evaluation at Chebyshev's points can be performed by means of the algorithm of [7] or Discrete Cosine Transform, which is similar to DFT (cf. [15, Section 3.11 and the notes to it in Sect. 3.12]). In both cases, the computation is simplified if d is a power of 2; we ensure this by replacing p with $x^u p$ for the minimal non-negative integer u such that $d + u$ is a power of 2. Studying stage 3 observe that the polynomial $g(z)$ has the same roots $z_1, \dots, z_{2w}$ on the circle $C(0,1)$ and in a concentric annulus defined as the difference of two θ-isolated discs $D(0,\theta)$ and $D(0,1/\theta)$ for some $\theta > 1$.

All power sums s_h of these roots are the differences of the power sums s_h of the roots in these two discs. At first closely approximate these pairs of power sums by applying Algorithm 1; their differences approximate the power sums of the roots of $g(z)$ on the circle $C(0,1)$. Then approximate $g(z)$ by applying the algorithms of [22, Section 13].

The converse transition from the unit circle $C(0,1)$ to the unit segment $S[-1,1]$ enables us to simplify stages 4 and 5 of Algorithm 3 by moving from $g(z)$ to a polynomial of degree w whose all roots lie in the segment $S[-1,1]$. Then again we achieve this by means of evaluation and interpolation.

Algorithm 4. Transition to unit segment.

1. Compute the values $u_h = g(\zeta_{2K}^h)$ of the polynomial $g(z)$ at the $2K$-th roots of unity for $h = 0, 1, \dots, K-1$ and $K > w$.
2. Interpolate to the polynomial $f(x)$ of degree at most w from its values $f(\Re(\zeta_{2K}^h)) = (-1)^h u_h$ at the Chebyshev points $\Re(\zeta_{2K}^h) = \cos(\frac{\pi h}{2K})$, for $h = 0, 1, \dots, K-1$. [Recall that $\zeta_{2K}^{Kh} = (-1)^h$.]
3. Approximate the w roots of the polynomial $f = f(x)$ by applying to it MPSolve, subdivision iterations, or another real polynomial root-finder, e.g., that of [12].

We propose to perform steps 1 and 2 above by means of forward DFT and inverse Cosine Transforms, applying them to the polynomial $z^v g(z)$, replacing $g(x)$, for the minimal non-negative integer v such that $w + v$ is a power 2.

Remark 4. Represent complex numbers as $z := u + v\mathbf{i}$. 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 $E(0,\theta)$, and so the circle $C(0,1)$ is θ-isolated if and only if no roots of p lie in the latter domain.

We can simplify Algorithm 3 and use only evaluations if we restrict our task to counting the roots that lie on or near a line segment. Here is a high level description of this algorithm where we do not specify the parameters θ and q and assume that the input includes the polynomial $s(z)$.

Algorithm 5. Root-counting on a line segment.
Input: A real $\theta > 1$ and the polynomial $s(z)$ of Eq. (16) such that the unit circle $C(0,1)$ is θ^2-isolated, that is, the annulus $A(0,1/\theta^2,\theta^2)$ contains no roots of $s(z)$ except possibly some roots on the unit circle $C(0,1)$.
Output: The number w of the roots of p in the segment $S[-1,1]$ or *FAILURE*.
Computations:

1. Compute the polynomial $s_-(z) = s(z/\sqrt{\theta})$.
2. Choose a sufficiently large q and compute Cauchy's sums $\tilde{s}^*_{0,-}$ of the roots of $s_-(z)$ in the unit disc $D(0,1)$.
3. If the value $s^*_{0,-}$ is sufficiently close to an integer $\tilde{s}_{0,-}$, then output $w := d - \tilde{s}_{0,-}$ and stop. Otherwise output *FAILURE*.

By assumption, the circle $C(0,1/\theta)$ is $(\theta-\epsilon)$-isolated for $s(z)$ and any positive ϵ, and so $\tilde{s}_{0,-}$ is the number of the roots of $s(z)$ in the disc $D(0,1/\theta)$ by virtue of bound (11). Clearly the same number $\tilde{s}_{0,-}$ of the roots of the polynomial $s(z)$ of degree $2d$ lie inside and outside the unit disc, and so $2d - 2\tilde{s}_{0,-}$ its roots lie on the unit circle $C(0,1)$. Divide this bound by 2 and obtain the number of the roots of $p(x)$ on the unit segment $[-1,1]$.

Acknowledgements. This research has been supported by NSF Grants CCF–1563942 and CCF–1733834 and PSC CUNY Award 69813 00 48. We also thank the reviewers for thoughtful comments.

Appendix A. Discrete Fourier transform (DFT)

DFT(**p**) outputs the vector of the values $p(\zeta^j) = \sum_{i=0}^{d-1} p_i \zeta^{ij}$ of a polynomial $p(x) = \sum_{i=0}^{d-1} p_i x^i$ on the set $\{1, \zeta, \dots, \zeta^{d-1}\}$. The *fast Fourier transform (FFT) algorithm*, for $d = 2^h$ recursively splits $p(x)$:

$$p(x) = p_0(y) + xp_1(y), \text{ where } y = x^2,$$
$$p_0(y) = p_0 + p_2x^2 + \cdots + p_{d-2}x^{d-2},\ p_1(y) \quad = x(p_1 + p_3x^2 + \cdots + p_{d-1}x^{d-2}).$$

This reduces DFT_d for $p(x)$ to two $\mathrm{DFT}_{d/2}$ (for $p_0(y)$ and $p_1(y)$) at a cost of d multiplications of $p_1(y)$ by x, for $x = \zeta^i$, $i = 0, 1, \dots, d-1$, and of the pairwise addition of the d output values to $p_0(\zeta^{2i})$. Since $\zeta^{i+d/2} = -\zeta^i$ for even d, we perform multiplication only $d/2$ times, that is, $f(d) \le 2f(d/2) + 1.5d$ if $f(k)$ ops are sufficient for DFT_k. Recursively we obtain the following estimate.

Theorem 3. *For $d = 2^h$ and a positive integer h, the DFT_d only involves $f(d) \le 1.5dh = 1.5d\log_2 d$ arithmetic operations.*

Inverse DFT is the converse problem of interpolation to a polynomial $p(x)$ from its values at the dth roots of unity. At the cost of performing d divisions, this task can be reduced to DFT (see, e.g., [15, Theorem 2.2.2]).

References

1. Baur, W., Strassen, V.: On the complexity of partial derivatives. Theoret. Comput. Sci. **22**, 317–330 (1983)
2. 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), Proceedings version. In: ACM ISSAC, pp. 71–78 (2016). https://doi.org/10.1016/j.jsc.2017.03.009
3. Bini, D.A., Fiorentino, G.: Design, analysis, and implementation of a multiprecision polynomial rootfinder. Numer. Algorithms **23**, 127–173 (2000). https://doi.org/10.1023/A:1019199917103
4. Bini, D., Pan, V.Y.: Graeffe's, Chebyshev-like, and Cardinal's processes for splitting a polynomial into factors. J. Complex. **12**, 492–511 (1996)
5. Bini, D.A., Robol, L.: Solving secular and polynomial equations: a multiprecision algorithm. J. Comput. Appl. Math. **272**, 276–292 (2014). https://doi.org/10.1016/j.cam.2013.04.037
6. Erdős, P., Turán, P.: On the distribution of roots of polynomials. Ann. Math **2**(51), 105–119 (1950)
7. Gerasoulis, A.: A fast algorithm for the multiplication of generalized Hilbert matrices with vectors. Math. Comput. **50**(181), 179–188 (1988)
8. Henrici, P.: Applied and Computational Complex Analysis, Vol. 1: Power Series, Integration, Conformal Mapping, Location of Zeros. Wiley, New York (1974)
9. Householder, A.S.: Dandelin, Lobachevskii, or Graeffe? Amer. Math. Mon. **66**, 464–466 (1959). https://doi.org/10.2307/2310626
10. Imbach, R., Pan, V.Y.: New progress in univariate polynomial root-finding. In: Proceedings of ACM-SIGSAM ISSAC 2020, pp. 249–256, July 20–23, 2020, Kalamata, Greece, ACM Press, New York (2020). ACM ISBN 978-1-4503-7100-1/20/07. https://doi.org/10.1145/3373207.3403979
11. 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
12. Kobel, A., Rouillier, F., Sagralo, M.: Computing real roots of real polynomials... and now for real! In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation (ISSAC 2016), pp. 301–310. ACM Press, New York (2016) https://doi.org/10.1145/2930889.2930937
13. Linnainmaa, S.: Taylor expansion of the accumulated rounding errors. BIT **16**, 146–160 (1976)
14. 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). https://doi.org/10.1006/jcom.1999
15. 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

16. Pan, V.Y.: Univariate polynomials: nearly optimal algorithms for factorization and rootfinding. J. Symb. Comput. **33**(5), 701–733, : Proceedings version. ACM STOC **1995**, 741–750 (2002). https://doi.org/10.1006/jsco.2002.0531
17. Pan, V.Y.: Old and new nearly optimal polynomial root-Finding. In: England,M., Koepf, W., Sadykov, T.M., Seiler, W.M., Vorozhtsov, E.V. (eds.) CASC2019. LNCS, vol. 11661, pp, 393–411. Springer, Nature Switzerland (2019) https://doi.org/10.1007/978-3-030-26831-2
18. Pan, V.Y.: New Acceleration of Univariate Polynomial Root-finders, August 2020. arXiv: 1805.12042
19. Pan, V.Y.: Acceleration of subdivision root-finding for sparse polynomials. to appear. In: Boulier, F., England, M., Sadikov, T.M., Vorozhtsov, E.V. (eds.) CASC 2020. Springer Nature, Switzerland (2020)
20. 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. 347–358. Springer, Heidelberg (2015). arXiv:1501.05386
21. Renegar, J.: On the worst-case arithmetic complexity of approximating zeros of polynomials. J. Complex. **3**(2), 90–113 (1987). https://doi.org/10.1016/0885-064X(87)90022-7
22. Schönhage, A.: The fundamental theorem of algebra in terms of computational complexity. Math. Dept., Univ. Tübingen, Germany (1982)
23. Weyl, H.: Randbemerkungen zu Hauptproblemen der Mathematik. II. Fundamentalsatz der Algebra und Grundlagen der Mathematik. Mathematische Zeitschrift 20, 131–151 (1924)