Good News for Polynomial Root-finding

Victor Y. Pan[1,2,3],[a]
[1] Department of Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
[2] Ph.D. Program in Mathematics and
[3] Computer Science
The Graduate Center of the City University of New York
New York, NY 10036 USA
[a] victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

Abstract

Given a black box oracle that evaluates a univariate polynomial $p(x)$ of a degree d with unknown real or complex coefficients, we seek its complex zeros, aka the roots of the equation $p(x) = 0$. This model is most appropriate for a large and important class of polynomials $p(x)$ whose values are much more readily available than the coefficients, e.g., for the Bernstein, Mandelbrot, and sparse polynomials, which admit numerically stable and/or particularly fast evaluation. At FOCS 2016, Louis and Vempala have approximated within a fixed error tolerance ϵ an absolutely largest zero of a black box polynomial $p(x)$ at the cost of its evaluation at $O(b\log(d))$ points x provided that $2^b = R/\epsilon$ and all zeros of $p(x)$ lie in the segment $[-R, R]$; by extending this algorithm they approximated an absolutely largest eigenvalue of a symmetric matrix at a record Boolean cost.

By applying *distinct approach and techniques* we obtain *much more general results at the same computational cost.*

We present a stream of novel techniques and algorithms, e.g., our use of Cauchy integrals and randomization is non-trivial and pioneering,

Some of our algorithmic extras are of independent interest. E.g., as by-product we efficiently approximate the ℓ-th root radius, that is, the ℓ-th smallest distance to a zero of a black box polynomial from any fixed point on the complex plane.

Our black box polynomial root-finders perform efficiently even under the model where a polynomial $p(x)$ is general and is given with its coefficients. Even under that model and with incorporation of recent acceleration, they run in *nearly optimal* Boolean time and compete with and frequently supersede user's choice root-finders.

Keywords: polynomial computations, matrix eigenvalues, complexity, polynomial zeros, black box polynomials, symbolic-numeric computing

2020 Math. Subject Classification: 65H05, 26C10, 30C15

PART 0: Introduction

1 Introduction

1.1 Root-finding problem: early history

Solution of a univariate polynomial equation, hereafter referred to as *polynomial root-finding*, has been the central problem of mathematics and computational mathematics for about 4,000 years, since Sumerian times and well into the XIX century (see Eric Temple Bell [7], Uta C. Merzbach and Carl B. Boyer [72], and P. [86, 87]).

In the long way from its rudimentary form, written on clay tablets and in manuscripts on papyrus, this study has led to modern mathematical notation and concepts of irrational, negative, complex numbers, meromorphic functions, algebraic groups, rings, fields etc.

Here is a modern version of this venerated problem:

Problem 1: Given a positive $\epsilon = 1/2^b > 0$ and $d + 1$ real or complex numbers $p_0, p_1, \dots, p_d$, $p_d \neq 0$, approximate within ϵ all d complex roots[1] $z_1, \dots, z_d$ of the equation $p(x) = 0$ for

$$p = p(x) := \sum_{i=0}^{d} p_i x^i := p_d \prod_{j=1}^{d} (x - z_j), \;\; p_d \neq 0. \tag{1.1}$$

The existence of d complex roots of the equation $p(x) = 0$ is called the *Fundamental Theorem of Algebra*. A number of leading mathematicians of 18th century attempted to prove it. Johann Carl Friedrich Gauss alone proposed four proofs during his long career, although the proofs were by far not complete by modern standards (cf. Smale Smale [119]).

Initially people looked for explicit rational expression for the roots z_j of Eqn. (1.1) through the coefficients p_i, but it was proved already in Pythagorean school in the 5th century BC (possibly by Hippasus of Metapontum) that there exists no such expressions even for the equation $x^2 - 2 = 0$, and this was the first rigorous mathematical proof known to us.

By virtue of Abel's impossibility theorem, due to Paolo Ruffini 1799 (completed by Augustin-Louis Cauchy) and Niels Henrik Abel 1824, one cannot express the roots through the coefficients rationally even with radicals unless the polynomial has degree less than 5, and the Group theory, by Évariste Galois, implies that this is impossible even for specific polynomials, e.g., $p(x) = x^5 - 4x - 2$.

1.2 Root-finding algorithms and complexity: overview

One can, however, approximate the roots of (1.1) closely; this task has begun a new life with the advent of computers. Algorithms for approximate solution of Problem 1 can be traced back to Herman Weyl's constructive proof of Fundamental Theorem of Algebra in [134], which enables rational approximation to all zeros of any polynomial with any precision. His algorithm has been strengthened by Peter Henrici, Games Renegar, P., Soo Go, Qi Luan, and Liang Zhao in [46, 114, 88, 99, 101] and now promises to have practical value.

By now, hundreds of efficient techniques and algorithms have been proposed for Problem 1 and related computational problems (see McNamee [70], McNamee and P. [71]) and new algorithms keep appearing (see P. and Elias P. Tsigaridas [108], Dario Andrea Bini and Leonardo Robol [21], Ioannis Z. Emiris, P., and Tsigaridas [36], Alexander Kobel, Fabrice Rouillier, and Michael Sagraloff [62], P. and Liang Zhao [113, 95], Ruben Becker, Sagraloff, Vikram Sharma, Juan Xu, and Chee Yap

[1] Hereafter "roots" stands for "roots of equation $p = 0$", aka "zeros of a polynomial p"; we count them with their *multiplicity* and occasionally call them just *roots* or *zeros*.

[22], Becker, Sagraloff, Sharma, and Yap [23], Rémi Imbach, P., and Chee Yap [55], P. [97, 98], Qi Luan, P., Wongeun Kim, Vitaly Zaderman [67], Imbach and P. [51, 53, 54], P., Soo Go, Qi Luan, and Liang Zhao [101], and the bibliography therein), and we report new significant progress by combining some selected known methods with our brains and sweat and hoping to enliven research activity in the field with our conceptual and technical novelties.

Smale in [119] proposed to study Problem 1 in terms of arithmetic complexity and computational precision and later (jointly with Lenore Blum, Felipe Cucker, and Michael Shub) (cf. [13, 27]) linked this study to rounding errors and conditioning of the roots.

Arnold Schönhage in [120] proposed Boolean time as the complexity measure and devised an advanced algorithm that solved Problem 1 in Boolean time $\tilde{O}(d^3b)$.

Here and hereafter $\tilde{O}(w)$ denotes w up to a poly-logarithmic factor.

Subsequent study in the 1980s and 1990s has culminated at ACM STOC 1995 with a divide-and-conquer algorithm of [83, 84, 86, 87, 90], which extended the work of Schönhage [120] and Andrew C. Neff and John H. Reif [75]. It approximates all d zeros of p in arithmetic time $\tilde{O}(d^2)$ by computing with $O(b + d\log(d))$-bit precision. Hence it runs in nearly optimal Boolean time $\tilde{O}((b + d)d^2)$, that is, approximates all linear factors and complex roots of a polynomial almost as fast as one accesses the input coefficients with the precision required for these tasks.

The algorithm, however, is quite involved and has never been implemented.[2]

Two other polynomial root-finders known to be nearly optimal have been proposed by Ruben Becker, Michael Sagraloff, Vikram Sharma, Juan Xu, and Chee Yap and by four of these authors in 2016 in [22] and in 2018 in [23], respectively.[3] The authors have revisited and modified the popular subdivision iterative root-finders, introduced by Herman Weyl in 1924 [134], advanced by Peter Henrici in [46], by James Renegar in [114], and by P. in [88], and under the name of *Quad-tree Construction* extensively used in Computational Geometry.

Becker et al claimed just two algorithmic novelties, that is, an extension of John Abbott's combination of Newton's and secant iterations [2] from the real line to the complex plane and application of Pellet's classic theorem to estimating the distances from the origin to the roots. Extension of Abbot's iterations is useful although not impressive, while already Pellet's theorem itself, a direct consequence of Rouché's theorem, immediately defines the distances to the roots in terms of input coefficients, and so the novelty is limited to its proper incorporation.

Otherwise 46 pages of [23] have been devoted to alternative formal support, based on the known algorithms, for the record low and nearly optimal Boolean complexity of root-finding, achieved earlier in [83, 84, 86, 87, 90]. In Secs. 1.7 and 1.12 we demonstrate, however, that the proved upper bound on the complexity of the root-finders of [22, 23] definitely exceeds the proved bound of the root-finder of [83, 84, 86, 87, 90] by a factor of d and that this is likely to be the deficiency of the root-finders of [22, 23] rather than of their analysis. This leaves little hope to salvage the optimality claim of [22, 23], which, however, prompted us to reexamine our algorithm of [88].

Unlike [22, 23] our current extensive study of subdivision root-finding focuses on introducing a stream of technical and algorithmic novelties, which enables us to support root-finding in nearly optimal time, although unlike [83, 84, 86, 87, 90] we only achieved this under Las Vegas randomization in the space of zeros of an input polynomial.

For their highly important advantage versus [83, 90, 22, 23] and all previous best root-finders, our root-finders can be applied to a polynomial represented by an oracle (that is, black box sub-

[2] Due to extensive progress on implementation of fast polynomial root-finders in the last decades, implementation of that algorithm seems to be much less formidable now than in 2002. On the other hand, nowadays a faster root-finders readily lose competition to ones that are slower but better implemented.

[3] For its full version the proceedings paper [22] refers to pre-print of [23], but the Main theorems of the two papers are quite different.

routine) for their evaluation rather than by their coefficients. Hence we gain additional acceleration where that polynomial can be evaluated fast, e.g, is a sum of a small number of shifted monomials $\sum_{i=1}^{h} a_i(x-c_i)^d$ for complex constants a_i and c_i and a reasonably small positive integer h.

1.3 Applications and implementation of root-finders: brief overview

Numerous areas of mathematics and computational mathematics arisen and developed in the last 100 years had no direct link to Problem 1. Furthermore, many computational problems arising in the sciences, engineering, business management, and statistics have been linearized and then solved by using tools from linear algebra, linear programming, fast Fourier transform (FFT), and Fast Multipole Method (FMM).[4] Such tools may involve the solution of Problem 1 but usually for smaller degree d, where the available subroutines are sufficiently effective in most cases.

Solution of Problem 1 of large degree, however, still has important applications to various areas of Scientific Computing, in particular to Computer Algebra, signal processing, geometric modeling, control, financial mathematics, and complex dynamics (see John Michael McNamee and P. [71, Introduction]).

Unlike the nearly optimal root-finder of [83, 84, 86, 87, 90], various non-optimal complex polynomial root-finders have been implemented (see e.g,, MATLAB "roots", Maple "solve", MPSolve, and CCluster).

In particular, the complex polynomial root-finder of [22] has been implemented in [55] and then greatly strengthened in [51, 54] based on combining root-finder of [22] with out recent algorithms.

Since its creation in 2000 the package MPSolve by Bini and Guiseppe Fiorentino [14], revised in 2014 by Bini and Leonardo Robol [21], has been remaining user's choice for approximation of all d complex roots of a polynomial p. It implements Ehrlich's iterations of 1967 by Louis W. Ehrlich [34], aka by Oliver Aberth of 1973 [3].

Empirically, MPSolve is the fastest known root-finder where one seeks all complex roots of a polynomial. This is because every Ehrlich's iteration is performed fast and because empirically the iterations converge very fast globally – right from the start, under proper initialization, involving the coefficients of p. Such convergence has been consistently observed in decades-long worldwide applications of these iterations as well as in the case of some other root-finders by means of functional iterations, notably Weierstrass's iterations, due to Karl Theodor Wilhelm Weierstrass [133], aka Emile Durand and Immo O. Kerner's [28, 59]).

Persistent attempts of supporting this empirical behavior formally, e.g., [4, 47], however, have recently ended with surprising proofs that Weierstrass's and Ehrlich's iterations diverge in the case of inputs that lie on some complex manifolds, being an open domain in the case of Weierstrass's iterations (see Bernhard Reinke, Dierk Schleicher, and Michael Stoll [117, 115]).

Serious competition to MPSolve came in 2001 from the package EigenSolve by Steven J. Fortune [38], but the later version of MPSolve [21] combines the benefits of both packages. An implementations of Newton's iterations for approximation of all roots by Marvin Randig, Schleicher, and Robin Stoll in [116] and by Schleicher and Robin Stoll in [128] competed with MPSolve in 2017, but then MPSolve has regained upper hand by means of incorporation of Fast Multipole Method (FMM) .

The root-finder of [22], efficiently implemented in [55], slightly outperforms MPSolve for root-finding in a region of the complex plane containing a small number of roots, although it is by far inferior for the approximation of all d roots of a polynomial $p(x)$.[5]

[4] Both FFT and FMM are among the Top Ten Algorithms of 20-th century (cf. Barry Cipra [25]). For the history of FFT see Donald Ervin Knuth [61]; on FMM see Leslie F. Greengard and Vladimir Rokhlin Jr. [44] and Rio Yokota and Lorena A. Barba [137].

[5] The computational cost of root-finders of [83, 90, 22, 23] decreases at least proportionally to the number of roots

The implementation of real subdivision root-finding in [62] is user's current choice, and this is a highly important problem because in many applications, e.g., to optimization in algebraic geometry and geometric modeling, only real roots of a polynomial are of interest and typically are much less numerous than all d complex roots (cf. Marc Kac [58], Alan Edelman and Eric Kostlan [35]).

It is intriguing to witness the competition for the user between functional and subdivision iterations. Currently only a part of our progress has been implemented (see [51, 53, 54]), but [53] has already noticeably accelerated the real root-finder of [62], and [54] has dramatically accelerated [55]. Moreover, according to tests with standard test polynomials [54] has competed with MPSolve even for approximation of all zeros of $p(x)$, and for a large class of inputs accelerated it significantly.

1.4 Two variations of Problem 1 and their complexity

Problem 1 is a special case for $m = d$ of the following important

Problem 1_m: Polynomial Root-finding in a Disc.[6] Restrict Problem 1 of root-finding within ϵ to a fixed disc $D = D(c, \rho)$ having center c and radius ρ and containing $m \le d$ zeros of p such that the $d - m$ external zeros lie at the distance at least $(\theta - 1)\rho$ from the disc D for $\theta - 1$ exceeding a fixed positive constant.

Now write $|u| = \sum_{i=0}^{d} |u_i|$ for a polynomial $u(x) = \sum_{i=0}^{d} u_i x^i$ and recall that the root-finders of [120, 83, 84, 86, 87, 90] first solve the following problem of independent importance and then extends its solution to the solution of Problem 1:

Problem 2. Approximate Polynomial Factorization:[7] Given a real b and $d+1$ coefficients $p_0, p_1, \dots, p_d$ of (1.1), compute $2d$ complex numbers u_j, v_j for $j = 1, \dots, d$ such that

$$|p - \prod_{j=1}^{d}(u_j x - v_j)| \le 2^{-b}|p|. \tag{1.2}$$

Hereafter $\mathbb{B}_1(b, d)$, $\mathbb{B}_1(b, d, m)$, and $\mathbb{B}_2(b, d)$ denote Boolean time for the solution of Problems 1, 1_m, and 2, respectively, for an error bound $\epsilon = 1/2^b$, and next we specify a class of polynomials whose *zeros make no large clusters* (cf. [13, 27]).

Definition 1.1. Polynomials with bounded cluster sizes. *The zero set of p consists of clusters, each made up of at most $m = O(1)$ zeros and each having a diameter at most $1/2^{3-b/d}$; furthermore, these clusters are pairwise separated by the distances at least $1/d^{\tilde{O}(1)}$.*

Theorem 1.1. *The algorithm of [83, 84, 86, 87, 90] solves Problem 2 in Boolean time $\tilde{O}((b+d)d)$.*

In Sec. A.4 we reduce Problem 1 to Problem 2 and vice versa by applying Schönhage's results of [120, Cor. 4.3 and Sec. 19]:

in a region of interest such as a disc or a rectangle, while Ehrlich's, Weierstrass's, and a number of other functional iterations approximate the roots in such a region almost as slow as all complex roots.

[6]The problem and many of its solution algorithms are naturally extended to root-finding in other regions of interest, such as square or convex polygons. Problem 1 is Problem 1_d for a large disc covering all d roots.

[7]The solution of Problem 2 can be also extended to isolation of the zeros of a polynomial with integer coefficients and approximation of matrix eigenvalues (cf. [120, Secs. 20 and 21] and our Sec. 8) and has various other applications to modern computations, e.g., to time series analysis, Wiener filtering, noise variance estimation, co-variance matrix computation, and the study of multichannel systems (see Granville Tunnicliffe Wilson [135], George Edward Pelham Box and Gwilym Meirion Jenkins [17], Stephen M. Barnett [8], Cédric J. Demeure and Clifford T. Mullis [31, 32], and Paul M. Van Dooren [132]). For a polynomial of degree d the known sharp upper and lower bounds on the input precision and Boolean time for root-finding exceed those for numerical factorization by a factor of d.

Theorem 1.2. *(i)* $\mathbb{B}_1(b,d) = \tilde{O}(\mathbb{B}_2(bd,d))$ *for any polynomial* $p(x)$ *of degree* d, *(ii)* $\mathbb{B}_2 1(b,d) = \tilde{O}(\mathbb{B}_1(b+d+\log(d),d))$ *for any polynomial* $p(x)$ *of degree* d, *and (iii)* $\mathbb{B}_1(b,d) = \tilde{O}(\mathbb{B}_2(b,d))$ *for polynomials of Def. 1.1, with bounded cluster sizes.*

By combining Thms. 1.1 and 1.2 obtain the following estimates.

Theorem 1.3. *One can solve Problem 1 in Boolean time (i)* $\tilde{O}((b+d)d^2)$, *which decreases to (ii)* $\tilde{O}((b+d)d)$ *in the the case of polynomials* p *with bounded cluster sizes.*

Theorem 1.4. *One can solve Problem* 1_m *in Boolean time* $\tilde{O}((b+d)d+(b+m)m^2)$, *which is* $\tilde{O}((b+d)d$ *where* $m^2 = \tilde{O}(d)$.

Proof. Proceed in two steps:
(i) in Boolean time $\tilde{O}((b+d)d)$ solve Problem 2 and then
(ii) apply the algorithm that supports Thm. 1.3 to the mth degree factor of the polynomial $\prod_{j=1}^{d}(u_j x - v_j)$ of (1.2) that shares with this polynomial all its zeros lying in the input disc D.[8] ☐

1.5 Lower bounds on the Boolean complexity

Readily verify the following information lower bounds.

Observation 1.1. *To solve Problems 1 or 2 one must access* $(d+1)b$ *bits of input coefficients and perform at least* $0.5(d+1)b$ *Boolean operations – at most one operation for two input bits.*

Next we strengthen the lower bound for Problem 1.

Hereafter $\zeta := \zeta_q$ denotes a primitive q-th root of unity:

$$\zeta := \exp\left(\frac{2\pi\mathbf{i}}{q}\right), \ \mathbf{i} := \sqrt{-1}. \tag{1.3}$$

Observation 1.2. *Let* $p(x) = (x-z_1)^j f(x)$ *for an integer* $j > 0$ *and a monic polynomial* $f(x)$ *of degree* $d-j$. *Fix a real* b. *Then the polynomial* $p_j(x) = p(x) - 2^{-jb} f(x)$ *has the* j *zeros* $z_1 + \zeta^g 2^{-b}$ *for* $g = 0,1,\dots,j-1$.

Observation 1.2 implies that one must access the coefficient p_{d-j} within 2^{-jb} to approximate within 2^{-b} a j-multiple zero of p or its zero in a cluster of j zeros, and so a root-finder must process at least jb bits of the coefficient p_{d-j}. Sum these bounds for $j = 1,\dots,d$ and obtain

Corollary 1.1. *One must access at least* $0.5(d-1)db$ *bits of the coefficients of* p *and perform at least* $0.25(d-1)db$ *Boolean operations to approximate even a single zero of* p *within* 2^{-b}.

Likewise, we deduce lower bounds on the Boolean time for Problem 1_m.

Corollary 1.2. *To solve Problem* 1_m *and even to approximate within* ϵ *a single zero of* p *in the disc* D, *one must access at least* $0.5b\max\{d+1,(m-1)m\}$ *bits of input coefficients and perform at least* $0.25b\max\{d+1,(m-1)m\}$ *Boolean operations for* $b = \log_2(R/\epsilon)$ *and* $R = |c| + \rho$.

In view of the lower bounds of this subsection, the algorithm of [83, 84, 86, 87, 90] and its extension supporting Thm. 1.4 solve Problems 1, 1_m, and 2 of large size in optimal Boolean time up to poly-logarithmic factors or, in simple terms, almost as fast as one reads all bits of input coefficients required to support computing the output with prescribed accuracy.

[8] We can slightly speed up stage (i) because at that stage we actually only need to approximate factorization of the factor f of degree m of p that shares with p its zeros in D.

1.6 Linking Factorization and Root-finding

Next we prove Thm. 1.2, by applying the results of [120, Cor, 4.3 and Sec. 19].

Recall that $\mathbb{B}_1(b,d)$ and $\mathbb{B}_2(b,d)$ denote the optimal Boolean time for the solution of Problems 1 and 2, respectively, for an error bound $\epsilon = 1/2^b$, and furthermore recall the lower bounds

$$\mathbb{B}_1(b,d) \geq 0.5(d+1)b \text{ and } \mathbb{B}_2(b,d) \geq 0.25(d-1)db \tag{1.4}$$

of [83, 90] and the following result (cf. [120, Thm. 19.1]).

Theorem 1.5. *Let $r_1 = \max_{j=1}^d |z_j| \leq 1$. Then a solution of Problem 1 within an error bound $1/2^b$ can be recovered at a dominated Boolean cost from the solution of Problem 2 within an error bound $1/2^{bd+2}$, and so $\mathbb{B}_1(b,d) \leq \mathbb{B}_2(bd+2,d)$.*

In view of (1.4), relaxing the bound $r_1 \leq 1$ would little affect the latter estimate because map (2.2), of a disc $D(c,\rho)$ into the unit disc $D(0,1)$, can change the error bound of Problem 1 by at most $\log_2(\max\{\rho, 1/\rho\})$, which we assumed to be at most b. Based on [120, Cor. 4.3] we can also readily relax the assumption that $p(x)$ is monic. [122, Thm. 2.7] shows an alternative way to relaxing both assumptions – that $r_1 \leq 1$ and $p(x)$ is monic.

Schönhage in [120, Sec. 19] has greatly strengthened the estimate of Thm. 1.5 unless "the zeros of p are clustered too much". Namely, he proved (see [120, Eqn. (19.8)]) the following result.

Lemma 1.1. *Assume that the polynomial $p(x)$ is monic, the values $u_j = 1$ for all j and complex $v_1, \dots, v_d$ satisfy bound (1.2), and $|z_j - v_j| \leq r := 2^{2-b/d}$ for all j. Furthermore, write $\Delta_{j,1} := |v_j - v_1|$ for $j = 2, 3, \dots, d$ and suppose that $1.25\Delta_{j,1} \leq r$ for $j = 2, 3, \dots, d$ and $\Delta_1 := \min_j \Delta_{j,1} > r$. Then $|z_1 - v_1| \leq 2^{2-b/m}/|\Delta_1 - r|^{\frac{d-m}{m}}$.*

If $1/|\Delta - r|^{\frac{d-m}{m}} = 2^{\tilde{O}(d)}$, then Lemma 1.1 implies that $|z_1 - v_1| \leq 1/2^{b/m - \tilde{O}(d)}$, and we can similarly estimate $|v_j - z_j|$ for all $j = 1, 2, \dots, d$.

By extending the above estimate, we strengthen Thm. 1.5 as follows.

Corollary 1.3. *Under Cluster Restriction of Sec. 1.1 it holds that $|z_j - v_j| \leq 1/2^{b/m - \tilde{O}(d)}$, $j = 1, 2, \dots, d$. Hence if $m = O(1)$ is a constant, then a solution of Problem 1 within an error bound $1/2^b$ can be recovered at a dominated Boolean cost from the solution of Problem 2 with a precision $1/2^{\bar{b}}$ for a sufficiently large $\bar{b} = \tilde{O}(b+d)$, and so $\mathbb{B}_1(b,d) \leq \mathbb{B}_2(\tilde{O}(b+d),d)$.*

For converse reduction of Problem 2 to Problem 1 recall [120, Cor. 4.3] in the case of $k = 2$, covering the product of two polynomials:

Theorem 1.6. *Let $p = gh$ for three polynomials p of (1.1), g, and h. Then $||g||_1 ||h||_1 \leq 2^{d-1}||p||_1$.*

Now consider the products $p = gh$ for $g = x - z_j$, for $j = 1, \dots, d$. For a fixed j, an error of at most $\epsilon/2^d$ for the zero z_j contributes at most $\epsilon||h||_1$ to overall factorization error of Problem 2. Therefore, $||h||_1 \leq 2^{d-1}||p||_1/||x - z_j||_1 \leq 2^{d-1}||p||_1$ by virtue of Thm. 1.6. Hence the approximation errors of all the d zeros of p together contribute at most $2^{d-1}d\epsilon$ to the relative error of the solution of Problem 2 (ignoring dominated impact of higher order terms), and we arrive at

Corollary 1.4. *If $b = O(b - d - \log(d))$. Then Problem 2 for an error bound $1/2^b$ can be reduced at a dominated Boolean cost to Problem 1 for an error bound $1/2^{\widehat{b}}$ for a sufficiently large $\widehat{b} = O(b + d + \log(d))$, and so $\mathbb{B}_2(b,d) = O(\mathbb{B}_1(b,d))$.*

1.7 Are there any research challenges still left?

The algorithm of [120] was implemented by Xavier Gourdon in 1996 for the Magma and PARI/GP computer algebra systems, while the nearly optimal but more involved algorithm of [83, 90] has never been implemented.

Next recall that Becker et al in [22, 23] claimed that their extension of subdivision root-finders of [114, 88] runs within the same nearly optimal Boolean time bounds of [83, 84, 86, 87, 90] but unlike [83, 84, 86, 87, 90] is easy to implement. This claim has been firmly accepted by the research community of Computer Algebra and has established the State of the Art in polynomial root-finding and a high bar for competing works, but the acceptance was a little premature.

Indeed, careful implementation of the root-finder of [22] by Imbach et al in [55] provided no empirical support to optimality claimed by Becker et al because the root-finder of [54], using some novel techniques from [99] (see [54, Sec. 1.2]), runs much faster.

Unfortunately, formal support for optimality claimed by Becker et al *has been missing* as well.

Indeed, for all proofs the paper [22] refers to a preprint of [23], but the lists of the authors of [22, 23] are distinct, and their main algorithms and Main Theorems differ significantly.

Furthermore, the Boolean time estimates of the Main Theorem of [23] contain the term

$$\log(1/\mathrm{Discr}(p(x))),$$

with "Discr" standing for "Discriminant", such that

$$\log(1/\mathrm{Discr}(p(x/2^b))) = (d-1)db\log(1/\mathrm{Discr}(p(x))).$$

Subdivision iterations of the known root-finders for Problem 1_m, in particular, of those of [22, 23], scale the variable x up to $4m$ times by factors of up to 2^b. Hence the bit-complexity specified in the Main Theorem [23] has order of mbd^2 (see our further comments in Sec. 1.10).

Can one salvage the claim by Becker et al in another way?

This seems to be problematic: the algorithms of [22, 23] apply Pellet's celebrated theorem to more than $4m$ polynomials obtained from polynomial p by means of shifting the variable x, its scaling by up to 2^b, and root-squaring (see our Sec. 1.10).

Incorporation of Pellet's theorem into the subdivision root-finding process was the only major algorithmic novelty of [22, 23], but application of this theorem to a polynomial $t(x) = \sum_{i=0}^{d} t_i x^i$ relies on computing the values $|\frac{t_i}{t_d}|$ for $i = 0, 1, \dots, d-1$. Hence by virtue of Observation 1.1, the algorithms of [22, 23] involve Boolean time at least of order bd^2m for Problem 1_m. This exceeds the nearly optimal Boolean time bound of [83, 90] by a factor of m, that is, by a factor of d for Problem 1 where $m = d$.

In any case, devising polynomial root-finders that are both nearly optimal and practically promising had remained a challenging open problem in spite of the claim to the contrary by Becker et al. Furthermore, next we study a highly important root-finding problem not addressed by Becker et al.

1.8 Black Box Polynomial Root-finding and Factorization: motivation and computational problems

The known record fast algorithms for Problems 1, 1_m, and 2 as well as those of [114, 88, 22, 23] operate with the coefficients of p and cannot handle *black box polynomials* – given with an oracle (black box) for their evaluation.

Louis and Vempala studied this class in [68] motivated by the extension to *approximation of the eigenvalues of a matrix M* as the zeros of its characteristic polynomial $p(x) = \det xI - M$; this can be further reduced to matrix inversion via Jacobi formula for matrix derivative (see Remark 8.1).

For another motivation, one must use at least $2d$ arithmetic operations *(ops)* to evaluate a general polynomial p of (1.1) given with its $d+1$ coefficients (see [76] and the accounts of Strassen in [125, Section "Pan's method"] and Knuth in [61]), whereas $O(\log(d))$ ops are sufficient to evaluate it for a large and important class of dth degree polynomials, such as the Mandelbrot polynomials $p := p_k(x)$, where $p_0 := 1,\ p_1(x) := x,\ p_{i+1}(x) := xp_i^2(x) + 1,\ i = 0, 1, \ldots, k,\ d = 2^k - 1$, and the sums of a small number of shifted monomials, e.g., $p := \alpha(x-a)^d + \beta(x-b)^d + \gamma(x-c)^d$ for six constants a, b, c, α, β, and γ. Interpolation to such polynomials can *dramatically slow down fast polynomial root-finders* in a region containing a small number of zeros of p.

Even for a polynomial given with its coefficients the algorithms, black box computations avoid numerical problems and the growth of computational precision and Boolean complexity caused by shift and scaling of the variable (see Appendix B).

Such observations motivate the following modification of Problems 1, 1_m, and 2.

Problems 1bb, 1bb_m, and 2bb. Black Box Polynomial Root-finding and Factorization: Solve Problems 1, 1_m, and 2 in case where a polynomial p is given by a black box oracle (subroutine) for its evaluation rather than by its coefficients.[9]

1.9 Black Box Polynomial Root-finding: State of the Art

Motivated by the extension to approximation of an eigenvalue of an absolutely largest symmetric matrix, Louis and Vempala in [68] approximated within $\epsilon > 0$ an absolutely largest zero of a real-rooted black box polynomial p based on their novel variant of Newton's iterations.

The algorithm reduces this task to the evaluation of *Newton's ratio* $\mathrm{NR}(x) := \frac{p(x)}{p'(x)}$ at $O(b\log(d))$ points, for $b = \log(R/\epsilon)$ and $R = \max_j |z_j|$, and thus is said to perform at a *High Level Arithmetic cost* (hereafter *HLA cost*) $O(b\log(d))$.

Louis and Vempala also proved that their algorithm only requires computations with the precision of $O(\log(d))$ bits provided that one can access with infinite precision the values of $p(x)$ computed by the black box oracle. By combining their root-finder with Storjohan's Las Vegas randomized algorithm of [123] for computing the determinant of an integer matrix,[10] they approximated an absolutely largest eigenvalue of a symmetric matrix in a record expected Boolean time.

This admirable pioneering work, however, has made only a rudimentary step towards the solution of Problems 1bb and 1bb_m, leaving this as a research challenge with no progress since 2016 until preprint [99] appeared in 2018 (it was expanded to 139 pages by August 2022).

1.10 Our complexity estimates

Our novel algorithms solve Problem 1bb_m for all m from 1 to d; they match HLA cost bound of [68] where m is a constant and remove restrictions of [68] on the input and the output.

Instead of Newton's ratio $\mathrm{NR}(x)$ we evaluate *Newton's inverse ratio* $\mathrm{NIR}(x) := \frac{p'(x)}{p(x)}$.

The overall number of evaluation points x involved in our root-finders is said to be their *HLA cost.* Besides ops involved in the evaluation of NIR our root-finders perform some additional ops but less numerous than the former ops, even in the case where $p(x) = x^d$.

[9] One can evaluate p at $d_+ > d$ points and then interpolate to it, but this would destroy sparsity and would blow up the precision and the Boolean cost of the computations.

[10] Here it is also relevant to recall Gershgorin's bound for any eigenvalue λ of a matrix $M = (a_{i,j})_{i,j=1}^d$ [124, Thm. 1.3.2]: $|\lambda - a_{i,i}| \le \sum_{j \ne i} |a_{i,j}|$ for some i, $1 \le i \le d$. Clearly, this implies that $|\lambda| \le ||M||_F$.

Theorem 1.7. *Given a black box polynomial $p = p(x)$ of a degree d, a complex c, a triple of positive b, ρ, and θ such that $\theta - 1$ exceeds a positive constant, let each of the two discs $D(c, \rho)$ and $D(c, \theta\rho)$, both centered at c and having radii ρ and $\theta\rho$, respectively, contain precisely m zeros of p. Under these assumptions our root-finders, performing at HLA cost in $\tilde{O}((q(m)m)$ approximate within $R/2^b$, for $R := \rho + |c|$, all m zeros of p lying in these discs. Here $q(m) \leq m + 1$ for any polynomial $p(x)$ and $q(m) = 1$ under Las Vegas randomization (under which output error can be detected at a dominated computational cost) where the zeros of $p(x)$ are* independent identically distributed (iid) *random variables sampled from a fixed disc on the complex plane under the uniform probability distribution on that disc.*[11]

Louis and Vempala use precision of $O(\log(d))$ bits in their root-finder assuming infinite precision for both computations and cost-free access to the output of a black box oracle. By allowing Las Vegas randomized linear map of x we prove that precision bound even where we only access the output of the oracle with a relative error in $1/d^{\nu}$ for a constant ν of our choice and where we compute with rounding errors.

Namely, let $\mathbb{B}_d$ denotes the optimal Boolean cost of such evaluation of a polynomial p, which decreases by a factor of $d/\log(d)$ in transition from a general polynomial p to the Mandelbrot polynomial and a sum of $O(\log(d))$ shifted monomials. Then our analysis in Sec. 7 combined with Thm. 1.7 implies the solution of Problem 1bb_m in Boolean time

$$\mathbb{B}_{\text{roots}} = \tilde{O}(\mathbb{B}_d m q(m)) \tag{1.5}$$

for $q(m)$ of Thm. 1.7.

In Sec. 9 we combine our algorithms supporting the above estimates with Kirrinnis's [60, Thm. 3.9], which bounds the Boolean complexity of Fiduccia-Moenck-Borodin's algorithm of [37, 73] for multipoint polynomial evaluation (see [18, Sec. 4.5] for the history of that study).

This leads us to the following result.

Theorem 1.8. *Under the assumptions of Thm. 1.7 about a polynomial $p(x)$, discs $D(c, \rho)$ and $D(c, \theta\rho)$, integers m and $q(m)$, positive R, and real b, one can approximate within $R/2^b$, all m zeros of p lying in these discs in Boolean time $\tilde{O}((q(m)m + d)(q(m)m + d + b))$ for $q(m)$ of Thm. 1.7.*

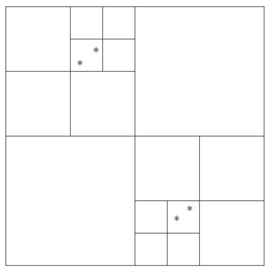
Remark 1.1. Optimality of our root-finding and factorization. *If $m^2 = \tilde{O}(d)$ or if m is unrestricted but if $q(m) = \tilde{O}(1)$, then $q(m)m = \tilde{O}(d)$. If in addition $d = \tilde{O}(b)$, then the bound of Thm. 1.8 turns into $\tilde{O}(bd)$. This is* below the lower bound $0.25(d-1)db$ *of Cor. 1.1, whcih can be applied to the Boolean cost of approximation of even a single zero of a general polynomial p of degree d, but that lower bound does not apply to our study where the polynomial $p(x)$ is not general: its m zeros lie in a disc well isolated from its external zeros; in this case we only have the lower bound $0.25b\max\{d + 1, (m-1)m\}$ of Cor. 1.2, and our upper bound reaches it, up to poly-logarithmic factor, provided that $q(m)m = \tilde{O}(d)$ and $d = \tilde{O}(b)$. The same comments apply to our next estimates for factorization Problem 2.*

By combining the estimate of Thm. 1.8 for $m = d$ with Cor. 1.4 for $d = O(b)$ we obtain the following Boolean time estimate for solving Problem 2 of factorization:

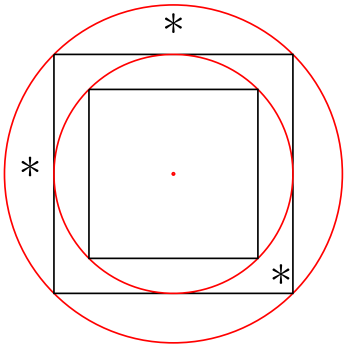
$$\mathbb{B}_{\text{factors}} = \tilde{O}((q(d)d + b)dq(d)) \text{ if } d = O(b). \tag{1.6}$$

By following [68] we extend our root-finder to *approximation of the eigenvalues of a matrix M*. Namely, combine Gershgorin's bound on matrix eigenvalues, our root-finders for $p(x) = \det(xI - M)$,

[11] A variant of our root-finders performing within the above cost bound for $q(m)$ of order $\log^2(d)$ has consistently output approximations to all d zeros of $p(x)$ within $1/2^b$ in extensive tests in [98, 99].



(a) Four roots marked by asterisks lie in suspect sub-squares; the other (empty) sub-squares are discarded.



(b) Both exclusion and $\sqrt{2}$-soft inclusion criteria hold for e/i test applied to the disc bounded by internal circle.

Figure 1

and Storjohann's algorithm of [123] for computing determinant to approximate all m eigenvalues of a $d \times d$ matrix, for $1 \le m \le d$, that lie in a fixed disc on the complex plane isolated from external eigenvalues (like zeros in Thm. 1.7). Deduce the following record bound on Las Vegas expected Boolean time for this problem,

$$\mathbb{B}_{\text{eigen}} = \tilde{O}(mq(m)\tilde{b}d^{\omega}) \tag{1.7}$$

where ω denotes an exponent of *feasible* or *unfeasible* matrix multiplication (MM),[12] $||M||_F$ denotes the Frobenius norm of a matrix M, and $\tilde{b} := \log(||M||_F/\epsilon)$. Then again, for a constant m we match the Boolean time bound of [68] under no restrictions of [68] on the input and the output.

1.11 Background and our technical novelties (very briefly)

To solve Problem 1bb$_m$ we forget about the algorithms of [68] and rely *distinct approach and completely distinct techniques.*

We depart from the classical subdivision root-finders, traced back to [134, 46, 114, 88], which divide every suspect square into four congruent sub-squares and to each apply *exclusion/inclusion (e/i) test*: a square is discarded if it definitely contains no roots and is called *suspect* and processed in the next iteration otherwise (see Fig. 1a).

Observation 1.3. *Let subdivision iterations be applied to a square with m zeros of p. Then every iteration (i) decreases by twice the diameter of a suspect square and (ii) processes at most $4m$ suspect squares, (iii) whose centers approximate all m zeros of p within the half-diameter of the squares.*

Corollary 1.5. *Subdivision iterations applied to a square with m roots and side length Δ approximate them within $\epsilon = \Delta/2^b$ after testing at most $4mk$ suspect squares for $k \le \lceil \log_2(\frac{\Delta}{\epsilon}\sqrt{2}) \rceil = \lceil b + 0.5 \rceil$.*

We follow [46] and instead of testing whether a suspect square contains zeros of p we test whether its covering disc does this. In Secs. 3 and 4 we complete classical subdivision root-finding by devising *σ-soft e/i tests* for a black box polynomial and $\sigma > 1$.

[12] The current records are about 2.7734 for exponents of feasible MM [77, 64], unbeaten since 1982, and about 2.372 for exponents of unfeasible MM [6, 33]. The exponents below 3.74 are only for unfeasible MM – all known algorithms that support them are slower than the classical $d \times d$ MM unless $\log_2(\log_2(\log_2(d)))) > 100$, say.

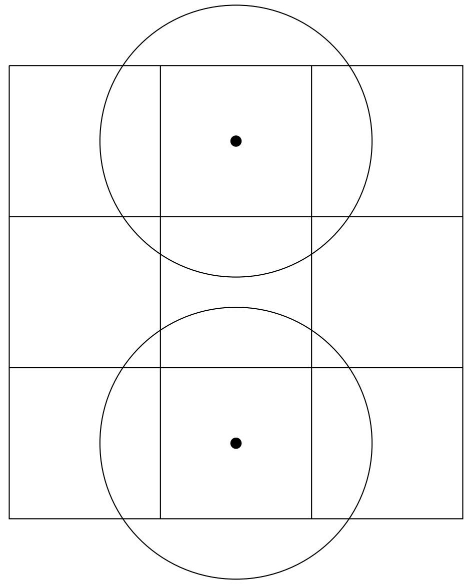


Figure 2: Two discs superscribing two non-adjacent suspect squares are not minimal but do not lie close to one another.

Such tests report *exclusion* and discard an input square S if they certify that its minimal covering disc $D(c, \rho) := \{x : \ |x - c| \le \rho\}$ contains no roots. The tests report σ*-soft inclusion*, and then we call the square S *suspect* and subdivide it, if for a fixed $\sigma > 1$ these tests certify that a little larger concentric disc $D(c, \sigma\rho)$, said to be the σ*-cover* of the square S, contains a root or roots.

Criteria for both exclusion and soft inclusion can hold simultaneously (see Fig. 1b), but we stop as soon as we verify any of them.

The estimate $4mk$ of Cor. 1.5 increases to $\alpha(\sigma)mk$ for $\alpha(\sigma) > 4$ bounded by a constant for a constant σ.

For $\sigma < \sqrt{2}$ the σ-covers of two non-adjacent suspect squares are separated by the fraction $a := 2 - \sigma\sqrt{2}$ of the side length (see Fig. 2).

Our techniques, in particular our use of randomization for root-finding, can be of independent interest.

As by-product we efficiently approximate the ℓ-th root radius, that is, the ℓ-th smallest distance to a zero of a black box polynomial from any fixed point on the complex plane.

1.12 Costly scaling for Pellet-based e/i tests

An e/i test based on Pellet's theorem for a polynomial $t(x) = \sum_{i=0}^{d} t_i x^i$ involves the values $|\frac{t_i}{t_d}|$ for $i = 1, \dots, d$, and hence, by virtue of Observation 1.2, must process at least $0.5(d-1)db$ bits of the coefficients of $t(x)$ to support approximation of its zeros within $1/2^b$.

The algorithms of [23] cannot avoid this because they apply e/i tests to the polynomials obtained in $\log(\log(d))$ steps of root-squaring of Sec. 2.2 from polynomials $t_{c,\rho}(x) = p(\frac{x-c}{\rho})$ where $\rho > 0$ can be as small as $1/2^b$.

One may need more than $4m$ e/i tests even in a single subdivision iteration; this means that one must involve more than $2(d-1)dmb$ bits and $(d-1)dmb$ bit-operations to solve Problem 1_m by applying such a Pellet-based root-finder.

1.13 Related works

The pioneering paper [68] proposed a novel algorithm for approximation of an absolutely largest zero of a real-rooted black box polynomial, estimated the computational precision and Boolean complexity of that algorithm, and extended it to approximation of an absolutely largest eigenvalue of a symmetric matrix. We adopt their model of computation and incorporate the five first lines of the proof of [68, Lemma 3.5] into our proof of the bound on the computational precision of our root-finders, but otherwise use completely distinct approach and techniques.

Further progress in that area is due to arXiv preprint [99] of 2018, expanded to 139 pages in August 2022, partly covered in [97, 51, 98, 67], and further advanced here, in its revision.

By following [99] we rely on classical subdivision root-finders, based on e/i tests and traced back to [134, 46, 114, 88]. All known e/i tests, however, operate with the coefficients of p: they involve higher order derivatives $p^{(i)}(x)$, $i = 1, 2, \dots, d$ [46, 114], Newton's identities and fast root-squaring algorithms based on convolution [88], or Pellet's theorem [23].

Neither of these approaches can work for black box polynomials. We overcome the problem by means of approximation of a Cauchy integral by finite sums. Such approximation was used by Schönhage in [120] for the distinct goal of deflation of a polynomial.

Estimating Boolean complexity of our root-finders we use [60, Thm. 3.9].

Our randomization techniques for black box polynomial root-finding and our acceleration of black box subdivision root-finding by means of nearly maximal compression of a disc on the complex plane without losing any zero $p(x)$ are novel, to the best of our knowledge. As by-product we efficiently approximate the ℓ-th root radius, that is, the ℓ-th smallest distance to a zero of a black box polynomial, from any fixed point on the complex plane. Then again, we are not aware of any previous algorithms for this well-known basic task of polynomial root-finding.

1.14 Implementation and testing

Our current algorithms are less involved than those of [83, 90], have good chances to become user's choice root-finders.[13] Preliminary versions of our entire subdivision root-finders have been implemented and tested in [51, 54]. Already incorporation in 2020 of some of our techniques into the previous best implementation of subdivision root-finder of [55] has enabled its noticeable acceleration (see [51]). The more advanced publicly available implementation in [54] "relies on novel ideas and techniques" from [99] (see [54, Sec. 1.2]) and also uses some earlier algorithms, which are slower by order of magnitude than our current ones but still performs *dramatically faster* than [55] or any other known variant of subdivision complex root-finder and, according to extensive tests with standard test polynomials, at least competes with user's choice package MPSolve even for approximation of all d zeros of a polynomial p given with its coefficients.[14] This was only a *dream for subdivision root-finders before 2022.*

Organization of the rest of the paper.

In Parts I – III we support the upper bound on the complexity of Problem 1_m, exceeding that of Thm. 1.8 by a factor of b. In Part I we do this only under the bound $q(m) \le m + 1$ based on approximation of a Cauchy integral associated with root-counting in a disc on the complex plane.

[13] [90] was very cautious about practical prospect of its algorithms, but since 2002 there was dramatic progress in devising subroutines for polynomial computations.

[14] The complexity of a subdivision root-finder is proportional to the number of roots in a region, while MPSolve is about as fast and slow for all roots as for their fixed subset. MPSolve implements Ehrlich-Aberth's root-finding iterations, which empirically converge to all d roots very fast right from the start but only under empirical support and only under initialization that operates with the coefficients of p [115].

In Part II we cover alternative derivation of the same complexity estimates based on lifting the zeros of p to a fixed power $h = O(\log(d))$. We also deduce the estimates for $q(m) = 1$ under random root model. We make presentation in Parts II independent of Part I by including again some definitions, figures, and theorems.

We devote Part III to Compression of a Disc and Acceleration of Subdivision Iterations

We cover Deflation in Part IV.

In Part V we study Polynomal Root-finding by means of Functional Iterations as well as Polynomial Factorization.

We devote Part VI to brief Conclusions.

In Appendices A and B we recall some Algorithms and Complexity Estimates from [120, 122] on Recursive Factorization of a polynomial with extensions to Root-finding and Root Isolation and on Newton's refinement of Splitting a Polynomial into the product of two factors, respectively.

We dramatically revised Parts 0, I, II, and III, deleted tghe former Appendix C and some sections from Parts IV and V, but have not touched Parts IV – VI and the Appendix otherwise. In particular, we keep denoting the zeros of $p(x)$ by x_j unlike Parts 0 –III, where we write z_j.

PART I: Subdivision with exclusion/inclusion tests based on approximation of Cauchy integrals

Organization of the presentation in Part I

We devote the next section to some basic definitions. In Sec. 3 we study root-counting in a disc on the complex plane by means of approximation of a Cauchy integral over its boundary circle by finite sums.[15] In Sec. 4 we apply this study to devise and analyze root-counters and e/i tests in a disc on the complex plane. In short Sec. 5 we discuss error detection and correction for polynomial root-finders. In Sec. 6 we extend our e/i tests to estimating root radii, that is, the distances from a fixed complex point to the roots. In Secs. 7 and 9 we estimate precision of computing of our root-finders and their Boolean complexity, respectively, in both cases assuming that $p(x)$ is general polynomial of (1.1) represented with its coefficients. We devote short Sec. 8 to extension of our root-finders to approximation of the eigenvalues of a matrix or a matrix polynomial.

2 Background: basic definitions

- $S(c, \rho) := \{x : |\Re(c - x)| \le \rho,$
  $|\Im(c - x)| \le \rho\}$, $D(c, \rho) := \{x : |x - c| \le \rho\}$,
  $C(c, \rho) := \{x : |x - c| = \rho\}$, and
  $A(c, \rho, \rho') := \{x : \rho \le |x - c| \le \rho'\}$
  denote a square, a disc, a circle (circumference), and an annulus on the complex plane, respectively.
- $\Delta(\mathbb{R})$, $X(\mathbb{R})$, $CH(\mathbb{R})$, and $\#(\mathbb{R})$ denote the diameter, the root set, the convex hull, and the index (root set's cardinality) of a region $\mathbb{R}$ on the complex plane, respectively.
- A disc $D(c, \rho)$, a circle $C(c, \rho)$, or a square $S(c, \rho)$ is said to be *θ-isolated* for $\theta > 1$ if $X(D(c, \rho)) = X(D(c, \theta\rho))$, $X(C(c, \rho)) = X(A(c, \rho/\theta, \rho/\theta))$, or $X(S(c, \rho)) = X(S(c, \theta\rho))$, respectively.

[15] We do not use the material of Sec. 3.5, but it should be of independent interest.

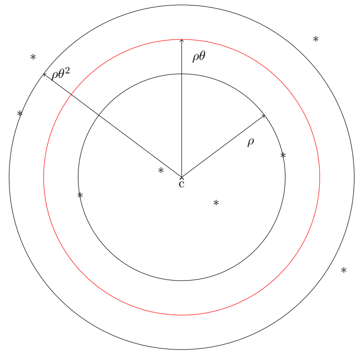


Figure 3: The roots are marked by asterisks. The red circle $C(c, \rho\theta)$ and the disc $D(c, \rho\theta)$ have isolation θ. The disc $D(c, \rho)$ has isolation θ^2.

- The largest upper bound on such a value θ is said to be the *isolation* of the disc $D(c, \rho)$, the circle $C(c, \rho)$, or the square $S(c, \rho)$, respectively (see Fig. 3), and is denoted $i(D(c, \rho))$, $i(C(c, \rho))$, and $i(S(c, \rho))$, respectively.

- $r_1(c, t) = |y_1 - c|, \dots, r_d(c, t) = |y_d - c|$ in non-increasing order are the d *root radii*, that is, the distances from a complex center c to the zeros $y_1 \dots, y_d$ of a dth degree polynomial $t(x)$. $r_j(c) := r_j(c, p)$, $r_j := r_j(0)$ for $j = 1, \dots, d$.

- Define *Newton's inverse ratio* $\mathrm{NIR}(x) := \frac{p'(x)}{p(x)}$. Differentiate factorization (1.1) of $p(x)$ to express $\mathrm{NIR}(x)$ as follows:
$$\mathrm{NIR}(x) = \frac{p'(x)}{p(x)} = \sum_{j=1}^{d} \frac{1}{x - z_j}. \tag{2.1}$$

- Generalize e/i tests as follows: For $p(x)$ of (1.1), $\sigma > 1$, and integers ℓ and m such that $1 \le \ell \le m \le d$ and $\#(D(0, 1)) \le m$, a σ-**soft** ℓ-**test**, or just ℓ-**test** for short (1-test being e/i test), either outputs 1 and stops if it detects that $r_{d-\ell+1} \le \sigma$, that is, $\#(D(0, \sigma)) \ge \ell$, or outputs 0 and stops if it detects that $r_{d-\ell+1} > 1$, that is,[16] $\#(D(0, 1)) < \ell$. *ℓ-test$_{c,\rho}$*, aka *ℓ-test for the disc $D(c, \rho)$*, is an ℓ-test applied to the polynomial $t(y)$ of (2.2).

- By properly shifting and scaling the variable x define the linear map of the disc $D(c, \rho)$ into the unit disc $D(0, 1)$:
$$x \mapsto \frac{x - c}{\rho},\ D(c, \rho) \mapsto D(0, 1),\ \text{and } p(x) \mapsto t(x) = p\Big(\frac{x - c}{\rho}\Big). \tag{2.2}$$

Observation 2.1. *(2.2) maps the zeros z_j of $p(x)$ into the zeros $y_j = \frac{z_j - c}{\rho}$ of $t(y)$, for $j = 1, \dots, d$, and preserves the index $\#(D(c, \rho))$ and the isolation $i(D(c, \rho))$.*

[16] Bounds $r_{d-\ell+1} \le \sigma$ and $r_{d-\ell+1} > 1$ can hold simultaneously (cf. Fig. 1a), but as soon as an ℓ-test verifies any of them, it stops without checking if the other bound also holds.

3 Cauchy root-counting in a disc

3.1 Root-counting in the unit disc

Recall that the power sums of the zeros of $p(x)$ in a complex domain $\mathcal{D}$ is given by a Cauchy integral over its boundary contour $\mathcal{C}$ (cf. [1]):

$$s_h = s_h(\mathcal{D}) := \sum_{j=1}^{d} z_j^h = \frac{1}{2\pi\sqrt{-1}} \int_{\mathcal{C}} \frac{p'(x)}{p(x)}\ dx,\ \ h = 0, , \dots . \tag{3.1}$$

We are particularly interested in computing the 0-th power sum $s_0 = \#(\mathcal{D})$; We can just approximate it within less than $1/2$ and then obtain it by means of rounding.

For $\mathcal{D} = D(0,1)$ we fix a positive integer q and approximate s_h with *Cauchy sum*

$$s_{h,q} := \frac{1}{q} \sum_{g=0}^{q-1} \zeta^{(h+1)g}\ \frac{p'(\zeta^g)}{p(\zeta^g)} \text{ for } \zeta \text{ of (1.3).} \tag{3.2}$$

Alternatively, one can apply Exponentially Convergent Adaptive Trapezoidal Quadrature [131] efficiently implemented in Boost C++ libraries.

3.2 Cauchy sums as weighted power sums of the roots

Cor. 3.1 from the following lemma expresses the hth Cauchy sum $s_{h,q}$ as the sum of the hth powers of the roots z_j with the weights $\frac{1}{1-x^q}$.

Lemma 3.1. *For a complex z, two integers $h \ge 0$ and $q > 1$, and ζ of (1.3), it holds that*

$$\frac{1}{q} \sum_{g=0}^{q-1} \frac{\zeta^{(h+1)g}}{\zeta^g - z} = \frac{z^h}{1 - z^q}. \tag{3.3}$$

Proof. First let $|z| < 1$ and obtain

$$\frac{\zeta^{(h+1)g}}{\zeta^g - z} = \frac{\zeta^{hg}}{1 - \frac{z}{\zeta^g}} = \zeta^{hg} \sum_{u=0}^{\infty} \left(\frac{z}{\zeta^g}\right)^u = \sum_{u=0}^{\infty} \frac{z^u}{\zeta^{(u-h)g}}.$$

The equation in the middle follows from Newman's expansion

$$\frac{1}{1-y} = \sum_{u=0}^{\infty} y^u \text{ for } y = \frac{z}{\zeta^g}.$$

We can apply it because $|y| = |z|$ while $|z| < 1$ by assumption.

Sum the above expressions in g, recall that

$$\zeta^u \ne 1 \text{ and } \sum_{s=0}^{q-1} \zeta^u = 0 \text{ for } 0 < u < q,\ \zeta^q = 1, \tag{3.4}$$

and deduce that

$$\frac{1}{q} \sum_{g=0}^{q-1} \frac{z^u}{\zeta^{(u-h)g}} = z^u$$

for $u = h + ql$ and an integer l, while $\sum_{g=0}^{q-1} \frac{z^u}{\zeta^{(u-h)g}} = 0$ for all other values u. Therefore

$$\frac{1}{q}\sum_{g=0}^{q-1}\frac{\zeta^{(h+1)g}}{\zeta^g - z} = z^h \sum_{l=0}^{\infty} z^{ql}.$$

Apply Newman's expansion for $y = z^q$ and deduce (3.3) for $|z| < 1$. By continuity extend (3.3) to the case of $|z| = 1$.

Now let $|z| > 1$. Then

$$\frac{\zeta^{(h+1)g}}{\zeta^g - z} = -\frac{\zeta^{(h+1)g}}{z}\ \frac{1}{1-\frac{\zeta^g}{z}} = -\frac{\zeta^{(h+1)g}}{z}\sum_{u=0}^{\infty}\left(\frac{\zeta^g}{z}\right)^u = -\sum_{u=0}^{\infty}\frac{\zeta^{(u+h+1)g}}{z^{u+1}}.$$

Sum these expressions in g, write $u := ql - h - 1$, apply (3.4), and obtain

$$\frac{1}{q}\sum_{g=0}^{q-1}\frac{\zeta^{(h+1)g}}{\zeta^g - z} = -\sum_{l=1}^{\infty}\frac{1}{z^{ql-h}} = -z^{h-q}\sum_{l=0}^{\infty}\frac{1}{z^{ql}}.$$

Apply Newman's expansion for $y = z^q$ again and obtain

$$\frac{1}{q}\sum_{g=0}^{q-1}\frac{\zeta^{(h+1)g}}{\zeta^g - z} = -\frac{z^{h-q}}{1-\frac{1}{z^q}} = \frac{z^h}{1-z^q}.$$

Hence (3.3) holds in the case where $|z| > 1$ as well. $\square$

Sum equations (3.3) for $z = z_j$ and $j = 1, \ldots, d$, combine Eqns. 2.1 and (3.2), and obtain

Corollary 3.1. *For the zeros z_j of $p(x)$ and all h, the Cauchy sums $s_{h,q}$ of (3.2) satisfy* $s_{h,q} = \sum_{j=1}^{d}\frac{z_j^h}{1-z_j^q}$ *unless $z_j^q = 1$ for some j.*

3.3 Approximation of the power sums in the unit disc

Cor. 3.1 implies that the values $|s_{h,q-}s_h|$ decrease exponentially in $q - h$ if the unit disc $D(0,1)$ is isolated.

Theorem 3.1. *[120, Eqn. (12.10)].*[17] *Let the unit disc $D(0,1)$ be θ-isolated and let $d_{\rm in}$ and $d_{\rm out} = d - d_{\rm in}$ denote the numbers of the roots lying in and outside the disc $D(0,1)$, respectively. Then*

$$|s_{h,q} - s_h| \le \frac{d_{\rm in}\theta^{-h} + d_{\rm out}\theta^h}{\theta^q - 1} \le \frac{d\theta^h}{\theta^q - 1} \text{ for } h = 0, 1, \ldots, q-1. \tag{3.5}$$

Proof. Combine the expression $s_h = \sum_{j=1}^{d} z_j^h$ with Cor. 3.1 and obtain

$$s_{h,q} - s_h = \sum_{j=1}^{d}\frac{z_j^{q+h}}{1-z_j^q} = s_{h,q,{\rm in}} + s_{h,q,{\rm out}}$$

where

$$s_{h,q,{\rm in}} = \sum_{j=1}^{d_{\rm in}}\frac{z_j^{q+h}}{1-z_j^q} \text{ and } s_{h,q,{\rm out}} = \sum_{j=d_{\rm in}+1}^{d}\frac{z_j^h}{1-z_j^q}.$$

[17] Unlike the proof in [120], we rely on Cor. 3.1 of independent interest.

To prove bound (3.5) combine these equations with the bounds

$$|z_j| \le \eta \text{ for } j \le d_{\rm in} \text{ and } \frac{1}{|z_j|} \le \eta \text{ for } d_{\rm in} < j \le d.$$

□

By choosing q that exceeds $\log_\theta(2d+1)$ we ensure that $|s_0 - s_{0,q}| < 1/2$, and then obtain s_0 by rounding $s_{0,q}$ to the nearest integer.

Example 3.1. *Let the unit circle $C(0,1)$ be 2-isolated. Then Thm. 3.1 implies that $|s_{0,q}-s_0| < 1/2$ for $d \le 1,000$ if $q = 10$ and for $d \le 1,000,000$ if $q = 20$.*

> **Algorithm 3.1.** *INPUT: a black box polynomial p of a degree d and $\theta > 1$.*
>
> *INITIALIZATION: Compute the integer $q = \lfloor \log_\theta(4d+2) \rfloor > \log_\theta(2d+1)$.*
>
> *COMPUTATIONS: Compute Cauchy sum $s_{0,q}$ and output an integer $\bar{s}_0$ closest to it.*

Observation 3.1. *(i) Alg. 3.1 runs at HLA cost $q = \lfloor \log_\theta(4d+2) \rfloor$. (ii) It outputs $\bar{s}_0 = \#(D(0,1))$ if the circle $C(0,1)$ is θ-isolated for $\theta > 1$. (iii) If the algorithm outputs $\bar{s}_0 > 0$, then $\#(D(0,\theta)) > 0$.*

Proof. Thm. 3.1 immediately implies claim (ii) but also implies that $\#(D(0,1)) > 0$ if $\bar{s}_0 > 0$ unless $\#(A(0,1/\theta,\theta)) > 0$. In both cases $\#(D(0,\theta)) > 0$. □

3.4 Extension to any disc

Map (2.2) enables us to extend definition (3.2) of Cauchy sum to any disc $D(c,\rho)$ as follows:

$$s_{0,q}(p,c,\rho) := s_{0,q}(t,0,1) := \frac{\rho}{q} \sum_{g=0}^{q-1} \zeta^g \, \frac{p'(c+\rho\zeta^g)}{p(c+\rho\zeta^g)}, \tag{3.6}$$

for ζ of (1.3), that is, $s_{h,q}(p,c,\rho)$ is the Cauchy sums $s_{h,q}(t,0,1)$ of the zeros of the polynomial $t(y) = p(\frac{y-c}{\rho})$ in the unit disc $D(0,1)$.

Observation 3.2. *Given a polynomial $p(x)$, a complex c, a positive ρ, a positive integer q, and the q-th roots of unity, evaluation of $s_{0,q}(p,c,\rho)$ can be reduced to the evaluation of $NIR(\zeta^g)$ for $g = 0,1,\dots,q-1$ at HLA cost q and in addition performing $q+1$ divisions, $2q-1$ multiplications, and $2q-1$ additions.*

Refer to Alg. 3.1 applied to the polynomial $t(y)$ of (2.2) for an isolation parameter $\theta > 1$ as *Alg. 3.1$_{c,\rho,\theta}$* and also as *Alg. 3.1 applied to θ-isolated circle $C(c,\rho)$*. Combine Observations 2.1 and 3.2 to extend Observation 3.1 as follows.

Theorem 3.2. *(i) Alg. 3.1$_{c,\rho,\theta}$ runs at HLA cost $q = \lfloor \log_\theta(4d+2) \rfloor$ and in addition performs $5q-2$ ops. (ii) It outputs $\bar{s}_0 = \#(D(c,\rho))$ if the circle $C(c,\rho)$ is θ-isolated. (iii) If the algorithm outputs $\bar{s}_0 > 0$, then $\#(D(c,\theta\rho)) > 0$.*

By applying Thm. 3.1 we can estimate the errors $|s_{h,q}(t,0,1) - s_h(t,0,1)|$ of the approximation of the hth power sums $s_h(t,0,1)$ of the roots of $t(x)$ in the unit disc $D(0,1)$. Next we specify this for $h = 1$.

Theorem 3.3. *(Cf. [54].) Let for $\theta > 1$ a θ-isolated disc $D(c,\rho)$ contain exactly m roots. Fix a positive integer q and define the Cauchy sum $s_{1,q}(p,c,\rho)$ of (3.6). Then*

$$|mc + \rho s_{1,q}(p,c,\rho) - s_1(p,c,\rho)| \le \frac{\rho d\theta}{\theta^q - 1}.$$

3.5 The real part of the 0-th Cauchy sum

Clearly, $|\Re(s_{0,q})-s_0|\le|s_{0,q}-s_0|$ because s_0 is an integer. Next we estimate the bound $|\Re(s_{0,q})-s_0|$ for the unit disc $D(0,1)$, point out some resulting directions to algorithmic gain for root-finding, but leave further elaboration as a challenge for formal and empirical investigation.

Recall that $s_{0,q}=\sum_{j=1}^d\frac{1}{1-z_j^q}$ unless $z_j^q=1$ for some q (cf. Cor. 3.1), write $z:=z_j^q$ and estimate the contribution of a roots z_j to $\Re(s_{0,q})$.

Theorem 3.4. *Let $z=\rho\cdot(\cos(\phi)+\sin(\phi)\mathbf{i})$ for a non-negative ρ and a real $\phi\in[0,2\pi)$. Then*

$$\Re\Big(\frac{1}{1-z}\Big)=\frac{1-\rho\cos(\phi)}{1+\rho^2-2\rho\cos(\phi)},\tag{3.7}$$

$$1-\Re\Big(\frac{1}{1-z}\Big)=\rho\frac{\rho-\cos(\phi)}{1+\rho^2-2\rho\cos(\phi)}.\tag{3.8}$$

Proof. Multiply both numerator and denominator of the fraction

$$\frac{1}{1-z}=\frac{1}{1-\rho\cdot(\cos(\phi)+\sin(\phi)\mathbf{i})}$$

by $1-\rho\cdot(\cos(\phi)-\sin(\phi)\mathbf{i})$, substitute $\cos^2(\phi)+\sin^2(\phi)=1$, and obtain (3.7). Then readily verify (3.8). ☐

Corollary 3.2. *Under the assumptions of Thm. 3.4 it holds that*

$$\Re\Big(\frac{1}{1-z}\Big)\ge 0 \text{ if } \rho\le 1,\tag{3.9}$$

$$\Re\Big(\frac{1}{1-z}\Big)=\frac{1}{2} \text{ if } \rho=1 \text{ and if } \phi\ne 0,\tag{3.10}$$

$$\Re\Big(\frac{1}{1-z}\Big)\ge\frac{1-0.8\rho}{1+\rho^2-1.6\rho} \text{ if } \rho\le 5/4 \text{ and if } \cos(\phi)\le 0.8,\tag{3.11}$$

$$\Re\Big(\frac{1}{1-z}\Big)=\frac{1}{1-\rho} \text{ if } \phi=0 \text{ and if } \rho\ne 1,\tag{3.12}$$

$$\frac{1}{1+\rho}\le\Big|\Re\Big(\frac{1}{1-z}\Big)\Big|\le\frac{1}{|1-\rho|},\tag{3.13}$$

$$\frac{\rho}{1+\rho}\le\Big|1-\Re\Big(\frac{1}{1-z}\Big)\Big|\le\frac{\rho}{|1-\rho|}.\tag{3.14}$$

Proof. Readily deduce (3.9), (3.10), and (3.12) from (3.7).

Write

$$y:=\cos(\phi),\ f(y):=\Re\Big(\frac{1}{1-z}\Big),\ \text{and } g(y):=\frac{1}{f(y)}.$$

Then deduce from (3.7) that $g(y)=2+\frac{\rho^2-1}{1-\rho y}$. Hence $g'(y)=\frac{(\rho^2-1)\rho}{(1-\rho y)^2}$ is a monotone increasing or monotone decreasing function, singular if and only if $y=\frac{1}{\rho}$. Extend these properties to $f(y)$ and $1-f(y)$ if $\rho<1$. This implies (3.11), (3.13), and (3.14) provided that $\rho<1$ because

$$f(1)=\frac{1}{1-\rho},\ 1-f(1)=\frac{\rho}{1-\rho},\ f(-1)=\frac{1}{|1+\rho|},\ 1-f(-1)=\frac{\rho}{|1+\rho|},$$

and $f(0.8)=\frac{1-0.8\rho}{1+\rho^2-1.6\rho}$.

The proof can be extended to the case where $\rho \geq 1$ except that the zero $y=\frac{1}{\rho}$ of $f(y)$ turns into a pole of $\frac{1}{f(y)}$. We complete the proof of the bounds (3.13) for all positive ρ by observing that $f(\frac{1}{\rho})$ is undefined for $\rho=1$ and that otherwise $f(\frac{1}{\rho})=0$ and $1-f(\frac{1}{\rho})=1$. $\square$

Next we outline but do not elaborate upon some potential links of Cor. 3.2 to root-counting. Let the unit disc $D(0,1)$ be θ-isolated, combine bounds (3.13) and (3.14), and obtain

$$|\Re(s_{0,q})-s_0| \leq \frac{d}{\theta^q-1}.$$

The bound is similar to (3.5) for $h=0$ but is overly pessimistic: it is only reached where $\cos(\phi)=1$ for $z=z_j$ and all j.

Equation (3.12) implies numerically unstable dependence of the fraction $\Re(\frac{1}{1-z})$ on ρ where $\cos(\phi)\approx 1\approx \rho$, but bound (3.11) shows that outside a rather small neighborhood of these extreme values of $\cos(\phi)$ and ρ the fraction becomes smooth and approaches $1/2$ as $\rho \to 1$ and/or $\cos(\phi)\to -1$.

Remark 3.1. *In view of bound (3.11) random rotation of the circle $C(0,1)$ is a heuristic recipe for countering instability in z of the fraction $\Re(\frac{1}{1-z})$ because $\cos(\phi)>0.8$ for only one of four equally-spaced points ϕ on the unit circle $C(0,1)$ and hence with a probability less than 1/4 for a random choice of ϕ under the uniform probability distribution in the range $[0,2\pi)$. Moreover, random rotation of the circle can limit the adverse and numerically unstable impact on an e/i test by boundary roots – lying on or near the boundary circle $C(0,1)$.*

We conclude with a simple but surprising result on root-counting on an isolated circle.

Corollary 3.3. *Suppose that $p(1)\neq 0$ and that $p(x)$ has precisely m roots in an annulus $A(0,\frac{1}{\theta},\theta)$ for a fixed $\theta>1$ and that all of them lie on the unit circle $C(0,1)$. Let s' and s'' denote the 0th Cauchy sums for the discs $D(0,\frac{1}{\theta})$ and $D(0,\theta)$, respectively, and write $s_{0,q}=s''-s'$. Then $\#(C(0,1))=2\Re(s_{0,q})$.*

Proof. Combine Cor. 3.1 for $h=0$ with equation (3.10). $\square$

3.6 The power sums and Newton's identities

If a polynomial $p(x)$ of (1.1) is monic, e.g., has been made monic by scaling, then the power sums of all its d roots can be linked to the coefficients through the linear system of Newton's identities (cf. [89, equations (2.5.4) and (2.5.5)]):

$$\begin{aligned} s_1+p_{d-1}&=0,\\ p_{d-1}s_1+s_2+2p_{d-2}&=0,\\ p_{d-2}s_1+p_{d-1}s_2+s_3+3p_{d-3}&=0,\\ &\cdots\cdots\cdots\cdots\cdots\cdots \end{aligned}$$

$$s_i+\sum_{j=1}^{i-1}p_{d-j}s_{i-j}=-ip_{d-i},\ i=1,\dots,d, \tag{3.15}$$

and for any $K>d$ it holds that

$$\sum_{j=0}^{d}p_{d-j}s_{d+i-j}=0,\ i=1,\dots,K-d. \tag{3.16}$$

Algorithm 3.2. From the coefficients to the power sums. *Given the coefficients of a polynomial p of (1.1), compute the power sums s_h of its roots for $h = 1, 2, \dots, d$ by solving a triangular Toeplitz linear system, which is equivalent to computing the reciprocal of a polynomial in x modulo x^d [89, Sec. 2.5].*

Conversely, given the power sums, we can compute the coefficients by solving a triangular linear system defined by the same identities or by applying Newton's iterations (see Algs. 21.3 and 21.4).

4 Root-counting and ℓ-tests

4.1 Overview

We seek fast root-counters and ℓ-tests based on computing Cauchy sum $s_{0,q}$ for a smaller integer q, but if we go too far the output can deceive us: the polynomial $p(x) := (1 - wx^{d-m})(x^m - dx + w)$ has zero near the origin if $w \approx 0$, while $\mathrm{NIR}(x) = p'(x)/p(x)$ vanishes for x equal to any $(d-1)$st root of unity.

Such deceptive e/i tests have been extremely rare guests in the extensive experiments in [51, 54] for q of order $\log^2(d)$) and were completely excluded where the values $|s_{h,q}|$ were required to be small for $h = 0, 1, 2$, but formal support for this empirical behavior is still a challenge. In the next two subsections we formally support root-counters and ℓ-tests running at randomized HLA cost in $O(m \log(d))$. We rely on sampling the radius of an input circle at random in a fixed range and conjecture that we would obtain similar results by computing the Cauchy sum $s_{0,q}$ for the polynomial $p(ax)$ for random $a \in C(0,1)$.

Randomization in our and all root-finders is of Las Vegas type (see Sec. 5). Besides, we apply randomization in Sec. 7 anyway – to bound computational precision in our root-finders.

For the sake of completeness, however, we present a deterministic ℓ-test in Sec. 5.1.

4.2 A basic randomized root-counter

Next we avoid adverse impact whp for any input polynomial p and thus devise faster ℓ-test$_{c,\rho}$ by applying Las Vegas randomization: Fix the center c of a circle $C(c, \rho)$ but choose its radius ρ at random in a small range and then prove that whp isolation $\theta := i(C(c, \rho))$ sufficiently exceeds 1 to keep the boundary circle away from the zeros of p whp.

Accordingly we modify Alg. 3.1$_{c,\rho}$: we initialize it by choosing a positive integer q, thus arriving at *Alg. 3.1*$^{(q)}_{c,\rho}$. Wlog let $c = 0$ and compute $\#(D(0, \rho))$ for a random ρ in the range $[2^{0.2}, 2^{0.4}]$, devising a root-counter in such a random disc $D(0, \rho)$. This can serve as a σ-soft ℓ-test for the disc $D(0, 2^{0.2})$ for $\sigma = 2^{0.2}$: indeed, if $\#(D(0, \rho)) < \ell$, then $\#(D(0, 2^{0.2}) < \ell$; otherwise $\#(D(0, 2^{0.4})) \geq \ell$.

Algorithm 4.1. Basic randomized root-counter.

> *INPUT: $\gamma \geq 1$ and a d-th degree black box polynomial p having at most m zeros in the disc $D(0, \sqrt{2})$.*
>
> *INITIALIZATION: Sample a random value w in the range $[0.2, 0.4]$ under the uniform probability distribution in that range and write $\rho := 2^w$.*
>
> *COMPUTATIONS: Apply Alg. 3.1$^{(q)}_{0,\rho}$ for $q = \lfloor 10m\gamma \log_2(4d + 2) \rfloor$.*

Theorem 4.1. *Alg. 4.1 runs at HLA cost $q = \lfloor 10m\gamma \log_2(4d + 2) \rfloor$ and outputs $\bar{s}_0 = \#(D(0, \rho))$ with a probability at least $1 - 1/\gamma$.*

Proof. The HLA cost is as claimed by definition.
By assumption, the annulus $A(0, 1, \sqrt{2})$ contains at most m roots.
Hence at most m root radii $r_j = 2^{e_j}$, for $j = 1, \dots, m' \le m$, lie in the range $[1, \sqrt{2}]$, that is, $0 \le e_j \le 0.5$ for at most $m' \le m$ integers j.
Fix m' intervals, centered at e_j, for $j = 1, \dots, m'$, each of a length at most $1/(5m'\gamma)$. Their overall length is at most $1/(5\gamma)$. Let $\mathbb{U}$ denote their union.
Sample a random u in the range $[0.2, 0.4]$ under the uniform probability distribution in that range and notice that Probability$(u \in \mathbb{U}) \le 1/\gamma$.
Hence with a probability at least $1 - 1/\gamma$ the circle $C(0, \rho)$ is θ-isolated for $\theta = 2^{\frac{1}{10m\gamma}}$, in which case Alg. 4.1 outputs $\bar{s}_0 = s_0 = \#(D(0, \rho))$ by virtue of claim (ii) of Thm. 3.2. □

4.3 Refining the bound on error probability

Next reapply Alg. 4.1 v times. Clearly, this increases its HLA cost v times but also decreases the error probability dramatically – below $1/\gamma^v$.

Next we will prove this for $\ell = 1$ and $\ell = m$. First again reduce the root radius approximation problem to the decision problem of ℓ-test. Namely, *narrow Alg. 4.1 to an ℓ-test* for a fixed ℓ in the range $1 \le \ell \le m$, so that with a probability at least $1 - 1/\gamma$ this ℓ-test outputs 0 if $\#(D(0, 1)) < \ell$ and outputs 1 if $\#(D(0, 2^{0.5})) \ge \ell$. Moreover, output 1 is certified by virtue of claim (iii) of Thm. 3.2 if $\ell = 1$; similarly the output 0 is certified if $\ell = m$.

Next extend the ℓ-test by means of applying Alg. 4.1 for v iid random variables w in the range $0.2 \le w \le 0.4$. Call this ℓ-test **Alg. 4.1v,ℓ**.

Specify its output g for $\ell = 1$ as follows: let $g = 1$ if Alg. 4.1$v, 1$ outputs 1 at least once in its v applications; otherwise let $g = 0$. Likewise, let $g = 0$ if Alg. 4.1v, m outputs 0 at least once in its v applications; otherwise let $g = 1$.

Recall Thm. 4.1 and then readily verify the following theorem.

Theorem 4.2. *(i) For two integers $1 \le \ell \le m$ and $v \ge 1$ and a real $\gamma \ge 1$, Alg. 4.1v,ℓ runs at HLA cost $\lfloor 10m\gamma \log_2(4d + 2)\rfloor v$, which dominates the cost of the other ops involved, even in the cases of the Mandelbrot and sparse polynomials p (cf. Observation 3.2). (ii) Output 1 of Alg. 4.1v,1 and the output 0 of Alg. 4.1v,m are certified. (iii) Output 0 of Alg. 4.1v,1 and output 1 of Alg. 4.1v,m are correct with a probability at least $1 - 1/\gamma^v$.*

The theorem bounds the cost of ℓ-tests for $\ell = 1$ (e/i tests) and $\ell = m$. This is sufficient for our root-finding, but our next natural extension covers ℓ-tests for any integer ℓ in the range $1 < \ell < m$.

Assign the values 0 or 1 to the output integers g as follows: compute the average of the v outputs in v independent applications of the test and output g denoting the integer closest to that average; if the average is $1/2$, write $g = 0$ to break ties.

This extension of Alg. 4.1v,ℓ is only interesting for $1 < \ell < m$; otherwise its cost/error estimates are inferior to those of Thm. 4.2.

Theorem 4.3. *For an integer ℓ such that $1 < \ell < m \le d$ and real $v \ge 1$ and $\gamma > 1$, Algs. 4.1v,ℓ runs at HLA cost $\lfloor 10m\gamma \log_2(4d + 2)\rfloor v$, and its output is correct with a probability at least $1 - (4/\gamma)^{v/2}$.*

Proof. Since $D(0, 1) \subset D(0, \sqrt{2})$, we only consider the following three cases:
(i) $\#(D(0, 1)) < \ell$ and $\#(D(0, 2^{0.5})) \ge \ell$,
(ii) $\#(D(0, 2^{0.5})) < \ell$, and then also $\#(D(0, 1)) < \ell$,
(iii) $\#(D(0, 1)) \ge \ell$, and then also $\#(D(0, 2^{0.5})) \ge \ell$.

In case (i) the output g is always correct.

In case (ii) the output 0 is correct, while the output 1 is wrong and occurs if and only if more than $v/2$ tests go wrong, by outputting 1.

Likewise, in case (iii) the output 10 is correct, while the output 0 occurs if and only if more than $v/2$ tests go wrong, by outputting 0.

In both cases (ii) and (iii) we deal with the binomial distribution having failure probability $1/\gamma$, and so (cf. [24]) the probability P of at least $v/2$ failures among v tests is at most $\sum_{j=\bar{v}}^{v}\begin{pmatrix} v \\ j \end{pmatrix} T_j$ for $T_j = \gamma^{-j}(1-1/\gamma)^{v-j} \le \gamma^{-j}$.

Therefore, $P \le \sum_{j=0}^{v}\begin{pmatrix} v \\ j \end{pmatrix} = T_{\bar{v}}2^v \max_j T_j$. Substitute $\sum_{j=0}^{v}\begin{pmatrix} v \\ j \end{pmatrix} = 2^v$ and $\bar{v} = \lceil v/2 \rceil$ and obtain $P < 2^v T_{\bar{v}} \le 2^v/\gamma^{v/2} = (4/\gamma)^{v/2}$. □

5 Error detection and correction

A randomized root-finder can lose some roots, although with a low probability, but we can detect such a loss at the end of root-finding process, simply by observing that among the m roots in an input disc only $m-w$ *tame* roots have been closely approximated,[18] while $w>0$ *wild* roots remain at large. Then we can recursively apply the same or another root-finder until we approximate all m roots.[19] Next we estimate the cost and error probability of this recursive process.

Theorem 5.1. *Let the above recipe for error detection and correction be applied to the output of a basic randomized root-finder performing at the cost α (under a fixed model) and correct with a probability $1/\beta$ for $\beta > 1$ and let the basic algorithm be reapplied, each time with error detection at cost δ. Then in v recursive applications of the algorithm, its output is certified to be correct at the cost less than $(\alpha+\delta)v$ with a probability at least $1-1/\beta^v$.*

Having $d-w$ tame roots $z_{w+1}, z_{w+1}, \dots, z_d$ approximated, we can deflate the factor $q(x) = \prod_{j=1}^{w}(x-z_j)$, e.g., by applying evaluation-interpolation techniques [129], but this can blow up the coefficient length and destroy numerical stability unless w is a sufficiently small integer.

Working with a black box polynomial, however, we have an option of implicit deflation of *the wild factor* of p of any degree w, whose root set is made up of the w wild roots $z_1, \dots, z_w$: we can apply the same or another root-finder to the quotient polynomial $q(x) = p(x)/\prod_{j=w+1}^{d}(x-z_j)$, without computing the coefficients of this quotient. We can compute the ratio $\frac{q'(x)}{q(x)}$ by applying the following expression: $\frac{q'(x)}{q(x)} = \frac{p'(x)}{p(x)} - \sum_{j=1}^{d-w}\frac{1}{x-z_j}$.

5.1 Deterministic ℓ-test

In view of our remark at the end of Sec. 4.1, randomization is only a nominal restriction for ℓ-tests, but next we get rid of it, although at the price of increasing the HLA and Boolean cost by a factor of m.

Any output $\bar{s}_0 > 0$ of Alg. $3.1_{c,\rho}$ certifies soft inclusion by virtue of claim (iii) of Thm. 3.2, but the output $\bar{s}_0 = 0$ does not certify exclusion because some unknown roots can lie on or near the

[18]We call a complex c a tame root for an error tolerance ϵ if c lies in an isolated disc $D(c,\epsilon)$, whose index $\#(D(c,\epsilon))$ we can readily compute (cf. Thm. 3.1).

[19]This recipe detects output errors of any root-finder at the very end of computations, but for subdivision root-finders one can detect the loss of a root earlier whenever at a subdivision step the indices of all suspect squares sum to less than m.

circle $C(c,\rho)$. To counter potential adverse impact of such *boundary roots*, apply Alg. $3.1_{c,\rho_i}$ for $i = 0,1,\dots,2m$, where the $2m+1$ concentric circles $C_i = C(c,\rho_i,\theta)$, $\rho_i = \rho_0/\theta^{2i}$, $i = 0,1,\dots,2m$, are sufficiently well isolated pairwise, so that a single root can corrupt the output of Alg. $3.1_{c,\rho_i,\theta}$ for only a single i, and so the majority vote certifies that at least ℓ roots lie in the θ-dilation of the outermost disc $D(c,\theta\rho_0)$ of the family or less than ℓ roots lie in its innermost disc $D(c,\rho_{2m})$. This defines a σ-soft ℓ-test for $\sigma > 1$ equal to the ratio of the radii of the latter two discs.

Next we elaborate upon this test.

Without loss of generality *(wlog)* let $c = 0$, $\theta\rho = 1$, $C(c,\theta\rho) = C(0,1)$.

Algorithm 5.1. Deterministic ℓ-test.

INPUT: A black box polynomial p of a degree d and two positive integers ℓ and $m \geq \ell$ such that the unit disc $D(0,1)$ contains at most m roots.

INITIALIZATION: Fix

$$\sigma > 1,\ \theta = \sigma^{\frac{1}{4m+1}},\ q = \lfloor \log_\theta(4d+2) \rfloor,\ \rho_i = \frac{1}{\theta^{2i+1}},\ i = 0,1,\dots,2m. \tag{5.1}$$

OUTPUT: 0 if $\#(D(0,1/\sigma)) < \ell$ or 1 if $\#(D(0,1)) \geq \ell$.

COMPUTATIONS: Apply Alg. $3.1_{0,\rho_i,\theta}$ for q of (5.1) and $i = 0,1,\dots,2m$.

Output 1 if at least $m+1$ integers $\bar{s}_0$ output by the algorithm (among all $2m+1$ such integers) exceed $\ell - 1$. Otherwise output 0. (For $\ell = 1$ we can output 1 already on a single output $\bar{s}_0 > 0$; for $\ell = m$ we can output 0 already on a single output $\bar{s}_0 < m$.)

Theorem 5.2. *Alg. 5.1 is a σ-soft ℓ-test for $\sigma > 1$ of our choice (see (5.1)).*

Proof. Thm. 3.1 implies that for every i, Alg. $3.1_{0,\rho_i,\theta}$ outputs $\bar{s}_0 = \#(D(0,\rho_i))$ unless there is a root in the open annulus $A_i := A(0,\rho_i/\theta,\theta\rho_i)$.

The $2m+1$ open annuli A_i, $i = 0,1,\dots,2m$, are disjoint. Hence a single root cannot lie in two annuli A_i, and so Alg. $3.1_{0,\rho_i,\theta}$ can fail only for a single integer $i \in [0,2m]$, and so at least $m+1$ outputs of the algorithm are correct.

To complete the proof, combine this property with the relationships $\rho_{2m} = 1/\theta^{4m+1}$ (cf. (5.1)), $\rho_0 = 1/\theta < 1$, and $\#(D(0,\rho_{i+1})) \leq \#(D(0,\rho_i))$, which hold because $D(0,\rho_{i+1}) \subseteq (D(0,\rho_i)$ for $i = 0,1,\dots,2m-1$. $\square$

Observation 5.1. *[Cf. Observation 3.2.] Alg. 5.1 runs at HLA cost $\mathcal{A} = (2m+1)q$ for $q = \lfloor \log_\theta(4d+2) \rfloor = \lfloor (4m+1)\log_\sigma(4d+2) \rfloor$, which dominates the cost of the remaining ops involved, even in the cases of the Mandelbrot and sparse input polynomials and which is in $O(m^2\log(d))$ if $\sigma - 1$ exceeds a positive constant.*

6 Root radius estimation based on ℓ-test

In this section we assume that we are given an algorithm for ℓ-test and go back from the decision problem of ℓ-test to approximation of $r_{d-\ell+1}(c')$, the ℓth smallest root radius from a complex point c', assumed to lie in a fixed range $[a_-,a_+]$.[20] Wlog let this be the range $[\epsilon,1]$ for a fixed small positive $\epsilon = 1/2^b$. The bisection algorithm is too slow for us, and we accelerate it by applying it to

[20]For our root-finders we only need to approximate $r_{d-\ell+1}(c')$ for $\ell = 1$ and $\ell = m$ and only with a constant bound on the relative error, but we will estimate $r_{d-\ell+1}(c')$ within $1/2^b$ for any real b and ℓ, $1 \leq \ell \leq m$, and estimates, which is of independent interest for root-finding.

approximate the exponent of a target value. Call that algorithm *bisection of exponent* or *BoE* and apply it to bracket the ℓth smallest root radius $r_{d-\ell+1}$ in a range $[\rho_-, \rho_+]$ such that $0 < \rho_< - \rho_+ \le \epsilon$ or $0 < \rho_+ < \beta\rho_-$ for two fixed tolerance values $\epsilon = 1/2^b$ and $\beta = 2^\phi > 1$.

Assume that the unit disc $D(0,1)$ contains at most m roots for $1 \le \ell \le m$ and that besides the above tolerance values the input of BoE includes a softness bound $\sigma = 1 + \alpha$ for a fixed small positive parameter $\alpha < 1/2$ and a σ-soft ℓ-test$_{c,\rho}$, to be used as the pivot of BoE.

Since the test is soft, so is BoE as well: for a range $[b_-, b_+]$ with midpoint $\widehat{b} = 0.5(b_+ + b_-)$ regular rigid bisection continues the search in the range $[b_-, \widehat{b}]$ or $[\widehat{b}, b_+]$, but our soft BoE continues it in the range $[b_-, (1+\alpha)\widehat{b}]$ or $[(1-\alpha)\widehat{b}, b_+]$ for a fixed small positive α. The approximations still converges linearly and just a little slower than the regular bisection: the approximation error decreases by a factor of $\frac{2}{1+\alpha}$ rather than by twice.[21]

Algorithm 6.1. Bracketing the ℓ-th smallest root radius based on ℓ-test.

INPUT: Two integers ℓ and m such that $1 \le \ell \le m$ and $\#(D(0,1)) \le m$, $\sigma = 1 + \alpha$ for $0 < \alpha < 1/2$, a σ-soft ℓ-test$_{0,\rho}$, and two tolerance values $\beta := 2^\phi > 1$ and $\epsilon = 1/2^b$, $b \ge 1$.

OUTPUT: A range $[\rho_-, \rho_+]$ such that $[\rho_-, \rho_+] = [0, \epsilon]$ or $\rho_- \le r_{d-\ell+1} \le \rho_+ \le \beta\rho_-$.

INITIALIZATION: Write $\bar{b} := \log_2(\frac{\sigma}{\epsilon})$, $b_- := -\bar{b}$, and $b_+ := 0$.

COMPUTATIONS: Apply ℓ-test$_{0,\frac{\epsilon}{\sigma}}$. Stop and output $[\rho_-, \rho_+] = [0, \epsilon]$ if it outputs 1. Otherwise apply ℓ-test$_{0,\rho}$ for

$$\rho_- = 2^{b_-},\ \rho_+ = 2^{b_+},\ \rho := 2^{\widehat{b}},\ \text{and}\ \widehat{b} := \lceil (b_- + b_+)/2 \rceil. \tag{6.1}$$

Write $b_- := (1-\alpha)\widehat{b}$ if the ℓ-test outputs 0; write $b_+ := (1+\alpha)\widehat{b}$ if the test outputs 1. Then update $\widehat{b}$ and ρ according to (6.1).

Continue recursively: stop and output the current range $[\rho_-, \rho_+] = [2^{b_-}, 2^{b_+}]$ if $b_+ - b_- \le \phi$, that is, if $\rho_+ \le \beta\rho_-$; otherwise update $\widehat{b}$ and ρ and reapply the computations.

The disc $D(0,\epsilon)$ contains at least ℓ roots, that is, the range $[0,\epsilon]$ brackets the ℓth smallest root radius $r_{d-\ell+1}$ if the σ-soft ℓ-test applied to the disc $D(0, \frac{\epsilon}{\sigma})$ outputs 1. Otherwise, the disc $D(0, \frac{\epsilon}{\sigma})$ contains less than ℓ roots, and so the root radius $r_{d-\ell+1}$ lies in the range $[\frac{\epsilon}{\sigma}, 1]$, in which we shall continue our search.

Readily verify that our policy of recursively updating the range $[b_-, b_+]$ maintains this property throughout and that updating stops where $b_+ - b_- \le \phi$, that is, where $\rho_+ \le \beta\rho_-$. This implies *correctness* of Alg. 6.1.

Let us estimate the number $\mathbb{I}$ of its invocations of an ℓ-test.

Theorem 6.1. *Alg. 6.1 invokes an ℓ-test $\mathbb{I}$ times where $\mathbb{I} \le \lceil \log_\nu(\frac{\bar{b}}{\phi}) \rceil + 1$ for $\nu = \frac{2}{1+\alpha}$, $\mathbb{I} = O(\log(b/\phi))$.*

Proof. The computations in Alg. 6.1 stop after the first application of an ℓ-test if it outputs 1. It remains to count the subsequent ℓ-tests if it outputs 0.

Each test decreases the length $\bar{b} = b_+ - b_-$ of its input range $[b_-, b_+]$ for the exponent to $0.5(1+\alpha)\bar{b}$. Hence i applications change it into $\frac{(1+\alpha)^i}{2^i}\bar{b}$ for $i = 1, 2, \ldots$. These values linearly converge to 0 as i increases to the infinity, and the theorem follows. $\square$

[21] If we applied BoE to the range $[0,1]$, we would have to bisect the infinite range $[-\infty, 0]$, but we deal with the narrower range $[-b, 0]$ since we have narrowed the range $[0,1]$ to $[\epsilon, 1]$.

Remark 6.1. *In the case where we know that $\#(D(0,\rho)) \ge \ell$ for a fixed ρ in the range $[\epsilon, 1]$, we only need to refine the initial range $[\rho, 1]$ for the root radius $r_{d-\ell+1}$.*

Clearly, $\mathbb{I}$ applications of ℓ-test increase the number of evaluation points for the ratio $\frac{p'(x)}{p(x)}$ by at most a factor of $\mathbb{I}$, and if the ℓ-test is randomized, then the error probability also grows by at most a factor of $\mathbb{I}$. Let us specify these estimates for Alg. 6.1.

Corollary 6.1. *Let precisely m zeros of a black box polynomial $p(x)$ of a degree d lie in the unit disc, that is, let $\#(D(0,1)) = m$, and let $i(D(0,1)) \ge \theta$, where $\theta - 1$ exceeds a positive constant. Given two positive tolerance values $\epsilon = 1/2^b$ and ϕ, an integer ℓ, $1 \le \ell \le m$, and a black box ℓ-test, running at HLA cost $\mathcal{NR}$, Alg. 6.1 approximates the ℓ-th smallest root radius $|z_j|$ within the relative error 2^ϕ or determines that $|z_j| \le \epsilon$, performing at HLA cost $\mathbb{I} \cdot \mathcal{NR}$ for $\mathbb{I} = O(\log(\frac{b}{\phi}))$ estimated in Thm. 6.1.*

7 Precision of computing in our root-finders

7.1 Doubling precision and a posteriori Boolean complexity estimates

How should one choose computational precision for approximation of a polynomial $p(x)$, given x, an absolute error bound TOL, and a black box algorithm $\mathbb{A}$ for polynomial evaluation? The solution depends on the value of $|p(x)|$ at x, unknown a priori. Schönhage proved a worst case lower bound on $|p(x)|$ for $x \in C(0,1)$ and $||p||_1 = 1$ (see [120, Eqn. (9.4)] and some elaboration upon this in our Sec. 8) and noticed that his bound tended to be a great overestimate in his practice.

A simple *practical recipe* of [10, 14, 21] is immune to such an overestimate and was the basis for highly successful performance of MPSolve. Namely, first apply the algorithm $\mathbb{A}$ with a low precision and then recursively double it until the output error decreases below TOL.

The Boolean cost of the computations roughly doubles at every recursive step, while the output error bound decreases by a factor near 2. For simplified crude estimates let it be exactly 2. Then precision and the Boolean cost at the final recursive step exceed their optimal values by at most twice, and the overall Boolean cost at all recursive steps exceeds the Boolean cost at the final step by at most twice, even if one does not reuse the results of the previous steps.

The Boolean cost of polynomial evaluation and of root-finding under this recipe is proportional to HLA cost and thus decreases for polynomials that can be evaluated fast. Next we estimate Boolean cost of our root-finders for a *general input polynomial*, thus giving up potential acceleration in the case of the Mandelbrot polynomials and the sums of a small number of shifted monomials.

We can equally well apply the recipe of recursively doubling precision where a target tolerance bound RTOL is fixed for a *relative output error* of the evaluation, and in the rest of this section we prove that already under a choice of RTOL$= O(\log(d))$ at that stage, proceeding with a precision of $O(\log(d))$ bits is sufficient for correctness of our subdivision root-finders.

7.2 Reduction to approximation of xNIR(x) within $\frac{3}{8}$

We approximate $\mathrm{NIR}(x) = \frac{p'(x)}{p(x)}$ with the first order divided difference:

$$\mathrm{NIR}_\delta(x) := \frac{1}{\delta} - \frac{p(x-\delta)}{\delta\ p(x)} = \frac{p(x) - p(x-\delta)}{p(x)\delta} = \frac{p'(y)}{p(x)} \approx \mathrm{NIR}(x) \text{ for } \delta \approx 0. \tag{7.1}$$

Here y lies in the line segment $[x-\delta, x]$ by virtue of Taylor-Lagrange's formula, and so $y \approx x$ for $\delta \approx 0$. Next we estimate precision of computing that supports approximation of $\mathrm{NIR}(x)$ required in our root-finders.

Next we estimate precision of computations in our deterministic and randomized e/i tests, and we only need to consider them performed for θ-isolated discs $D(c,\rho)$ with $\theta - 1 > 0$ of order $\frac{1}{m}$.

Map (2.2) reduces such a test for $p(x)$ to the disc $D(0,1)$ for the polynomial $t(x) = p(\frac{x-c}{\rho})$

The output of an e/i test is an integer, and we only need to certify that the output errors of Algs. 5.1 and 4.1 are less than $\frac{1}{2}$.

We achieve this by performing our algorithms with a precision of $O(\log(d))$ bits at HLA cost of order $m\log(d)$ provided that a black box oracle furnishes to us the ratio $\frac{t(x-\delta)}{t(x)}$ within $\frac{\delta}{8}$ for any fixed positive $\delta = O(1/d^{O(1)})$.

Precision of computations at that stage defines overall Boolean cost of our root-finders, which we estimate in the next sections.

Our e/i tests compute the q values $\rho x\frac{p'(c+\rho x)}{p(c+\rho x)} = x\frac{t'(x)}{t(x)}$, for $x = \zeta^g$, ζ of (3.2), and $g = 0,1,\dots,q-1$, sum these values, and divide the sum by q.

By slightly abusing notation write $\mathrm{NIR}(x) := \frac{t'(x)}{t(x)}$ rather than $\mathrm{NIR}(x) = \frac{p'(x)}{p(x)}$. According to straightforward error analysis (cf. [20, Appendix A of Ch.3] and the references therein) rounding errors introduced by these additions and division are dominated, and we can ignore them; then it is sufficient to compute $\zeta^g\mathrm{NIR}(\zeta^g)$ within, say, $\frac{3}{8}$ for every g. Indeed, in that case the sum of q error bounds is at most $\frac{3q}{8}$ and decreases below $\frac{1}{2}$ in division by q.

7.3 Precision of representation of $x\mathrm{NIR}(x)$ within $\frac{1}{8}$

Recall Eqn. (2.1) and represent $x\mathrm{NIR}(x)$ for $|x| = 1$ within $\frac{1}{8}$ by using a precision of $O(\log(d))$ bits.

Theorem 7.1. *Write $C := C(c,\rho)$ and assume that $|x| = 1$ and the circle C is θ-isolated for $\theta > 1$. Then $|x\mathrm{NIR}(x)| \le \frac{d}{1-1/\theta} = \frac{\theta d}{\theta - 1}$.*

Proof. Eqn. (2.1) implies that $|\mathrm{NIR}(x)| = |\sum_{j=1}^{d}\frac{1}{x-z_j}| \le \sum_{j=1}^{d}\frac{1}{|x-z_j|}$ where $|x - z_j| \ge (1-1/\theta)\rho$ for $|z_j| < 1$, while $|x - z_j| \ge (\theta - 1)\rho$ for $|z_j| > 1$ since $i(C) \ge \theta$. Combine these bounds, write $m := \#(D(c,\rho))$, and obtain $|x\mathrm{NIR}(x)| \le \frac{m}{1-1/\theta} + \frac{d-m}{\theta-1} \le \frac{d}{1-1/\theta}$ for $|x| = 1$. $\square$

Corollary 7.1. *One can represent $x\mathrm{NIR}(x)$ within $\frac{1}{8}$ for $|x| = 1$ by using a precision of $3 + \lceil\log_2(\frac{d}{1-1/\theta})\rceil$ bits, which is in $O(\log(d))$ provided that $\frac{1}{\theta-1} = d^{O(1)}$.*

7.4 What is left to estimate?

Instead of $x\mathrm{NIR}(x) = \frac{xt'(x)}{t(x)}$ we actually approximate $\frac{xt'(y)}{t(x)}$ where $|y - x| \le \delta$ and $t'(y)$ is equal to the divided difference of (7.1). Thus we shall increase the above error bound $\frac{1}{8}$ by adding to it the upper bounds $\frac{1}{8}$ on $\alpha := |\frac{xt'(x)}{t(x)} - \frac{xt'(y)}{t(x)}|$ for $|x| = 1$ and on the rounding error β of computing $x\frac{t(x)-t(x-\delta)}{\delta t(x)}$.

Next we estimate α in terms of d, θ, and δ – by extending the proof of [68, Lemma 3.6], then readily ensure that $\alpha < \frac{1}{8}$ by choosing δ with a precision $\log_2(1/|\delta|)$ of order $\log(d)$. Finally we estimate β.

7.5 Precision of approximation with divided difference

We simplify our estimates by dropping the factor x for $|x| = 1$ and use the following lemma, which is [68, Fact 3.5].

Lemma 7.1. *For* $t(x)=\prod_{j=1}^{d}(x-y_j)$ *and a non-negative integer* $j\le d$ *it holds that*

$$t^{(j)}(x)=j!\ t(x)\sum_{S_{j,d}}\prod_{j\in S_{j,d}}\frac{1}{x-y_j}$$

where the summation $\sum_{S_{j,d}}$ *is over all subsets* $S_{j,d}$ *of the set* $\{1,\dots,d\}$ *having cardinality* j*.*

Theorem 7.2. *Under the assumptions of Thm. 7.1 let* $w=w(x):=\max_{j=1}\frac{1}{|x-y_j|}$*. Then* $|t(x)|\alpha=|\frac{t(x)-t(x-\delta)}{\delta}-t'(x)|\le|t(x)\frac{(d\ w)^2\delta}{1-\delta\ d\ w}|$*.*

Proof. The claimed bound on $|\xi'|$ follows from (2.1). It remains to apply the first five lines of the proof of [68, Lemma 3.6] with f, α, n, and ξ replaced by t, $-\delta$, d,and w, respectively. Namely, first obtain from Taylor's expansion that $t(x)-t(x-\delta)=\sum_{j=0}^{\infty}\frac{\delta^j}{j!}t^{(j)}(x)-t(x)$.

Substitute the expressions of Lemma 7.1 and obtain

$$t(x)-t(x-\delta)=\delta t'(x)+\sum_{j=2}^{\infty}\delta^j t(x)\sum_{S_{j,d}}\prod_{j\in S_{j,d}}\frac{1}{x-y_j}.$$

Combine this equation with the assumed bounds on $\frac{1}{|x-y_j|}$ and deduce that

$$\Big|\frac{t(x)-t(x-\delta)}{\delta}-t'(x)\Big|\le\Big|\frac{f(x)}{\delta}\sum_{j=2}^{\infty}\delta^j w^j d^j\Big|\le\Big|f(x)\frac{(d\ w)^2\ \delta}{1-\delta\ d\ w}\Big|.$$

□

Recall that in our estimates for α and δ we can apply Thm. 7.1 for $w\le\frac{\theta}{\theta-1}$ and obtain

Corollary 7.2. *Under the assumptions of Thm. 7.1, let* $w\le\frac{\theta}{\theta-1}$*. Then*

$$\alpha<1/8\text{ for }|\delta|\le\frac{(\theta-1)^2}{10d^2\theta^2}.$$

7.6 The impact of rounding errors

To approximate $|x\mathrm{NIR}_{c,\rho}(x)|$ within $\frac{3}{8}$ it remains to approximate β within $\frac{1}{8}$.

Theorem 7.3. *Suppose that a black box oracle computes for us the values* $t(y)$ *within a relative error bounds* $\nabla(y)$ *for* $y=x$, $y=x-\delta$, *and* δ *of Cor. 7.2. Then we can ensure that* $|\beta|<\frac{1}{8}$ *by choosing a proper* $\nabla(y)$ *of order* $1/d^{O(1)}$*, represented with a precision of* $\log(\frac{1}{\nabla(y)})=O(\log(d))$ *bits.*

Proof. Write $\beta\delta=\frac{t(x-\delta)(1+\nabla(x-\delta))}{t(x)(1+\nabla(x))}-\frac{t(x-\delta)}{t(x)}=\frac{t(x-\delta)}{t(x)}\ \frac{\nabla(x-\delta)-\nabla(x)}{1+\nabla(x)}$.

Taylor-Lagrange's formula implies that $\frac{t(x-\delta)}{t(x)}=1+\frac{t(x-\delta)-t(x)}{t(x)}=1+\delta\frac{t'(u)}{t(x)}$, for $u\in[x-\delta,x]$. Hence $|\frac{t(x-\delta)}{t(x)}|\le1+|(1+\alpha')\delta|$ for $\alpha'=\frac{t'(u)-t(x)}{t(x)}$.

By extending Cor. 7.2 obtain $|\alpha'|\le\frac{1}{8}$.

Hence $|\frac{t(x-\delta)}{t(x)}|\le w:=1+\frac{9|\delta|}{8}$, while $|\beta\delta|\le w\frac{|\nabla(x)|+|\nabla(x-\delta)|}{1-|\nabla(x)|}$.

Now choose $\nabla(y)$ such that $|\nabla(y)|\le\frac{|\delta|}{17w}$ for $y=x$ and $y=x-\delta$, and so $w(|\nabla(x)|+|\nabla(x-\delta)|)\le\frac{2}{17\delta}$. Then verify that $\log(\frac{1}{|\nabla(y)|})=O(\log(d))$, $1-|\nabla(x)|\ge1-\frac{|\delta|}{17w}>\frac{16}{17}$, and $|\delta\beta|<\frac{|\delta|}{8}$. □

By combining Cors. 7.1 and 7.2 and Thm. 7.3 obtain

Corollary 7.3. *Suppose that a black box oracle can output approximations of a d-th degree polynomial $t(x)$ with any relative error in $1/d^{O(1)}$ for $|x| \le 1$ and let the unit disc be θ-isolated for $t(x)$ and for $\frac{1}{\theta-1} = d^{O(1)}$. Then we can perform e/i test for $t(x)$ on the unit disc by using a precision of $O(\log(d))$ bits.*

8 Approximation of matrix eigenvalues

Given a $d \times d$ matrix M having characteristic polynomial $p(x) = \det(xI - M)$, our root-finders can approximate all m its zeros (that is, the eigenvalues of M) lying in a fixed disc D on the complex plane isolated from external eigenvalues.

Recall Gershgorin's bound for any eigenvalue λ of a matrix $M = (a_{i,j})_{i,j=1}^{d}$ [124, Thm. 1.3.2]: $|\lambda - a_{i,i}| \le \sum_{j \ne i} |a_{i,j}|$ for some i, $1 \le i \le d$.

Hence, clearly, $|\lambda| \le ||M||_F$.

Combine this bound with the precision estimates of the previous section for our root-finders[22] and with the following result of Storjohann [123] (cf. Remark 8.1):

Theorem 8.1. *One can evaluate the determinant of a $d \times d$ integer matrix A by using an expected number of $O(d^{\omega} \log^2(d) \log(|A||_F))$ bit operations.*

This implies a record Boolean complexity bound $\tilde{O}(m^2 \tilde{b}^2 d^{\omega})$ (cf. (1.7)) for approximation within ϵ of all m eigenvalues of M lying in the disc D provided that $\tilde{b} = \log_2(||M||_F/\epsilon)$.

Remark 8.1. *Our root-finders are essentially reduced to evaluation of $NIR(x) = \frac{p'(x)}{p(x)}$ at sufficiently many points x. Hence we can rely on the following expressions for approximation of the eigenvalues of a matrix or a matrix polynomial:*

$$\frac{p(x)}{p'(x)} = \frac{1}{\mathrm{trace}((xI - M)^{-1})}$$

where M is a matrix and $p(x) = \det(xI - M)$ (cf. (2.1)) and

$$\frac{p(x)}{p'(x)} = \frac{1}{\mathrm{trace}(M^{-1}(x)M'(x))}$$

where $M(x)$ is a matrix polynomial and $p(x) = \det(M(x))$ [19, Eqn. (5)].[23] Based on these expressions one can reduce approximation of eigenvalues of a matrix or a matrix polynomial to its inversion. This tends to be more numerically stable than computing determinants and is significantly accelerated where an input matrix or matrix polynomial can be inverted fast, e.g., is data sparse. In that highly important case using matrix inversion should supersede using evaluation of $\det(xI - M)$.

9 Boolean complexity of polynomial root-finding

9.1 Our goal and a basic theorem

Next we *estimate the Boolean complexity of our root-finders applied to a general polynomial* p of (1.1) whose coefficients are given with a precision sufficient to support our root-finding. In view

[22]Practically, one can combine our root-finders with the recipe of doubling the precision of computing of [10, 14, 21].

[23]These expressions have been obtained based on the well-known Jacobi's formula $\mathrm{d}(\ln(\det(A))) = \mathrm{trace}(A^{-1}\mathrm{d}A)$, which links differentials of a nonsingular (invertible) matrix A and of its determinant.

of our study inthe previous sections, we only need to estimate the cost of evaluation of p with a relative error in $1/d^c$ for any constant c and to combine the estimates with the results of Sec. 7.

Kirrinnis in [60, Thm. 3.9, Alg. 5.3, and Appendix A.3] bounds the Boolean complexity of modular representation of a polynomial, which implies its multipoint evaluation.[24] He applies his and Schönhage's study of Boolean complexity of basic polynomial computations (cf. [120]) and applies it to the algorithm of Fiduccia [37] and Moenck and Borodin [73] (see [18, Sec. 4.5] for its historical account).

Theorem 9.1. [60, Thm. 3.9]. *Given a positive b, the coefficients of a polynomial $t(x) := \sum_{i=0}^{d} t_i x^i$ such that $||t(x)||_1 = \sum_{i=0}^{d} |t_i| \le 1$, and $q \ge d$ complex points $z_1, \dots, z_q$, in the unit disc $D(0,1)$, one can approximate the values $t(z_1), \dots, t(z_q)$ within $1/2^b$ by using $O(\mu((d\log(d)+q)(b+q))) = \tilde{O}((b+q)(d+q))$ Boolean operations for*

$$\mu(s) := O(s\log(s)\log(\log(s))) = \tilde{O}(s) \tag{9.1}$$

denoting the Boolean complexity of multiplication of two integers modulo 2^s [127].

To cover also the case where $q \le d$, represent $t(x) = \sum_{j=0}^{h} t_j(x) x^{jq}$ for $h = \lceil d/q \rceil - 1$ and polynomials $t_j(x)$ of degrees at most q sharing their coefficients with $t(x)$ and then evaluate them at the points $z_1, \dots, z_q$:
(i) the polynomials $p_j(x)$,
(ii) the powers x^{jq} for $j = 0, 1, \dots, h$, and finally
(iii) $t(x) = \sum_{j=0}^{h} t_j(x) x^{jq}$.

At stages (ii) and (iii) we only perform $O(d)$ ops and increase precision of computing by $O(\log(d)$ bits because $\max_{j=0}^{h} |z_j| \le 1$.

At stage (i) we apply h times Thm. 9.1, for d replaced by m; the supporting Boolean cost bound $O(\mu((b+q)d\log(q))) = \tilde{O}((b+q)d)$ dominates the overall Boolean cost of the evaluation of $p(x)$; this extends the bound $\tilde{O}((b+q)(d+q))$ of Thm. 9.1 to the case where $q \le d$.

9.2 Reduction of our root-finders to multipoint polynomial approximation

Given a complex c, a pair of positive b and ρ, and $d+1$ coefficients of a polynomial $p = p(x)$ of (1.1) such that the disc $D(c, \rho)$ is isolated and contains precisely m zeros of p, we approximate all these zeros within $R/2^b$ for $R = |c| + \rho$ by applying subdivision root-finding iterations with deterministic or Las Vegas randomized e/i tests of Secs. 4 or 5, respectively. Then every subdivision step is reduced to application of e/i tests to $\bar{m} = O(m)$ discs $D(c_\lambda, \rho_\lambda)$ for $\rho_\lambda \ge R/2^b$ and all λ.

Such a test amounts to approximation of $p(x)$ at $O(q\log(d))$ equally spaced points x on each of v circles sharing the center c_λ and having radii in the range $[\rho_\lambda/\sigma, \rho_\lambda\sigma]$ for $1 < \sigma < \sqrt{2}$. Here $v = 2m+1$ and $q = m^2$ or $v = O(1)$ and $q = m$ in our e/i tests of Secs. 4 or 5, respectively, while $q = 1$ under the random root model.

In Thm. 9.1 a polynomial $t(x)$ with $||t(x)||_1 \le 1$ is evaluated at points $z_j \in D(0,1)$. To ensure these assumptions we first scale the variable $x \mapsto Rx$ to map the discs $D(c_\lambda, \rho_\lambda\sigma) \mapsto D(\bar{c}_\lambda, \bar{\rho}_\lambda) \subseteq D(0,1)$, for $\bar{c}_\lambda = c_\lambda/R$, $\bar{\rho}_\lambda = \rho_\lambda/R \ge 1/2^b$, and all λ.

[24] Precision of computing, affecting the overhead constants hidden in the above estimates and supporting the latter algorithm, grows fast as d increases; e.g., the IEEE standard double precision of the customary numerical computations is not enough in that algorithm already for $d > 50$; for $b = O(\log(d))$ one can fix this deficiency by applying the Fast Multipole Method of Greengard and Rokhlin [44, 26, 92, 93] to the equivalent task of multiplication of Vandermonde matrix by a vector.

Then we write $t(x) := p(Rx)/\psi$ for $\psi = O(R^d||p(x)||_1)$ such that $||t(x)||_1 = 1$ and approximate $t(x)$ within $1/2^{\bar{b}}$ or equivalently within the relative error bound $1/(|t(x)|2^{\bar{b}})$ at $\bar{q} := qv$ points x in $\bar{m}$ e/i tests.[25]

Due to Cor. 7.3 it is sufficient to ensure the bound $1/(|t(x)|2^{\bar{b}}) = 1/d^{O(1)}$; we achieve this by choosing any $\bar{b} \ge \log(\frac{1}{|t(x)|}) + O(\log(d))$.

In our e/i tests we only need to approximate $t(x)$ at the points x lying on θ_λ-isolated circles where $\frac{1}{\theta_\lambda - 1} = O(m)$. Map the disc $D(\bar{c}_\lambda, \bar{\rho}_\lambda)$ into the unit disc $D(0,1)$. This does not change isolation of the boundary circle and may increase the norm $||t(x)||_1$ by a factor of at most 2^{b+d} because $|\bar{c}_\lambda| \le 1$ and $\bar{\rho}_\lambda) \ge 1/2^b$. Now recall that $\log(\frac{||t(x)||_1}{|t(x)|}) = O(d\log(m))$ (cf. [120, Eqn.(9.4)]), [26] substitute $||t(x)||_1 \le 2^{b+d}$, and obtain that $\bar{b} = \log(\frac{1}{|t(x)|}) + O(\log(d)) = O(d\log(m) + b)$.

Remark 9.1. *For $m = d$ the above bound on $\log(\frac{||t(x)||_1}{|t(x)|})$ of [120, Eqn.(9.4)]) is sharp up to a constant factor and is reached at the polynomial $t(x) = (x + 1 - \frac{1}{m})^d/(2 - \frac{1}{m})^d$, $\rho_\lambda = 1$ and $x = -1$. For $m < d$ [120, Thm. 4.5] implies a little stronger bounds $\log(\frac{||t(x)||_1}{|t(x)|}) = O(d + m\log(m))$ and hence $\bar{b} = O(m\log(m) + d + b)$ because all discs $D(\bar{c}_\lambda, \bar{\rho}_\lambda)$ lie in θ-isolated unit disc $D(0,1)$ for $\theta - 1$ exceeding a positive constant.*

9.3 Boolean complexity of our e/i tests and root-finders

Now apply Thm. 9.1 with q and b replaced by $\bar{q} = O(qm)$ and $\bar{b} = O(b + d + m\log(m))$ (cf. Remark 9.1), respectively, and obtain that the Boolean complexity of our $O(m)$ e/i tests at any fixed subdivision step is in $O(\mu(qm + d\log(d))((q + \log(m))m + d + b)))$ for $\mu(s)$ of (9.1) and $q = q(m)$ of Thm. 1.7 for our e/i tests of Secs. 4 and 5. This implies the estimate $O(\mu(q(m)m + d\log(d))((q(m) + \log(m))m + d + b)b)) = \tilde{O}((q(m)m + d)(q(m)m + d + b)b)$ for the overall Boolean time of our $O(b)$ subdivision steps.

9.4 Higher Boolean complexity of general polynomial root-finding based on map (2.2) and root-lifting

Our Boolean complexity estimates for root-finding rely on restriction of computations to approximation of NIR without involving coefficients of p. Knowing the coefficients of the polynomial $t(x) = p(\frac{x-c}{\rho})$ one can devise very fast e/i test for the disc $D(c, \rho)$ by applying Pellet's theorem to $t_h(x)$ for h of order $\log(d)$, whose zero are the hth powers of the zeros of $t(x)$ [22, 23].

The paper [23] applies subdivision iterations of [134, 46, 114, 88] with e/i tests performed by means of Pellet's theorem, which involves all coefficients of an input polynomial, thus losing the benefits where,say, $p(x)$ is the Mandelbrot polynomial. The tests must be applied to discs covering all suspect squares, up to $4m$ squares per subdivision iteration.

A subdivision algorithm of [23] first maps the covering disc into the unit disc $D(0,1)$ (cf. (2.2)), thus losing the benefits where $p(x)$ is the sum of a small number of shifted monomials. Then the algorithm performs order of $\log(\log(d))$ root-squaring steps (one can apply order of $\log(d)$ root-lifting steps instead but within the same Boolean cost estimates), and finally applies Pellet's theorem to the resulting polynomial $t_{c,\rho}(x) = \sum_{i=0}^{d} t_i x^i$.

[25] Scaling p and x above is done once and for all e/i tests; to compensate for this we only need to change our bound on the output precision b by $\lceil|\log_2 R|\rceil$ bits.

[26] This bound is readily deduced from the following non-trivial result [120, Thm. 4.2]: Let $t(x) = f\prod_{j=1}^{m}(x - z_j)\prod_{j=m+1}^{d}(\frac{x}{z_j} - 1)$, $||t(x)||_1 = 1$, $|z_j| < 1$ for $j \le m$, and $|z_j| > 1$ for $j > m$. Then $|f| \ge 1/2^d$.

Transition from $p(x)$ to $t_{c,\rho}(x)$, however, involves scaling of the variable x by factors ranging from $R/2^b$ to $R = \max_{j-1}^d |z_j|$ (besides root-squaring and shift of the variable). Hence for every i, $i = 0, 1, \dots, d$, the coefficient t_i must be computed within $1/2^{b-i}$ to approximate the zeros of p within $1/2^b$ (see Observation 1.2).

This involves at least bg bits for $g = 1, \dots, d$, that is, at least $0.5(d-1)db$ bits and $0.25(d-1)db$ bit-operations overall for a single e/i test.

A single subdivision iteration for Problem 1_m can involve up to $4m$ e/i tests and hence up to at least $(d-1)dbm$ bit-operations. This is by a factor of m greater than in root-finder of [83, 90] overall.

The bit-count above and the Boolean cost bounds for root-finders of [83, 90, 23] decrease by a factor of d where the minimal pairwise distance between the zeros of p exceeds a positive constant, but the ratio of the bounds of [23] and [83, 90] remains at least m, where m can be as large as d.

Similar argument shows that even representation of the leading coefficients t_d of $t(x)$ alone after scaling for a singe e/i test involves odder of bd bits, which is amplified to d^2b for t_d^d in every e/i test (see the end of Sec. 1.7) and up to d^2bm per a subdivision iteration with m e/i tests.

It may be interesting that a shift by c such that $|c| \le 2$ is much less costly than scaling:

Theorem 9.2. Boolean cost of the shift of the variable, [122, Lemma 2.3] (cf. [60, Lemma 3.6]). *Given a positive b, the coefficients of a polynomial $p(x) := \sum_{i=0}^d p_i x^i$ such that $||p(x)||_1 \le 1$, and a complex c such that $|c| \le 2$, one can approximate within $1/2^b$ the coefficients $t_0, t_1, \dots, t_d$ of the polynomial $t(x) := \sum_{i=0}^d t_i x^i := p(x-c)$ by performing $O(\mu((b+d)d))$ Boolean operations.*

We can relax the restriction $|c| \le 2$ above by means of scaling the variable $x \mapsto xc$, but as we could see above, this would greatly increase the overall Boolean cost bound.

PART II: Subdivision with exclusion/inclusion tests based on root-lifting

Organization of Part II: We devote the next section to background. In Secs. 11 and 12 we cover our e/i tests and their extensions to m-tests, respectively. In Secs. 13 and 14 we estimate computational precision of our root-finders and their Boolean cost, respectively. In Sec. 15 we compute a disc that covers all d zeros of p.

10 Background

10.1 Definitions

We first redefine the concepts of HLA cost, NR and NIR, used in the Introduction.

- *HLA cost* of a root-finder is the number of points x at which it approximates an input polynomial $p(x)$.
- $\mathrm{NR}_t(x) := \frac{t(x)}{t'(x)}$ is *Newton's ratio*, $\mathrm{NIR}_t(x) := \frac{t'(x)}{t(x)}$ is *Newton's inverse ratio* for a polynomial $t(x)$, $\mathrm{NIR}_t(x)\mathrm{NR}_t(x) = 1$.
- We write $\mathrm{NR}(x) := \mathrm{NR}_p(x)$ and $\mathrm{NIR}(x) := \mathrm{NIR}_p(x)$.
- We approximate $\mathrm{NIR}(x)$ at HLA cost 2 by using the first order divided differences as follows: fix a $\delta \approx 0$ and compute

$$\mathrm{NIR}(x) \approx \frac{p(x) - p(x-\delta)}{p(x)\delta} = \frac{p'(y)}{p(x)} \text{ for } y \in [x-\delta, x]. \tag{10.1}$$

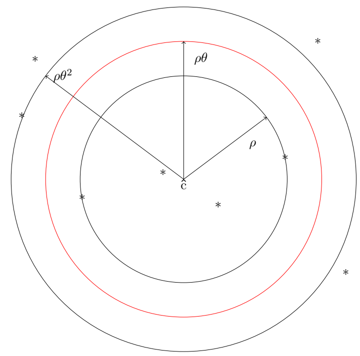


Figure 4: The roots are marked by asterisks. The red circle $C(c, \rho\theta)$ and the disc $D(c, \rho\theta)$ have isolation θ. The disc $D(c, \rho)$ has isolation θ^2.

By virtue of Taylor-Lagrange's formula, $y \approx x$ for $\delta \approx 0$.

- Define squares, discs, circles (circumferences), and annuli on the complex plane: $S(c, \rho) := \{x : |\Re(c-x)| \le \rho, |\Im(c-x)| \le \rho\}$, $D(c, \rho) := \{x : |x-c| \le \rho\}$, $C(c, \rho) := \{x : |x-c| = \rho\}$, $A(c, \rho, \rho') := \{x : \rho \le |x-c| \le \rho'\}$.

- $\Delta(\mathbb{S})$, $CH(\mathbb{S})$, $X(\mathbb{S})$ and $\#(\mathbb{S})$ denote the diameter, root set, convex hull, and index (root set's cardinality) of a set or region $\mathbb{S}$ on the complex plane, respectively.

- A disc $D(c, \rho)$, a circle $C(c, \rho)$, or a square $S(c, \rho)$ is *θ-isolated* for $\theta \ge 1$ if $X(D(c, \rho)) = X(D(c, \theta\rho))$, $X(C(c, \rho)) = X(A(c, \rho/\theta, \rho\theta)$ or $X(S(c, \rho)) = X(S(c, \theta\rho))$, respectively. The *isolation* of the disc $D(c, \rho)$, the circle $C(c, \rho)$, or the square $S(c, \rho)$, denoted $i(D(c, \rho))$, $i(C(c, \rho))$, and $i(S(c, \rho))$, respectively, is the largest upper bound on such a value θ (see Fig. 4).

- A disc D has *softness* $\sigma(D) = \Delta(D)/\Delta(X(D)) \ge 1$ and *rigidity* $\eta(\mathbb{S}) = 1/\sigma(D)$. A disc is *$\sigma$-soft* and *$\eta$-rigid* for any $\sigma \ge \sigma(D)$ and any $\eta \le \eta(D)$. We say *rigid* for "η-rigid", *soft* for "σ-soft", and *isolated* for "θ-isolated" if η, $1/\sigma$, and $\theta - 1$ exceed a positive constant.

- $|u| := ||\mathbf{u}||_1 := \sum_{i=0}^{q-1} |u_i|$, $||\mathbf{u}|| := ||\mathbf{u}||_2 := (\sum_{i=0}^{q-1} |u_i|^2)^{\frac{1}{2}}$, and $|u|_{\max} := ||\mathbf{u}||_\infty := \max_{i=0}^{q-1} |u_i|$ denote the 1-, 2-, and ∞-norm, respectively, of a polynomial $u(x) = \sum_{i=0}^{q-1} u_i x^i$ and its coefficient vector[27] $\mathbf{u} = (u_i)_{i=0}^{d-1}$, and then (see [42, Eqs. (2.2.5)–(2.2.7)])

$$||\mathbf{u}||_\infty \le ||\mathbf{u}||_2 \le ||\mathbf{u}||_1 \le \sqrt{q}\, ||\mathbf{u}||_2 \le q||\mathbf{u}||_\infty. \tag{10.2}$$

- $r_1(c, t) = |y_1 - c|, \dots, r_d(c, t) = |y_d - c|$ in non-increasing order are the d *root radii*, that is, the distances from a complex center c to the roots $y_1, \dots, y_d$ of a dth degree polynomial $t(x)$. Write $r_j^h(c, t) := (r_j(c, t))^h$, $r_j(c) := r_j(c, p)$.

- Differentiate factorization (1.1) of $p(x)$ to express $\mathrm{NIR}(x)$ as follows:

$$\mathrm{NIR}(x) := \frac{p'(x)}{p(x)} = \sum_{j=1}^{d} \frac{1}{x - z_j}. \tag{10.3}$$

[27]Here and hereafter $\mathbf{v}$ denotes a column vector, while $\mathbf{v}^T$ denotes its transpose.

- *Taylor's shift*, or translation, of the variable $x \mapsto y := x - c$ and its *scaling* $x \mapsto \rho x$ combined define following map:

$$x \mapsto \frac{x-c}{\rho},\ D(c,\rho) \mapsto D(0,1),\ \text{and } p(x) \mapsto t(x) = p\Big(\frac{x-c}{\rho}\Big), \tag{10.4}$$

which maps the zeros z_j of $p(x)$ into the zeros $y_j = \frac{z_j - c}{\rho}$ of $t(y)$, for $j = 1, \dots, d$, and preserves the index and the isolation of the disc $D(c,\rho)$.

Definition 10.1. *For $p(x)$ of (1.1), $\sigma > 1$, and integers ℓ and m such that $1 \le \ell \le m \le d$ and $\#(D(0,1)) \le m$, a σ-**soft** ℓ-**test**, or just ℓ-**test** for short (1-test being e/i test), either outputs 1 and stops if it detects that $r_{d-\ell+1} \le \sigma$, that is, $\#(D(0,\sigma)) \ge \ell$, or outputs 0 and stops if it detects that $r_{d-\ell+1} > 1$, that is,[28] $\#(D(0,1)) < \ell$. ℓ-test applied to the polynomial $t(y)$ of (10.4) is said to be* ℓ-test for the disc $D(c,\rho)$ *and/or* the circle $C(c,\rho)$ *and also* ℓ-test$_{c,\rho}$.[29]

10.2 Reverse polynomial

Define the reverse polynomial of $p(x)$,

$$p_{\text{rev}}(x) := x^d p\Big(\frac{1}{x}\Big) = \sum_{i=0}^{d} p_i x^{d-i},\ p_{\text{rev}}(x) = p_0 \prod_{j=1}^{d} \Big(x - \frac{1}{z_j}\Big) \text{ if } p_0 \neq 0, \tag{10.5}$$

and immediately observe its following properties.

Theorem 10.1. (i) *The roots of $p_{\text{rev}}(x)$ are the reciprocals of the roots of $p(x)$, and hence*

$$r_j(0,p) r_{d+1-j}(0,p_{\text{rev}}) = 1 \text{ for } j = 1, \dots, d. \tag{10.6}$$

(ii) *The unit circle $C(0,1)$ has the same isolation for $p(x)$ and $p_{\text{rev}}(x)$.*

$$\text{(iii) } p'_{\text{rev}}(x) = dx^{d-1} p\Big(\frac{1}{x}\Big) - x^{d-2} p'\Big(\frac{1}{x}\Big) \text{ for } x \neq 0,\ p_d = p_{\text{rev}}(0),\ p_{d-1} = p'_{\text{rev}}(0).$$

(iv) $\frac{p_{\text{rev}}(x)}{p'_{\text{rev}}(x)} = \frac{1}{x} - \frac{\text{NIR}(x)}{x^2},\ x \neq 0,\ \text{NIR}(0) = \frac{p_{d-1}}{p_d} = -\sum_{j=1}^{d} z_j.$

The first equation of (10.5) and claims (iii) and (iv) above express the reverse polynomial $p_{\text{rev}}(x)$, its derivative, and $\text{NIR}p_{\text{rev}}(x)$ through $p(\frac{1}{x})$, $p'(\frac{1}{x})$, and $\text{NIR}(\frac{1}{x})$. For the sake of completeness we include the following theorem.

Theorem 10.2. *For $p(x)$ of (1.1) and $R > 1$ it holds that*

$$\Big|\frac{p(R)}{R^d} - p_d\Big| \le \frac{1}{R-1} \max_{i<d} |p_i|, \tag{10.7}$$

$$\Big|\frac{Rp'(R) - dp(R)}{R^{d-1}} - p_{d-1}\Big| \le \frac{1}{R-1} \max_{0<i<d} |p_i|. \tag{10.8}$$

[28] It is possible that $1 < r_{d-\ell+1} \le \sigma$, but as soon as an ℓ-test verifies any of these lower and upper bounds it stops without checking whether the other bound also holds.

[29] We will only use ℓ-tests for $\ell = 1$ (which are e/i tests) and $\ell = m$.

Proof. To prove bound (10.7) notice that

$$\frac{p(R)}{R^d} - p_d = \sum_{i=0}^{d-1} p_i R^{i-d} = \frac{1}{R}\sum_{i=0}^{d-1} p_i R^{i-d+1},$$

and so

$$\Big|\frac{p(R)}{R^d} - p_d\Big| \le \frac{1}{R}\sum_{i=0}^{d-1} |p_i| R^{i-d+1} \le \max_{i<d} |p_i| \sum_{i=0}^{d-1} |p_i| R^{i-d} \le \frac{1}{R-1}\max_{i<d}|p_i|.$$

Likewise, to prove bound (10.8) notice that

$$Rp'(R) - dp(R) - dR^{d-1}p_{d-1} = \sum_{i=1}^{d-1} i p_i R^{i-1}.$$

Hence

$$\frac{Rp'(R) - dp(R)}{R^{d-1}} - p_{d-1} = \frac{1}{R}\sum_{i=1}^{d-1} \frac{i}{d} p_i R^{i-d+1},$$

and so

$$\Big|\frac{Rp'(R) - dp(R)}{R^{d-1}} - p_{d-1}\Big| \le \max_{0<i<d}|p_i| \sum_{i=1}^{d-1} R^{i-d} \le \frac{1}{R-1}\max_{0<i<d}|p_i|.$$

□

10.3 Root-squaring and root-lifting

The *DLG root-squaring iterations* of Dandelin 1826, Lobachevsky 1834, and Gräffe 1837 (see [45, 74, 100] for the history) compute polynomials $p_{2^i}(x) := \prod_{j=1}^{d}(x - z_j^{2i})$, for $i = 0, 1, \ldots$, as follows:

$$p_0(x) := \frac{p(x)}{p_d},\ \ p_{2^{i+1}}(x^2) := (-1)^d p_{2^i}(x) p_{2^i}(-x),\ \ i = 0, 1, \ldots \tag{10.9}$$

The known studies of DLG iterations always focus and stumble on numerically unstable computation of the coefficients of the polynomials $p_i(x)$ [74, 100], but we only compute their values bypassing those problem.

h-lifting of roots generalizes DLG iterations (cf. (1.3)):

$$p_h(x) := \prod_{g=0}^{h-1} p_0(\zeta_h^g x) = (-1)^d \prod_{j=1}^{d}(x^h - z_j^h),\ \zeta_h := \exp\Big(\frac{2\pi\mathbf{i}}{h}\Big), \tag{10.10}$$

for $h = 1, 2, \ldots$, $p_0(x)$ of (10.9), and $t_{2^k}(x) = p_k(x)$ for all k. We can fix $\delta > 0$ and an integer $h > 0$, compute $p_h(y)$ for $y = x$ and $y = x - \delta$, define the ratios $\mathrm{NR}_{p_h}(x) := \frac{p_h(y)}{p_h'(y)}$ and $\mathrm{NIR}_{p_h}(x) := \frac{p_h'(y)}{p_h(y)}$, and approximate them (at HLA cost $2h$) as follows (cf. (10.1)),[30]

$$\mathrm{NIR}_{p_h}(x) = \frac{1}{\mathrm{NR}_{p_h}(x)} \approx \frac{p_h(x) - p_h(x-\delta)}{p_h(x)\delta} = \frac{p_h'(y)}{p_h(x)}. \tag{10.11}$$

[30] The paper [43] explores direct root-squaring of NIR for a black box polynomial for $x \ne 0$: $\mathrm{NIR}_{p_{h+1}}(x) = \frac{1}{2\sqrt{x}}\left(\mathrm{NIR}_{p_h}(\sqrt{x})\mathrm{NIR}_{p_h}(-\sqrt{x})\right),\ h = 0, 1, \ldots$.

Remark 10.1. *Let $h = 2^k$. The set of 2^k-th roots of unity contains the sets of 2^i-th roots of unity for $i = 0, 1, \dots, k-1$, and one can recursively approximate the ratios $NR_{p_h}(x)$ and/or $NIR_{p_h}(x)$ for h replaced by 2^i in (10.10) and (10.11) and for $i = 0, 1, \dots$ One would stop at $i < k$ if an approximation is close enough.*

Observation 10.1. *(i) Root-lifting preserves the index $\#(D(0,1))$ of the unit disc. (ii) h-lifting (achieved with root-squaring for $h = 2^k$ and integer k) maps isolation $i(D(0,1))$ of the unit disc $D(0,1)$, its softness $\sigma(D(0,1))$, its rigidity $\eta(D(0,1))$, and root radii $r_j(0,p)$ for all j to their h-th powers.*

10.4 The distance to a closest zero of p and fast inclusion tests

We can extend our study of this subsection to estimation of the distance to the farthest zero z_1 of p by applying this study to the reverse polynomial $p_{\rm rev}$.

For bounding r_d and r_1 in terms of the coefficients of p, see [70, Sec. 1.10]; in particular, $r_d \le d|p_0/p_d|^{1/d}$ and $p_0 = p(0)$ for any polynomial $p(x)$.

Since $p_d = 1$ for $p(x) = \det(M)$, we can estimate r_d from above for $p(x) = \det(M)$ at HLA cost 1.

Next we estimate $r_d(c)$ for a black box polynomial p where p_d is not known. For a complex c and positive integers j and h combine Eqn. (10.3), Observation 10.1, and triangle inequality to obtain

$$r_d(c) \le d\ |\mathrm{NR}(c)|, \tag{10.12}$$

$$r_j^h(0,p) = r_j(0,p_h) \text{ and } r_j(c) \le |c| + r_j(0). \tag{10.13}$$

The roots lying far from a point c little affect $\mathrm{NIR}(c)$, and we extend bounds (10.12) as follows.

Theorem 10.3. *If $\#(D(c,1)) = \#(D(c,\theta)) = m$ for $\theta > 1$, then*
$|\mathrm{NIR}(c)| - \frac{d-m}{\theta-1} < \frac{m}{r_d(c)}$.

Proof. We can assume that $|z_j| > \theta$ if and only if $j > m$ and then deduce from Eqn. (10.3) that $|\mathrm{NIR}(c)| \le \sum_{j=1}^{m} \frac{1}{|c-z_j|} + \sum_{j=m+1}^{d} \frac{1}{|c-z_j|} < \frac{m}{r_d(c)} + \frac{d-m}{\theta-1}$. □

Certify inclusion into the disc $D(0,\sigma)$ if $\rho_d(0,p_h) \le \sigma^h$, $\sigma > 1$ and $h > 0$; this test is very poor for worst case polynomials p:

Theorem 10.4. *The ratio $|\mathrm{NR}(0)|$ is infinite for $p(x) = x^d - u^d$ and $u \ne 0$, while*

$$r_d(c,p) = r_1(c,p) = |u|.$$

Proof. Notice that $p'(0) = 0 \ne p(0) = -u^d$, $p(x) = \prod_{j=0}^{d-1}(x - u\zeta_d^j)$ for ζ_d of (1.3). □

Clearly, the problem persists for the root radius $r_d(w)$ where $p'(w)$ vanishes; rotation of the variable $p(x) \leftarrow t(x) = p(ax)$ for $|a| = 1$ does not fix it, but the bound $r_d \le |p_0/p_d|^{1/d}$ and shifts $p(x) \leftarrow t(x) = p(x-c)$ for $c \ne 0$ should fix it.

Furthermore, $\frac{1}{r_d(c)} \le |\mathrm{NIR}(c)| = \frac{1}{d}\left|\sum_{j=1}^{d} \frac{1}{c-z_j}\right|$ by virtue of (10.3), and so the approximation to the root radius $r_d(c,p)$ is poor if and only if severe cancellation occurs in the summation of the d roots.

Formal probabilistic estimation is hard, but intuitively such a cancellation only occurs for a very narrow class of polynomials p. If so, the above non-costly test certifies inclusion for a very large class of polynomials p.

More than 25% of all e/i tests in a subdivision process, however, output exclusion:

Theorem 10.5. *Let σ_i suspect squares enter the i-th subdivision step for $i = 1, \dots, t+1$. Let it stop at f_i of them, discard d_i and further subdivide $\sigma_i - d_i - f_i$ of them. Write $\sigma_1 := 4$, $\sigma_{t+1} := 0$, $\Sigma := \sum_{i=1}^{t} \sigma_i$, $D := \sum_{i=1}^{t} d_i$, and $F := \sum_{i=1}^{t} f_i$. Then $3\Sigma = 4(D + F - 1)$ and $D > \Sigma/4$.*

Proof. Notice that $\sigma_{i+1} = 4(\sigma_i - d_i - f_i)$ for $i = 1, 2, \dots, t+1$, sum these equations for all i, substitute $\sigma_1 = 4$ and $\sigma_{t+1} = 0$, and obtain $3\Sigma + 4 \geq 4D + 4F$. Represent subdivision process by a tree with $F \leq m$ leaves and other nodes representing components made up of suspect squares, to obtain pessimistic lower bounds $\Sigma + 1 \geq 2m \geq 2F$ and hence $D \geq \Sigma/4 + 1/2$. □

11 Our e/i tests

11.1 Overview

Map (10.4) reduces any e/i test to the test for the unit disc $D = D(0,1)$ where $\#(D) \leq m$ for a fixed m, $1 \leq m \leq d$.

Now recall from [1] that

$$\#(D) = \frac{1}{2\pi \mathbf{i}} \int_{C(0,1)} (\mathrm{NIR})(\mathrm{x})\ \mathrm{dx}, \tag{11.1}$$

that is, $\#(D)$ is the average of $\mathrm{NIR}(x) = \frac{p'(x)}{p(x)}$ on the unit circle $C(0,1)$.

Our Alg. 11.2 of Sec. 11.4 evaluates $\mathrm{NIR}(x)$ at q fixed equally space points of the unit circle $C(0,1)$, and by virtue of Thm. 11.2 of Sec. 11.4 the integer $\#(D)$ is exceeded by 1 and hence equal 0 if $|\mathrm{NIR}(x)| \leq \frac{1}{2\sqrt{q}}$ at $q > d$ equally space points of $C(0,1)$. This certifies exclusion at HLA cost q for any $q > d$.

Moreover, Alg. 11.3 of Sec. 11.5 modifies Alg. 11.2 to certify exclusion for a disc with m roots at HLA cost q for any $q > m$ if $|\mathrm{NIR}(x)| \leq \frac{1}{3\sqrt{q}}$.

The following example shows that we cannot certify exclusion by applying these algorithms for $q = m - 1$, in particular for $q = d - 1$ if $m = d$.

Example 11.1. *Let $q = m - 1$, $p(x) = (1 - ux^{d-m})(x^m - dx + u)$, $u \approx 0$ but $u \neq 0$. First let $m = d$. Then $p(x) = (1 - u)(x^m - dx + u)$, $v_g = 0$ for v_g of (11.12) and $g = 0, 1, \dots, q-1$, while $p(y) = 0$ for some $y \approx 0$, and Algs. 11.2 and 11.3 wrongly certify exclusion in this case.*

This example loses its power if we evaluate $\mathrm{NIR}(x)$ at q random rather than fixed points on the circle $C(0,1)$, and this leads us to **Conjecture 1** that such an extension of Alg. 11.2 *certifies exclusion with a high probability (whp)* for $q = o(m)$, but the example of the polynomials $p(x) = (\frac{x \pm 1}{2})^d$, pointed out to us by Oren Bassik of the Graduate Center of CUNY, shows some difficulty with proving Conjecture 1.

In view of bound (11.1) we can see, however, that such counter-examples occur with a lower probability (wlp) in the space of all polynomials of degree d. This leads us to **Conjecture 2** that Conjecture 1 holds whp under a proper randomization model in the space of polynomials of degree d.

Conjecture 2 has strong empirical support because in extensive experiments in [51, 54] a variant of Alg. 11.2 has never failed for q of order $\log^2(d)$.

The customary model of polynomials with random coefficients, however, is not of much use here, because we must apply e/i tests to the discs $D(c, \rho)$ for various pairs of c and ρ, while map (10.4) destroys all customary probability distributions in the space of the coefficients of p.

With some sweat we have finally succeeded in proving Conjecture 2 under the following model:

Random Root Model: the zeros of p are independent identically distributed (iid) random variables, sampled under the uniform probability distribution on the disc $D(0,R)$ for a sufficiently large positive R.

By virtue of Remark 11.1 of Sec. 11.3, we yield *Las Vegas randomization*, that is, verify correctness of the output at a dominated computational cost.

This, however, is not the full story of devising our e/i test yet. Namely, we certify exclusion at a low cost under the above upper bounds on $|\mathrm{NIR}(x)|$, but if $|\mathrm{NIR}(x)|$ is not small, we need further study to certify σ-soft inclusion for $\sigma < \sqrt{2}$.

That study exploits root-lifting: by virtue of claim (i) of Observation 10.1, $\#(D)$ for polynomials $p_h(x)$ is invariant in h, and we apply our e/i tests to the polynomial $p_h(x)$ for a sufficiently large integer h of order $\log(d)$.

If $|\mathrm{NIR}_{p_h}(x)|$ is small, we can certify exclusion for the polynomial $p_h(x)$ and hence, by virtue of claim (i) of Observation 10.1, for the polynomial $p(x)$ as well. Unless the value $|\mathrm{NIR}(x)|$ is small, Thm. 10.3 implies that $\#(D(0,\sigma)) > 0$ for $\sigma = (\frac{d}{v})^{1/2^h}$, exceeded by 1.2, say, for sufficiently large integers h of order $\log(d)$.

11.2 A randomized e/i test: an algorithm

Next we perform an e/i test where we evaluate $\mathrm{NIR}_{t_h}(x)$ at a single random point $x_1 \in C(0,1)$ and then prove success whp under the Random Root Model. Surely, we can further increase the probability of success by reapplying the algorithm at new iid random points sampled from the unit circle $C(0,1)$.

Algorithm 11.1. A randomized e/i test.[31]

INPUT: a polynomial $p(x)$ of a degree d, a real $\sigma > 1$ and a positive integer h, both to be specified in the next subsection, a complex c, and a positive ρ such that $|c| + \rho \le R$ for R of the Random Root Model.

COMPUTATIONS: (i) Sample a random value x_1 under the uniform probability distribution on the unit circle $C(0,1)$, (ii) Compute the value

$$v_1 := \mathrm{NIR}_{t_h}(x_1) = \Big|\frac{t'_h(x_1)}{t_h(x_1)}\Big|. \tag{11.2}$$

(iii) Stop and certify σ-soft inclusion, that is, $\#(D(0,\sigma)) > 0$, if $1 + d/v_1 \le \sigma^h$. Otherwise, stop and claim exclusion, that is, $\#(D(0,1)) = 0$.

Observation 11.1. *At HLA cost hq Alg. 11.1 samples the values of q iid random variables before it stops.*

11.3 A randomized e/i test: analysis under the Random Root Model

Eqn. (10.3) implies that map (10.4) only scales NIR by ρ.

Next we estimate the probability under the Random Root Model that the algorithm wrongly outputs exclusion, in which case $\#(D(0,1)) > 0$, that is, $z_j \in D(c,\rho)$ for some j in the range $1 \le j \le d$.

[31] Before applying computations below, one may first approximate NIR(0) based on (10.11) and then output inclusion if the approximation exceeds d.

Theorem 11.1. *Under the Random Root Model, Alg. 11.1 is correct with a probability at least* $1 - P$*, where*

$$P \le \gamma d \frac{\rho^2}{R^2} \text{ for } \gamma := \max\{\alpha\nu)^{2/h}, \frac{1}{(\alpha-1)^2}\},\ \nu = \frac{8d}{\sigma^h - 1}, \tag{11.3}$$

and α *and* σ *of our choice. In particular*

$$P \le \delta\rho^2/R^2 \tag{11.4}$$

for any fixed positive δ *if*

$$(\alpha-1)^2 \ge d/\delta \text{ and } \sigma^h - 1 \ge 8d^2\alpha/\delta, \tag{11.5}$$

which can be ensured for h *of order* $\log(d/\delta)$ *if, say,* $\sigma = 1.2$*.*

Proof. Bounds (10.12) and (10.13) imply that $r_d(x_1, t_h) \le d/v_1$ for $r_d(0, t_h) \le 1 + r_d(x_1, t_h)$ for $|x_1| = 1$. Hence $r_d(0, t_h) \le 1 + d/v_1 \le \sigma^h$ if Alg. 11.1 claims σ-inclusion. Then claim (iii) of Observation 10.1 implies that $r_d(0) \le \sigma$, and so *Alg. 11.1 outputs* σ*-soft inclusion correctly.*

So Alg. 11.1 can be incorrect only where it outputs exclusion for a disc $D(c,\rho)$ containing a zero of p.

Let this zero be z_1, say. Then fix the values of the random variables $z_2, \dots, z_d$ and $x_1 \in C(0,1)$ and let γ denote the area (Lebesgue's measure) of the set $\mathbb{S}$ of the values of the variable z_1 for which

$$v_1 = |\mathrm{NIR}_{t_h}(x_1)| \le \frac{d}{\sigma^h - 1}. \tag{11.6}$$

Then Alg. 11.1 applied to the disc $D(c,\rho)$ is incorrect with a probability at most

$$P = \frac{\gamma d}{\pi R^2} \tag{11.7}$$

where the factor of d enables us to cover the cases where the value of any of the d iid random variables $z_1, \dots, z_d$, and not necessarily z_1, lies in the disc $D(c,\rho)$.

Fix a value $z' \in \mathbb{S}$ lying at the maximal distance from c, let z denote another value in $\mathbb{S}$, write $y := (\frac{z-c}{\rho})^h$, $y' := (\frac{z'-c}{\rho})^h$, and $\Delta := y' - y$, and notice that $\max\{|y|, |y|'\} \le 1$ because $z_1 \in D(c,\rho)$ by assumption, and in particular $z, z' \in D(c,\rho)$.

Furthermore, write NIR and NIR$'$ to denote the values $\mathrm{NIR}_{t_h}(x_1)$ for $z_1 = z$ and $z_1 = z'$, respectively, and obtain from bounds (11.6) that

$$\max\{|\mathrm{NIR}|, |\mathrm{NIR}'|\} \le \frac{d}{\sigma^h - 1},$$

and so

$$|\Delta_{t_h}| \le \frac{2d}{\sigma^h - 1} \text{ for } \Delta_{t_h} := \mathrm{NIR}' - \mathrm{NIR}. \tag{11.8}$$

Now let $y_j := (\frac{z_j - c}{\rho})^h$, for $j = 1, \dots, d$, denote the d zeros of $t_h(x)$ and deduce from (10.3) that

$$|\mathrm{NIR}| = \Big|\frac{1}{x_1 - y} + S\Big| \text{ and } |\mathrm{NIR}'| = \Big|\frac{1}{x_1 - y'} + S\Big| \text{ for } S := \Big|\sum_{j=2}^{d} \frac{1}{x_1 - y_j}\Big|.$$

Hence

$$|\Delta_{t_h}| = |\mathrm{NIR} - \mathrm{NIR}'| = \Big|\frac{1}{x_1 - y} - \frac{1}{x_1 - y'}\Big| = \frac{|\Delta|}{|(x_1 - y)(x_1 - y')|}. \tag{11.9}$$

Recall that $x_1 \in C(0,1)$, while $y, y' \in D(0,1)$ by assumption, and so

$$\max\{|x_1 - y|, |x_1 - y'|\} \le 2.$$

Combine this inequality with (11.9) and deduce that

$$|\Delta_{t_h}| \ge |\Delta|/4.$$

Combine the latter bound with (11.8) and obtain

$$|\Delta| = |y' - y| \le \nu \text{ for } \nu := \frac{8d}{\sigma^h - 1}.$$

that is,

$$y = \Big(\frac{z - c}{\rho}\Big)^h \in D := D(y', \nu) \text{ for } \nu = \frac{8d}{\sigma^h - 1}.$$

Now fix a reasonably large $\alpha > 1$ and separately consider two cases where either (i) $|y'| < \alpha\nu$ or (ii) $|y'| \ge \alpha\nu$.

The distance $|z - c| = |y|^{1/h}\rho$ reaches its maximum for $z = z'$, $y = y'$, by assumption, and so in case (i) it holds that

$$|z - c| \le |z' - c| = |y'|^{1/h} \le (\alpha\nu)^{1/h}\rho,$$

and then we can bound γ of (11.7) as follows:

$$\gamma \le \pi(\alpha\nu)^{2/h}\rho^2. \tag{11.10}$$

In case (ii) the point y lies in one of the h components $W_1, \dots, W_h$ output by the map $W \leftarrow D^{1/h}$, where each component is associated with one of the hth roots y'.

Let $y = y' + u$ be a point of the disc D, that is, let $|u| \le \nu$, let $(y')^{1/h}$ and $y^{1/h}$ lie in the same component W_g, and then obtain that

$$\nabla := (y')^{1/h} - y^{1/h} = (y')^{1/h}(1 - (1 - u/y')^{1/h})$$

where $|u| \le \nu$ and $|y'| \ge \alpha\nu$. Hence $|u/y'| \le 1/\alpha$, and deduce from Taylor's expansion of $(1-1/\alpha)^{1/h}$ that $|\nabla| \le \frac{|y'|}{\alpha-1}$. Hence $|\nabla| \le \frac{1}{\alpha-1}$ because $|y'| \le 1$.

Now recall that $\nabla\rho = z' - c - (z - c) = z' - z$ and conclude that in case (ii)

$$|z' - z| \le \frac{\rho}{\alpha - 1},$$

and then we can bound γ of (11.7) as follows:

$$\gamma \le \frac{\pi\rho^1}{(\alpha - 1)^2}.$$

Combine this bound with (11.7) and (11.8) and obtain the following bound on the error probability: $P \le \max\{(\alpha\nu)^{2/h}, \frac{1}{(\alpha-1)^2}\}\frac{d\rho^2}{R^2}$ for $\nu = \frac{8d}{\sigma^h-1}$, as we claimed. □

Corollary 11.1. *Under Random Root Model one can solve Problem 1mm at Las Vegas randomized HLA cost in $O(mb\log(d/\delta))$ with overall error probability at most δ for any fixed positive δ.*

Proof. Notice that the overall area of all suspect squares eliminated by subdivision iterations cannot exceed the area R^2 of an initial suspect square of these iterations, while the areas of the covering disc of any square exceeds the area of the square precisely by a factor of $\sqrt{2}$. □

Remark 11.1. Root-finding with error detection and correction. *A randomized root-finder can lose some roots, although with a low probability, but we can detect such a loss at the end of root-finding process, simply by observing that among the m roots lying in an input disc only $m - w$* tame *roots have been closely approximated,*[32] *while $w > 0$* wild *roots remain at large. Then we can recursively apply the same or another root-finder until we approximate all m roots.*[33]

11.4 Basic deterministic e/i test

To simplify exposition, assume in this section that we compute rather than approximate the values $p'(x)$, $\mathrm{NR}_{p_h}(x)$, and $\mathrm{NIR}_{p_h}(x)$ and that we have fixed a together imply that $r_d(0,p) \le \sigma$. Otherwise proceed as follows.

Algorithm 11.2. Basic deterministic e/i test.

INPUT: a polynomial $p(x)$ of a degree d and σ, $1 < \sigma < \sqrt{2}$.

INITIALIZATION: Fix two integers $q > d$ and $h := \lceil \log_\sigma(1+2d\sqrt{q})\rceil$, such that $\sigma^h \le 1+2d\sqrt{q}$.

COMPUTATIONS: For $g = 0, 1, \dots, q-1$ recursively compute

$$v_g := \Big|\frac{p_h'(\zeta^g)}{p_h(\zeta^g)}\Big| \text{ for } g = 0, 1, \dots \text{ and } \zeta \text{ of (1.3).} \tag{11.11}$$

Stop and certify σ-soft inclusion $\#(D(0,\sigma)) > 0$ if $1 + d/v_g \le \sigma^h$ for an integer $g < q$. Otherwise stop and certify exclusion, that is, $\#(D(0,1)) = 0$.

Bounds (10.12) and (10.13) imply that $r_d(\zeta^g, p_h) \le d/v_g$ for all g and $r_d(0,p_h) \le 1 + r_d(\zeta^g, p_h)$ for $|\zeta| = 1$. Hence $r_d(0,p_h) \le 1 + d/v_g \le \sigma^h$ if Alg. 11.2 claims σ-inclusion. Then n claim (iii) of Observation 10.1 implies that $r_d(0) \le \sigma$, and so *Alg. 11.2 outputs σ-soft inclusion correctly.* Next we prove that it also *correctly claims exclusion.*

Lemma 11.1. *A polynomial p has no roots in the unit disc $D(0,1)$ if $|p(\gamma)| > 2\max_{w\in D(0,1)} |p'(w)|$ for a complex $\gamma \in C(0,1)$.*

Proof. By virtue of Taylor-Lagrange's theorem $p(x) = p(\gamma)+(x-\gamma)p'(w)$, for any pair of $x \in D(0,1)$ and $\gamma \in C(0,1)$ and for some $w \in D(0,1)$. Hence $|x-\gamma| \le 2$, and so $|p(x) \ge |p(\gamma)| - 2\ |p'(w)|$. Therefore, $|p(x)| > 0$ for γ satisfying the assumptions of the lemma and for any $x \in D(0,1)$. ☐

Theorem 11.2. *$\#(D(0,1)) = 0$ for a black box polynomial $p(x)$ of a degree d if*

$$\frac{1}{v} > 2\sqrt{q} \text{ for } q > d,\ v := \min_{g=0}^{q-1} v_g, \text{ and } v_g \text{ of (11.11).} \tag{11.12}$$

Proof. Bounds (11.11) and (11.12) together imply that $|p'(\zeta^g)|/v \le |p(\zeta^g)|$ for $g = 0, \dots, q-1$ and hence $2\sqrt{q}||(p'(\zeta^g))_{g=0}^{q-1}||_\infty < \widehat{P}$ for $\widehat{P} := ||(p(\zeta^g))_{g=0}^{q-1}||_\infty$. Combine this bound with (10.2) to obtain

$$2||(p'(\zeta^g))_{g=0}^{q-1}||_2 < \widehat{P}. \tag{11.13}$$

Pad the coefficient vector of the polynomial $p'(x)$ with $q-d-1$ initial coordinates 0 to arrive at q-dimensional vector $\mathbf{p}'$. Then $(p'(\zeta^g))_{g=0}^{q-1} = F\mathbf{p}'$ for the $q\times q$ matrix $F := (\zeta^{ig})_{i,g=0}^{q-1}$ of discrete Fourier

[32]We call a complex point c a tame root for a fixed error tolerance ϵ if it is covered by an isolated disc $D(c,\epsilon)$. Given such a disc $D(c,\rho)$, we can readily compute $\#(D(c,\rho))$ (cf. [99, 98, 51]).

[33]These observations are valid for any root-finder. In the case of subdivision root-finders we can detect the loss of a root at a subdivision step as soon as we notice that at that step the indices of all suspect squares sum to less than m.

transform. Substitute $F\mathbf{p}'$ for $(p'(\zeta^g))_{g=0}^{q-1}$ in (11.13) and obtain $2||F\mathbf{p}'||_2 < \widehat{P}$. Hence $2\sqrt{q}||\mathbf{p}'||_2 < \widehat{P}$ because the matrix $\frac{1}{\sqrt{q}}F$ is unitary (cf. [42, 89]). It follows (cf. (10.2)) that $2||\mathbf{p}'||_1 < \widehat{P}$. Since $\max_{x\in D(0,1)}|p'(x)| \le ||\mathbf{p}'||_1$, deduce that $2\max_{x\in D(0,1)}|p'(x)| < \widehat{P} = ||(p(\zeta^g))_{g=0}^{q-1}||_\infty$. Complete the proof by combining this bound with Lemma 11.1. $\square$

Now, if Alg. 11.2 outputs exclusion, then $1+\frac{d}{v_g} > \sigma^h$ for $g = 0, 1, \dots, q-1$, where $\sigma^h > 1+2d\sqrt{q}$ according to initialization of Alg. 11.2. Hence $\frac{1}{v_g} > 2\sqrt{q}$, and so the assumptions of Thm. 11.2 hold for $p_h(x)$ replacing $p(x)$. Hence the theorem implies that the disc $D(0,1)$ contains no zeros of $p_h(x)$, but then it contains no zeros of $p(x)$ as well (see Observation 10.1 (iii)), and so exclusion is correct.

Readily verify the following estimates for the HLA cost of Alg. 11.2.

Theorem 11.3. *Alg. 11.2 is a σ-soft e/i test, certifying inclusion at the HLA cost $2hj$ or exclusion at HLA cost $2hq$ for integers $q > d$, j, such that $1 \le j \le q$, and $h := \lceil \log_\sigma(1+2d\sqrt{q}) \rceil$, $h = O(\log(d))$ if $\sigma - 1$ exceeds a positive constant.*

11.5 A modified deterministic e/i test

Consider Problem $1bb_m$ in a θ-isolated disc D where θ is large. Then equation (10.4) implies that the impact of the $d-m$ external zeros of p on the values of NIR(x) for $x \in D$ is negligible for sufficiently large values θ, and then we can safely modify Alg. 11.2 to let it evaluate NIR at $q = m+1$ rather than $d+1$ points and still to obtain correct output. It turns out that we can ensure this by choosing θ sufficiently large even where $\log(\frac{1}{\theta}) = O(\log(d))$. Next we supply quantitative estimates.

Algorithm 11.3. A modified deterministic e/i test.

INPUT: a black box polynomial $p(x)$ of a degree d, two integers m and $q > m$, and real σ and $\theta > 1$ such that $1 < \sigma < \sqrt{2}$,

$$|z_j| \le 1 \text{ for } j \le m,\ |z_j| > \theta \text{ for } j > m,$$

$$\frac{d-m}{\theta^h - 1} \le \frac{1}{6\sqrt{q}} \text{ for } h := \lceil \log_\sigma(1+6m\sqrt{q}) \rceil \text{ such that } \sigma^h \ge 1 + 6m\sqrt{q}.$$

COMPUTATIONS: Recursively compute the values v_g of (11.11) for $g = 0, 1, \dots, q-1$. Certify σ-soft inclusion, that is, $\#(D(0,\sigma)) > 0$, if $v_g \ge \frac{1}{3\sqrt{q}}$ for some integer $g < q$. Otherwise certify exclusion, that is, $\#(D(0,1)) = 0$.

Thm. 10.3, for p_h replacing p, and the assumed bound $\frac{d-m}{\theta^h-1} \le \frac{1}{6\sqrt{q}}$ together imply

$$\frac{m}{r_d(\zeta^g, p_h)} \ge v_g - \frac{d-m}{\theta^h - 1} \ge v_g - \frac{1}{6\sqrt{q}} \text{ for all } g.$$

Alg. 11.3 claims inclusion if $v_g \ge \frac{1}{3\sqrt{q}}$ for some g, but then

$$\frac{m}{r_d(\zeta^g, p_h)} \ge \frac{1}{6\sqrt{q}},\ r_d(\zeta^g, p_h) \le 6m\sqrt{q},$$

and hence

$$r_d(0, p_h) \le 1 + 6m\sqrt{q}\ \le \sigma^h \text{ for } |\zeta| = 1.$$

Now Observation 10.1 (iii) implies that *Alg. 11.3 claims inclusion correctly.*

Next prove that it also *claims exclusion correctly.*

Theorem 11.4. *Let a polynomial $p(x)$ of a degree d have exactly m roots in the disc $D(0,\theta)$ for*

$$\theta^h \geq 1 + 6(d-m)\sqrt{q},\ \ h := \lceil \log_\sigma(1+6m\sqrt{q})\rceil,\ \text{and an integer } q > m$$

and let $v < \frac{1}{3\sqrt{q}}$ for v of (11.12). Then $\#(D(0,1)) = 0$.

Proof. Let $p(x)$ and its factor $f(x)$ of degree m share the sets of their zeros in $D(0,\theta)$. Then (10.3) implies that $\Big|\frac{f'(x)}{f(x)}\Big| < \Big|\frac{p'(x)}{p(x)}\Big| + \frac{d-m}{\theta-1}$ for $|x| = 1$, and so $\max_{g=0}^{q-1}\Big|\frac{f'(\zeta_q^g)}{f(\zeta_q^g)}\Big| < v + \frac{d-m}{\theta-1}$. Recall that $\frac{d-m}{\theta-1} \leq \frac{1}{6\sqrt{q}}$ and $v < \frac{1}{3\sqrt{q}}$ by assumptions.

Combine these bounds to obtain $\max_{g=0}^{q-1}\Big|\frac{f'(\zeta_q^g)}{f(\zeta_q^g)}\Big| < \frac{1}{2\sqrt{q}}$. To complete the proof, apply Thm. 11.2 for $f(x)$ replacing $p(x)$ and for m replacing d. □

Readily extend Thm. 11.3 to Alg. 11.3 by decreasing its bounds on q and h provided that $\#(D(0,\theta)) = m$ for $\theta \geq 1 + 6(d-m)\sqrt{q}$.

Theorem 11.5. *Suppose that $q > m := \#(D(0,\theta))$, $\sigma^h > 1 + 6m\sqrt{q}$, $\theta \geq 1 + 6(d-m)\sqrt{q}$, and $h := \lceil \log_\sigma(1+6m\sqrt{q})\rceil$, so that $h = O(\log(m))$ if $\sigma - 1$ exceeds a positive constant. Then Alg. 11.3 is a σ-soft e/i test: it certifies either inclusion or exclusion at HLA cost $2hq$ or $2hj$, respectively, where $1 \leq j \leq q$.*

12 Reduction of m-test to e/i tests

We reduce m-test to e/i tests motivated by application in Part III, where the tests are applied to an isolated disc. Our next simple algorithm in Sec. 12.1 achieves this goal, but in Sec. 12.2 we also present a little more involved reduction algorithm that can be applied to any disc.

12.1 Simple reduction to a single e/i test in case of an isolated disc

Clearly, we can immediately reduce the d-test in the unit disc $D(0,1)$ for $p(x)$ to performing 1-test in that disc for the reverse polynomial $p_{\mathrm{rev}}(x)$ of (10.5).

Let us next reduce an m-test $0 < m < d$ to a single application of 1-test provided that an input disc D with m roots is θ-isolated for θ of order $\log(d)$; we can ensure that provision by applying root-lifting or root-squaring, in both cases at HLA cost $O(\log(d))$ if an initial disc is just isolated. For an alternative extension, reproduced from earlier versions of [99], see our Appendix C.

Theorem 12.1. *Let the root sets of two polynomials $f(y)$ and $g(y)$ of degrees m and $d-m$, respectively, lie outside the disc $D(0,1)$ and in the disc $D(0,1/\theta)$, respectively, for a sufficiently large real θ. Then*

$$\mathrm{NIR}_f(z) = \mathrm{NIR}_{fg}(z) - \frac{d-m}{z} + \Delta \text{ for } |z| = 1 \text{ and } |\Delta| \leq \frac{d-m}{\theta-1}.$$

Proof. Notice that $\frac{1}{z-y_j} - \frac{1}{z} = \frac{y_j}{(z-y_j)z}$, recall that $|y_j| \leq \frac{1}{\theta}$ for $\theta > 1$ and all $d-m$ roots $y_1,\dots,y_{d-m}$ of $g(y)$ by assumption, and deduce that

$$\Big|\frac{1}{z-y_j} - \frac{1}{z}\Big| \leq \frac{1}{\theta-1} \text{ for } |z| = 1 \text{ and } j = 1,\dots,d-m. \tag{12.1}$$

By applying (10.3) obtain that $\mathrm{NIR}_{fg}(z) - \mathrm{NIR}_f(z) = \sum_{j=1}^{d-m}\frac{1}{z-y_j}$. Substitute bounds (12.1). □

Next fix an integer $q > m$, represent $p_{\rm rev}(y)$ as the product $f(y)g(y)$ for polynomials $f(y)$ and $g(y)$ of Thm. 12.1, and approximate $\mathrm{NIR}_f(z)$ by $\mathrm{NIR}_{fg}(z) - \frac{d-m}{z}$ for $z = \zeta^g$ and $g = 0, 1, \dots, q-1$. Thm. 12.1 bounds the approximation error by $\frac{d-m}{\theta-1}$.

Now define Alg. $11.2_{f,\rm rev}$ by applying Alg. 11.2 to the factor $f(y)$ of $p_{\rm rev}(y)$, replacing $\mathrm{NIR}(z)$ by the approximation $\mathrm{NIR}_{fg}(z) - \frac{d-m}{z}$ of $\mathrm{NIR}_f(z)$ within Δ, and replacing bound (11.12) by

$$V_f := \max_{g=0}^{q-1} |\mathrm{NIR}_{fg}(\zeta^g) - (d-m)\zeta^{-g}|. \tag{12.2}$$

We can ensure that $V_f < \frac{1}{2\sqrt{q}}$, say, by requiring that $\max_{g=0}^{q-1} |\mathrm{NIR}_{fg}(\zeta^g)| < \frac{1}{4\sqrt{q}}$ and $\frac{d-m}{\theta-1} < \frac{1}{4\sqrt{q}}$. Unless the latter bound is satisfied for a fixed $q > m$, we can ensure it at NR-cost $O(\log(d)$ if $\theta - 1$ exceeds a positive constant: we can ensure this by applying root-lifting or root-squaring.

12.2 m-test via e/i tests for any disc

Our next alternative reduction of m-test for $1 \le m \le d$ to to a bounded number of e/i tests can be applied for any disc, not necessarily isolated. If at least one of the e/i tests fails, then definitely there is a root in a narrow annulus about the boundary circle of an input disc, and hence the internal disc of the annulus would not contain m roots. Otherwise we would obtain a root-free concentric annulus about the boundary circle of an input disc and then reduce m-test to root-counting in an isolated disc (see, e.g., claim (i) of Thm. 3.2).

Algorithm 12.1. Reduction of m-test to e/i tests.

INPUT: a polynomial $p(x)$ of a degree d, two values $\bar\theta > 1$ and $\bar\sigma > 1$, an integer m such that $1 \le m \le d$ and $\#(D(0,\bar\sigma)) \le m$, and two algorithms $\mathbb{ALG}$ 0 and $\mathbb{ALG}$ 1, which a black box polynomial perform $\bar\sigma$-soft e/i test for in any disc and root-counting in an isolated disc, respectively.

INITIALIZATION (see Fig. 5): Fix a positive ρ and compute the integer

$$v = \lceil 2\pi/\phi \rceil \tag{12.3}$$

where ϕ is the angle defined by an arc of $C(0,1)$ with end points at the distance ρ. Then compute the values

$$c_j = \exp(\phi_j \mathbf{i}),\ j = 0, 1, \dots, v, \tag{12.4}$$

such that $|c_j - c_{j-1}| = \rho$ for $j = 1, 2, \dots, v$ and $\phi_j - \phi_{j-1} = \phi$.

OUTPUT: Either an upper bound $r_{d-m+1} < 1 - \rho\frac{\sqrt{3}}{2}$ or one of the two lower bounds $r_{d-m+1} \ge 1 - \bar\sigma\rho$ or $r_{d-m+1} > 1 + (\sqrt{3}-1)\rho$.

COMPUTATIONS: Apply the algorithm $\mathbb{ALG}$ 0 (soft e/i test) to the discs $D(c_j, \rho)$ for $j = 0, 1, \dots, v$. (a) Stop and certify that $r_{d-m+1} \ge 1 - \bar\sigma\rho$, that is, $\#(D(0, 1-\bar\sigma\rho)) < m$ for the open disc $(D(0, 1-\bar\sigma\rho)$, unless the algorithm $\mathbb{ALG}$ 0 outputs exclusion in all its $v+1$ applications.

(b) If it outputs exclusion $v+1$ times, then apply the algorithm $\mathbb{ALG}$ 1 (root-counter) to the θ-isolated disc $D(0,1)$ for

$$\theta := \min\Big\{1 + (\sqrt{3}-1)\rho, \frac{1}{1 - 0.5\rho\sqrt{3}}\Big\}. \tag{12.5}$$

Let $\bar s_0 = \#(D(0,1))$ denotes its output integer. Then conclude that
(i) $r_{d-m+1} \le 1 - \rho\frac{\sqrt{3}}{2}$, and so $\#(D(0, 1-\rho\frac{\sqrt{3}}{2})) = m$ if $\bar s_0 = m$, while
(ii) $r_{d-m+1} \ge 1 + (\sqrt{3}-1)\rho$, and so $\#(D(0, 1+(\sqrt{3}-1)\rho)) < m$ for the open disc $D(0, 1+(\sqrt{3}-1)\rho)$ if $\bar s_0 < m$.

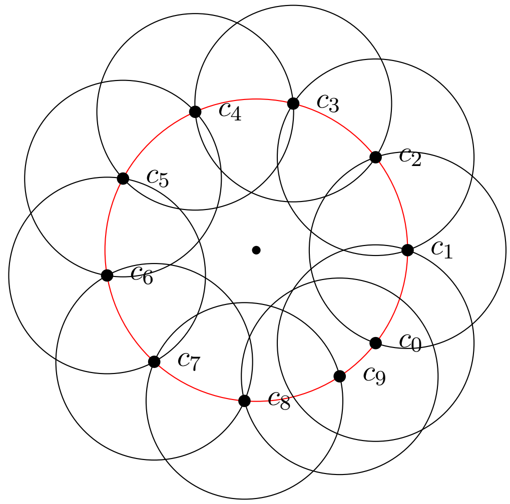


Figure 5: $v = 9$, $|c_j - c_{j-1}| = \rho > |c_9 - c_0|$ for $j = 1, 2, \dots, 9$.

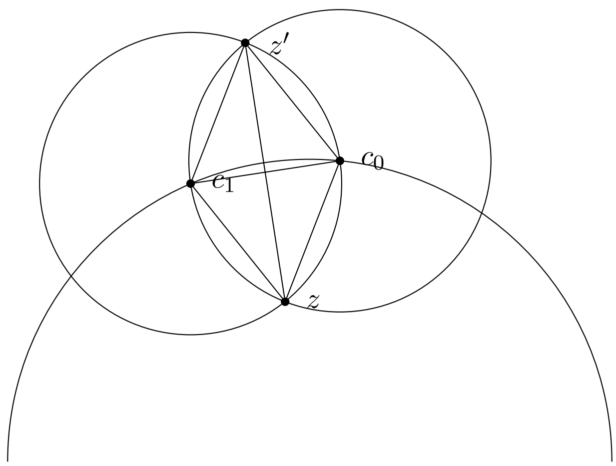


Figure 6: $|c_1 - c_0| = |c_0 - z| = |c_1 - z| = |c_0 - z'| = |c_1 - z'| = \rho$.

To prove correctness of Alg. 12.1 we need the following result.

Theorem 12.2. *For the above values $c_0, \dots, c_v$, and ρ, the domain $\cup_{j=0}^{v-1} D(c_j, \rho)$ covers the annulus $A(0, 1 - 0.5\rho\sqrt{3}, 1 + (\sqrt{3} - 1)\rho)$.*

Proof. Let c_{j-1} and c_j be the centers of two neighboring circles $C(c_{j-1}, \rho)$ and $C(c_j, \rho)$, let z_j and z'_j be their intersection points such that $|z_j| = |z'_j| - |z_j - z'_j|$ for $j = 1, 2, \dots, v$, observe that the values $|z_j - z'_j|$, $|z_j|$, $|z'_j|$, and $|c_{j-1} - c_j|$ are invariant in j, and hence so is the annulus $A(0, |z_j|, |z'_j|)$ of width $|z_j - z'_j|$ as well; notice that it lies in the domain $\cup_{j=0}^{v-1} D(c_j, \rho)$.

Let us prove that $|z_j - z'_j| = \rho\sqrt{3}$ for $j = 1$, and hence also for $j = 2, \dots, v$.

Write $z := z_1$ and $z' := z'_1$ and observe (see Fig. 6) that $|c_1 - c_0| = |c_0 - z| = |c_1 - z| = |c_0 - z'| = |c_1 - z'| = \rho$, and so the points c_0, c_1, and z are the vertices of an equilateral triangle with a side length ρ. Hence $|z - z'| = \rho\sqrt{3}$. The circle $C(c_j, \rho)$ intersects the line segment $[z_j, z'_j]$ at a point y_j such that $|y_j - z_j| > \rho\sqrt{3}/2$ and $|y_j - z'_j| > (\sqrt{3} - 1)\rho$ for $\rho < 1$, and this implies the theorem if $|c_v - c_0| = \rho$. If, however, $|c_v - c_0| < \rho$ and if $C(c_0, \rho) \cap C(c_v, \rho) = \{\bar{z}, \bar{z}\}$, then $|\bar{z} - \bar{z}'| > |z - z'|$, and the theorem follows in this case as well. $\square$

Corollary 12.1. *The disc $D(0,1)$ is θ-isolated for θ of (12.5) provided that the discs $D(c_j,\rho)$ contain no roots for $j=0,1,\ldots,v$.*

Correctness of Alg. 12.1. (i) If the algorithm $\mathbb{ALG}$ 1 outputs inclusion in at least one of its applications, then one of the discs $D(c_j,\bar{\sigma}\rho)$ contains a root, and then Alg. 12.1 correctly reports that $r_{d-m+1}\geq 1-\sigma\bar{\rho}$.
(ii) Otherwise Cor. 12.1 holds, and then correctness of Alg. 12.1 follows from correctness of $\mathbb{ALG}$ 1.
Softness is at most $\frac{1}{1-\bar{\sigma}\rho}$, $\frac{1}{1-0.5\rho\sqrt{3}}$, and $1+(\sqrt{3}-1)\rho$ in cases (a), (b,i), and (b,ii), respectively.
Computational complexity. Alg. 12.1 invokes the algorithm $\mathbb{ALG}$ 0 at most $v+1$ times for v of (12.3). If $\mathbb{ALG}$ 0 outputs exclusion $v+1$ times, then the algorithm $\mathbb{ALG}$ 1 is invoked.

Remark 12.1. *One can modify Alg. 12.1 as follows (see [54]): instead of fixing a positive ρ and then computing v of (12.3) fix an integer $v>0$, apply the algorithm $\mathbb{ALG}$ 0 at v equally spaced points $c_j=\exp(\frac{2j\pi\mathbf{i}}{v})$, $j=0,1,\ldots,v-1$, let $\phi=\frac{2\pi}{v}$, and then define ρ. We also arrive at this algorithm if we compute or guess this value ρ and then apply Alg. 12.1, but notice that $c_v=c_0$ for c_j of (12.4) and only invoke the algorithm $\mathbb{ALG}$ 0 for $c_0,\ldots,c_{v-1}$.*

13 Precision of computing

The arithmetic time $\mathcal{A}$ of our root-finders is roughly the product of their HLA cost and ops count for the evaluation of $p(x)$ (cf. (10.1)). Their Boolean cost is $O(\mathcal{A}\mu(s))=\tilde{O}(\mathcal{A}s)$ for precision s of their computations and $\mu(s)=O(s\log(s)\log(\log(s)))$.

Next we deduce the same precision bound $O(\log(d))$ as in [68], even though instead of *Unlimited Access Assumption* of [68] that we have cost-free access to the values of $p(x)$ computed with no errors by a black box oracle, we adopt **Assumption 0** of *Limited Access* to them within a relative error bound in $1/d^{\nu}$ for any fixed constant $\nu>0$ of our choice. Moreover, we strengthen the analysis of [68] by allowing computations with rounding errors, estimating their impact, and bounding precision required for representation of $p(x)$. We handle these tasks by sampling x randomly, excluding output errors whp, and detecting and correcting them at a dominated cost (see Remark 11.1).

In our e/i tests we only need to compare the values $|\mathrm{NIR}_{t_h}(x)|$ at some complex points x with some fixed values $v_d\geq 1/d^{\beta}$ for a constant $\beta>0$, and next we estimate that under Assumption 0 we can do this by performing our computations with a precision of $O(\log(d))$ bits.

Multiplication with a precision s contributes a relative error at most 2^s (cf. [20, Eqn. A.3 of Ch. 3]) and otherwise just sums the relative error bounds of the factors if we ignore the impact of higher order errors. Hence we deduce that $h=O(\log(d))$ multiplications of h-lifting increase these bounds at most by a factor of h, which is immaterial for $h=O(\log(d))$ since we allow relative errors of order $1/d^{\nu}$ for a constant $\nu>0$ of our choice.

We only need to apply this analysis to approximation of $\mathrm{NIR}(x)$ for $p_h(x)$ based on expression (10.1). The analysis is certified under the following **Assumption 1** *(Isolation Assumption):* all q evaluation points ζ^g of an e/i test lie at least at the distance $\phi:=\frac{2\pi}{mqw}$ from all zeros of p for any fixed $w>1$ of our choice. We justify using this assumption by applying our e/i tests under random rotation $\zeta^g\mapsto v\zeta^g$ for the random variable v specified in our next theorem. Remark 11.1 implies that this is Las Vegas randomization.

Theorem 13.1. *Let $\phi_g:=\min_{j=1}^{d}|v\zeta^g-z_j|$ and let $\phi:=\min_{g=0}^{q-1}\phi_g$ for the zeros z_j of p, ζ of (1.3) and random variable v sampled from a fixed arc of the circle $C(0,1)$ of length $2\pi/q$ under the uniform probability distribution on that arc. Let $i(D(0,1))=m$. Then $\phi\leq\frac{2\pi}{mqw}$ for $w>1$ with a probability at most $\frac{\pi}{w}$.*

Proof. Let ψ_j denote the length of the arc $C(0,1) \cap D(z_j, \frac{2\pi}{mqw})$, $j = 1, 2, \dots, d$. Then $\psi_j = 0$ for at least $d - m$ integers j, and $\max_{j=1}^{d} \psi_j \le \frac{\pi^2}{mqw}$ for sure. Hence $\sum_{j=1}^{d} \psi_j < \frac{\pi^2}{qw}$, and so $\phi_g \le \frac{2\pi}{mqw}$ for a fixed g with a probability at most $\frac{\pi}{qw}$. Therefore, $\phi \le \frac{2\pi}{mqw}$ with a probability at most $\frac{\pi}{w}$, as we claimed. □

Next we prove that precision of $O(\log(d))$ bits supports our e/i tests under Assumptions 0 and 1 by applying Thm. 13.1 for sufficiently large $w = d^{O(1)}$. We will (i) prove that $|\mathrm{NIR}(z)| = d^{O(1)}$, (ii) approximate this value within $d^{\nu}\delta$ for any fixed constant ν and for δ of (10.1) chosen such that $\log(\frac{1}{\delta}) = O(\log(d))$, and (iii) estimate that such a bound withstands the impact of rounding errors of computations for (10.1). Eqn. (10.3) immediately implies

Theorem 13.2. *Under Assumption 1,* $\max_{g=1}^{q} |\mathrm{NIR}(\zeta^g)| \le d^{\eta}$ *for* $\eta = O(1)$*.*

Corollary 13.1. *A precision of* $\lceil (\eta + \beta) \log_2(d) \rceil$ *bits is sufficient to represent* $\mathrm{NIR}(x)$ *within* $\frac{1}{d^{\beta}}$ *for* $g = 1, 2, \dots, q$*.*

Instead of $\mathrm{NIR}(x) = \frac{t'(x)}{t(x)}$ for $x \in C(0,1)$ we actually approximate $\frac{t'(y)}{t(x)}$ where $|y - x| \le \delta$ and $t'(y)$ is equal to the divided difference of (10.1). Thus we shall increase the above error bound by adding upper bounds α on $|\frac{t'(x)}{t(x)} - \frac{t'(y)}{t(x)}|$ and β on the rounding error of computing $\frac{t(x)-t(x-\delta)}{\delta t(x)}$, for $x = \zeta^g$ and $g = 1, 2, \dots, q$.

In the proof of the next theorem we use the following lemma, which is [68, Fact 3.5].

Lemma 13.1. *For* $t(x) = \prod_{j=1}^{d}(x - y_j)$ *and a non-negative integer* $j \le d$ *it holds that*

$$t^{(j)}(x) = j!\ t(x) \sum_{S_{j,d}} \prod_{j \in S_{j,d}} \frac{1}{x - y_j}$$

where the summation $\sum_{S_{j,d}}$ *is over all subsets* $S_{j,d}$ *of the set* $\{1, \dots, d\}$ *having cardinality* j*.*

Theorem 13.3. *Under the assumptions of Thm. 13.2 it holds that*
$|t(x)\alpha| = |\frac{t(x)-t(x-\delta)}{\delta} - t'(x)| \le |t(x)\frac{(d\xi')^2\delta}{1-d\xi'\delta}|$ *for* $\xi' = \frac{1}{\rho}$*.*

Proof. The claimed bound on $|\xi'|$ follows from (10.3). It remains to apply the first five lines of the proof of [68, Lemma 3.6] with f, α, n, and ξ replaced by t, $-\delta$, d,and w, respectively. Namely, first obtain from Taylor's expansion that $t(x) - t(x - \delta) = \sum_{j=0}^{\infty} \frac{\delta^j}{j!} t^{(j)}(x) - t(x)$.

Substitute the expressions of Lemma 13.1 and obtain

$$t(x) - t(x - \delta) = \delta t'(x) + \sum_{j=2}^{\infty} \delta^j t(x) \sum_{S_{j,d}} \prod_{j \in S_{j,d}} \frac{1}{x - y_j}.$$

Combine this equation with the assumed bounds on $\frac{1}{|x-y_j|}$ and deduce that

$$\Big|\frac{t(x) - t(x - \delta)}{\delta} - t'(x)\Big| \le \Big|\frac{f(x)}{\delta} \sum_{j=2}^{\infty} \delta^j w^j d^j\Big| \le \Big|f(x)\frac{(d\ w)^2\ \delta}{1 - \delta\ d\ w}\Big|.$$

□

Corollary 13.2. *We can ensure that* $\alpha < 1/d^{\bar{\eta}}$ *for any fixed constant* $\bar{\eta}$ *by choosing* δ *of order* $d^{O(1)}$*, represented with a precision of* $O(\log(d))$ *bits.*

Theorem 13.4. *Suppose that the values $t(y)$ have been computed by a black box oracle within a relative error bounds $\nabla(y)$ for $y = x$, $y = x - \delta$, and δ of Cor. 13.2. Then one can ensure that $\beta < \frac{1}{d^{\bar{\eta}}}$ for any fixed constant $\bar{\eta}$ by choosing a proper $\nabla(y)$ of order $1/d^{O(1)}$, represented with a precision of $O(\log(d))$ bits.*

Proof. Write
$\beta\delta = \frac{t(x-\delta)(1+\nabla(x-\delta))}{t(x)(1+\nabla(x))} - \frac{t(x-\delta)}{t(x)} = \frac{t(x-\delta)}{t(x)} \frac{\nabla(x)-\nabla(x-\delta)}{1+\nabla(x)}$.

Taylor-Lagrange's formula implies that
$\frac{t(x-\delta)}{t(x)} = 1 + \frac{t(x-\delta)-t(x)}{t(x)} = 1 + \delta\frac{t'(u)}{t(x)}$, for $u \in [x-\delta, x]$.
Hence $|\frac{t(x-\delta)}{t(x)}| \le 1 + |\delta\alpha'| + |\frac{t'(x)}{t(x)}|$ for $\alpha' = \frac{t'(u)-t(x)}{t(x)}$.

By extending Cor. 13.2 obtain $|\alpha'| \le \frac{1}{d^{O(1)}}$ and as in Cor. 13.1 deduce from (10.3) that $|\frac{t'(x)}{t(x)}| \le \frac{1}{d^{O(1)}}$. Hence $w := |\frac{t(x-\delta)}{t(x)}| \le d^{O(1)}$ and $|\beta\delta| \le |w|\frac{|\nabla(x)|+|\nabla(x-\delta)|}{1-|\nabla(x)|}$.

Now choose $\nabla(y)$ such that $|\nabla(y)| \le \frac{|\delta|}{20|w|}$ for $y = x$ and $y = x-\delta$. Then verify that $\log(\frac{1}{|\nabla(y)|}) = O(\log(d))$, $1 - |\nabla(x)| \ge 1 - \frac{|\delta|}{20|w|} > \frac{19}{20}$, and $|\delta\beta| < \frac{|\delta|}{8}$. □

Combine Cors. 13.1 and 13.2 and Thm. 13.4 to obtain

Corollary 13.3. *Given a black box (oracle) for the evaluation of a polynomial $p(x)$ of a degree d with a relative error of any order in $1/d^{O(1)}$, we can perform our black box subdivision root-finders with a precision of $O(\log(d))$ bits.*

Remark 13.1. *We derived our precision bound $O(\log(d))$ based on operating entirely with NIR(x). For comparison, if we relied on computing $p(x)$ and on the bound $\min_{x:\ |x|=1}(|p(x)|/\sum_{i=0}^{d}|p_i|) \ge (\frac{\theta-1}{2\theta})^d$ of [120, Eqn. (9.4)], we would have only obtained a bound of order m^2d on the computational precision of our root-finders. For another comparison, by extending our study in Sec. 1.2 or in [83, 90] we can deduce lower bound of order bd^2 on the precision of root-finding within error bound $1/2^b$ for any polynomial p of (1.1) given with its coefficients, and we can decrease it to order bd if the zeros of p are pairwise well separated.*

14 Boolean cost of general polynomial root-finding

14.1 A basic result

Next we *estimate the Boolean time of our root-finders applied to a general polynomial* p of (1.1) whose coefficients are given with a precision sufficient to support our root-finders. In this way we give up potential acceleration of our root-finders for polynomials that can be evaluated fast, but we still yield nearly optimal Boolean time estimates where $q(m)m = \tilde{O}(d)$ for $q(m)$ of Thm. 1.7.

The Boolean complexity of our root-finders is defined by their HLA cost and computational precision. We have already estimated HLA cost and in the previous section have reduce the precision estimation to estimation for Boolean complexity of the evaluation of p with a relative error in $1/d^{\nu}$ for any constant ν of our choice. Next we readily solve the latter problem based on [60, Thm. 3.9], which provides an estimate for the Boolean complexity of multipoint polynomial evaluation in [37, 73] (see a historical account in [18, Sec. 4.5]).[34]

[34] Precision of computing, affecting the overhead constants hidden in the above estimates and supporting the latter algorithm, grows fast as d increases; e.g., the IEEE standard double precision of the customary numerical computations is not enough in these algorithms already for $d > 50$; for $b = O(\log(d))$ one can fix this deficiency by applying the Fast Multipole Method of Greengard and Rokhlin [44, 26, 92, 93] to the equivalent task of multiplication of Vandermonde matrix by a vector.

Theorem 14.1. [60, Thm. 3.9]. *Given a positive b, the coefficients of a polynomial $t(x) := \sum_{i=0}^{d} t_i x^i$ such that $||t(x)||_1 = \sum_{i=0}^{d} |t_i| \le 1$, and $q \ge d$ complex points $x_1, \dots, x_q$, in the unit disc $D(0,1)$, one can approximate the values $t(x_1), \dots, t(x_q)$ within $1/2^b$ by using $O(\mu((d\log(d) + q)(b + q)))$ Boolean operations for $\mu(s) = O(s\log(s))\log(\log(s)))$.*

Proof. See [60, Alg. 5.3 and appendix A.3]. ☐

To cover also the case where $q \le d$, represent $t(x) = \sum_{j=0}^{h} t_j(x)x^{jq}$ for $h = \lceil d/q \rceil - 1$ and polynomials $t_j(x)$ of degrees at most q sharing their coefficients with $t(x)$ and evaluate at the points $x_1, \dots, x_q$: (i) the polynomials $p_j(x)$, (ii) the powers x^{jq} for $j = 0, 1, \dots, h$, and finally (iii) $t(x) = \sum_{j=0}^{h} t_j(x)x^{jq}$.

At stages (ii) and (iii) we only perform $O(d)$ ops and increase precision of computing by $O(\log(d)$ bits because $\max_{j=0}^{h} |x_j| \le 1$. Hence overall Boolean cost of the evaluation of $p(x)$ is dominated at stage (i), where we apply h times Thm. 14.1, for d replaced by m, and obtain the Boolean cost bound $O(\mu((b+q)d\log(q)))$, which extends Thm. 14.1 to the case where $q \le d$.

14.2 Reduction of our root-finders to multipoint polynomial approximation

Given a complex c, a pair of positive b and ρ, and $d+1$ coefficients of a polynomial $p = p(x)$ of (1.1) such that the disc $D(c, \rho)$ is isolated and contains precisely m zeros of p, we approximate all these zeros within $R/2^b$ for $R = |c| + \rho$ by applying subdivision root-finding iterations with e/i tests of Sec. 11. Then every subdivision step is reduced to application of our e/i tests to $\bar{m} = O(m)$ discs $D(c_\lambda, \rho_\lambda)$, where $\rho_\lambda \ge R/2^b$ for all λ. Such a test amounts to approximation of $p(x)$ at q equally spaced points x on each circle $C(c_\lambda, \rho_\lambda)$.

In Thm. 14.1 a polynomial $t(x)$ with $||t(x)||_1 \le 1$ is approximated at some points $x \in D(0,1)$. To ensure the assumption about x, we scale the variable $x \mapsto Rx$ to map the discs $D(c_\lambda, \rho_\lambda\sigma) \mapsto D(\bar{c}_\lambda, \bar{\rho}_\lambda) \subseteq D(0,1)$, for $\bar{c}_\lambda = c_\lambda/R$, $\bar{\rho}_\lambda = \rho_\lambda/R \ge 1/2^b$, and all λ.

Then we write $t(x) := p(Rx)/\psi$ for $\psi = O(R^d)$ such that $||t(x)||_1 = 1$ and approximate $t(x)$ within $1/2^{\bar{b}}$ or equivalently within the relative error bound $1/(|t(x)|2^{\bar{b}})$ at $q(\bar{m})$ points x in $\bar{m}$ e/i tests for $q(m)$ of Thm. 1.7. Due to Cor. 13.3 it is sufficient to ensure the bound $1/(|t(x)|2^{\bar{b}}) = 1/d^{O(1)}$; we achieve this by choosing $\bar{b} \ge \log(\frac{1}{|t(x)|}) + O(\log(d))$.

In our e/i tests we only need to approximate $t(x)$ at the points x lying on θ_λ-isolated circles where $\frac{1}{\theta_\lambda - 1} = O(m)$. Then $\log(\frac{1}{|t(x)|}) = O(d\log(m))$ by virtue of the bound of [120, Eqn. (9.4)]).[35] Substitute $\bar{\rho}_\lambda \ge 1/2^b$ and obtain $\bar{b} = \log(\frac{1}{|t(x)|}) + O(\log(d)) = O(d\log(m) + b)$.

Remark 14.1. *For $m = d$, the above bound on $\log(\frac{1}{|t(x)|})$ is sharp up to a constant factor and is reached at the polynomial $t(x) = (x+1-\frac{1}{m})^d/(2-\frac{1}{m})^d$, $\rho_\lambda = 1$ and $x = -1$. For $m < d$, [120, Thm. 4.5] implies a little stronger bounds $\log(\frac{1}{|t(x)|}) = O(d + m\log(m))$ and hence $\bar{b} = O(m\log(m) + d + b)$ because all discs $D(\bar{c}_\lambda, \bar{\rho}_\lambda)$ lie in θ-isolated unit disc $D(0,1)$ for $\theta - 1$ exceeding a positive constant.*

14.3 Boolean cost of our e/i tests and root-finders

Now apply Thm. 14.1 with q and b replaced by $\bar{q} = O(qm)$ and $\bar{b} = O(b+d+m\log(m))$, respectively, and obtain that the Boolean complexity of our $O(m)$ e/i tests at any fixed subdivision step is in

$$O(\mu(q(m)m + d\log(d))((q(m) + \log(m))m + d + b))) \text{ for } q(m) \text{ of Thm. 1.7.}$$

[35] That bound follows from the estimate $|f| \ge 1/2^d$ of [120, Thm. 4.2], which holds provided that $t(x) = f\prod_{j=1}^{m}(x - x_j)\prod_{j=m+1}^{d}(\frac{x}{x_j} - 1)$, $||t(x)||_1 = 1$, $|x_j| < 1$ for $j \le m$, and $|x_j| > 1$ for $j > m$.

Hence our resulting root-finders run in overall Boolean time

$$\mathbb{B}_{\text{roots}} = O(\mu(q(m)m + d\log(d))((q(m) + \log(m))m + d + b)b)),$$

which includes the time-complexity of the evaluation oracle.

Recall that $\mu(s) = \tilde{O}(s)$ and obtain

$$\mathbb{B}_{\text{roots}} = \tilde{O}((q(m)m + d)(q(m)m + d + b)b). \tag{14.1}$$

We have deduced this bound for a general polynomial $p(x)$ of (1.1) represented with its coefficients. If $p(x)$ can be evaluated fast, e.g., is a sum of a small number of shifted monomials, then the root-finder is accelerated accordingly.

15 Computation of a disc covering all zeros of $p(x)$

15.1 Introductory comments

Next we compute a range $[\rho_-, \rho_+]$ that brackets all root radii or, equivalently, compute an upper estimate for the largest root radius r_1 and a low estimate for the smallest root radius r_d of $p(x)$.

We can apply the estimates of Sec. 10.4 to the reverse polynomial $p_{\text{rev}}(x)$ if its coefficients are available, but next we study this problem for a black box polynomial. Its solution is required, e.g., in order to initialize algorithms for all d roots by means of subdivision or functional iterations (cf. [14, 21, 128]).

Suppose that we have an algorithm that computes a disc $D(0, \rho_-)$ that contains no zeros of a black box polynomial $p(x)$; clearly, $\rho_- \le r_d$. By applying this algorithm to the reverse polynomial $p_{\text{rev}}(x)$ we will obtain the upper bound $1/\rho_{-,\text{rev}}$ on r_1.

It remains to describe computation of the disc $D(0, \rho_-)$. Empirically we can do this by applying the algorithms of [100, 43], but next we study formally some alternative algorithms.

To obtain a range $[\rho_-, \rho_+]$ that brackets all root radii, we need upper estimate for the root radii r_1 and r_d, but next we only estimate r_d because we can extend them to r_1 by estimating r_d for the reverse polynomial $p_{\text{rev}}(x)$. We obtain a low bound on r_d by computing a disc that contains no zeros of a black box polynomial $p(x)$.

Hereafter let $p(x)$ denote a black box polynomial.

15.2 Probabilistic solution

Seek a disc $D(c, \rho)$ containing no zeros of $p(x)$ in its interior, where we can choose any positive ρ . Sample a random c in the unit disc $D(0, 1)$ under the uniform probability distribution in $D(0, 1)$.

The disc has area π, while the union of the d discs $D(x_j, \rho)$, $j = 1, 2, \ldots, d$ has area at most $\pi d\rho^2$. Therefore, c lies in that union with a probability at most $d\rho^2$, which we can decrease at will by decreasing ρ. Unless c lies in such a union, we obtain a desired disc $D(c, \rho)$.

By applying our e/i tests of Secs. 11.2 – 11.5 to this disc, we can verify correctness of its choice.

15.3 Deterministic solution

We can deterministically compute a desired disc $D(0, \rho_-)$ at a higher cost, based on a soft e/i test.

Namely, apply 1-test to $d + 1$ triples of parameters c_i, θ_i, and ρ_i, $i = 0, 1, \ldots, d$ such that the discs $D(c_i, \theta_i\rho_i)$ have no overlap pairwise. Then, clearly, at most d discs contain roots because $p(x)$ has only d zeros, and so definitely at least one of the disc contains no roots. Deterministic e/i of

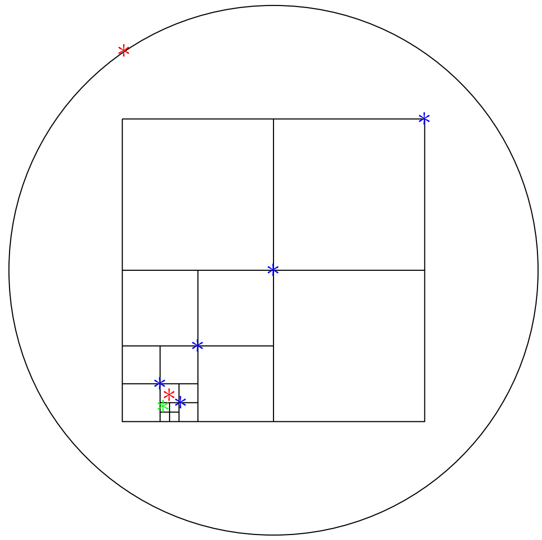


Figure 7: Five blue marked centers of suspect squares converge to a green-marked root cluster with linear rate; two red marked Schröder's iterates converge with quadratic rate.

Secs. 11.4 and 11.5 run at HLA cost $d+1$, and so we obtain a desired disc D at HLS cost at most $(d+1)^2$.

PART III: Compression of a Disc and Acceleration of Subdivision Iterations

16 Overview

The classical subdivision iterations converge to all zeros of a polynomial p with linear convergences rate (see Sec. 1.11), but clearly this is very slow when they process a component $\mathbb{C}$ made up of a single suspect square or up to four of them sharing a vertex. In such cases the algorithms of [114, 88] dramatically accelerate convergence to superlinear rate by means of Newton's or Schröder's iterations (see Fig. 7).

Alternatively one can split $p(x)$ into the product of two factors with the zero sets in and outside a disc covering the isolated component $\mathbb{C}$ (see Part V) and then apply a root-finder to each of the two factors of p of smaller degrees. They, however, do not preserve sparsity of $p(x)$ and may have significantly larger coefficient length than $p(x)$. (Consider, e.g., splitting $p(x) = x^{1000} + 1$ into factors of degrees 300 and 700.) If one of the factors has smaller degree, then we can alternatively approximate this factor f (see Part IV) and can avoid troubles with the other factor by operating with it implicitly as the ratio p/f.

Next we propose and analyze a simple novel alternative based on compression of a disc covering component $\mathbb{C}$ into nearly minimal disc that shares with that component its root set.

We organize our presentation in this part as follows. In the next section we

We will reuse the material of Sec. 10 (Background) of Part II but otherwise will make our presentation self-contained.

17 Component tree and compact components

17.1 Component tree: definition

Break or *partition* the union of all suspect squares at a subdivision step into *components*; a step can break some of them into more than one component. Represent all such partitions with a *component tree* $\mathbb{T}$. At most m components, said to be the leaves of the tree, can be covered by isolated discs of radius at most ϵ for a fixed tolerance $\epsilon \geq 0$. The other vertices are components-parents, each broken by a subdivision step into at least two components-children. The initial suspect square is the root of the tree $\mathbb{T}$. The edges of the tree connect parents and children.

Observation 17.1. *Having at most m leaves, the tree $\mathbb{T}$ has at most $m-1$ other nodes. These upper bounds are* sharp *– attained where a subdivision process defines the complete binary tree $\mathbb{T}$ with m leaves.*

17.2 Edge length, non-adjacent suspect squares, and compact components

The edges of the tree $\mathbb{T}$ represent subdivision steps between partitions of components. Cor. 17.1 will bound the number of such steps – the edge length.

Call a pair of suspect squares *non-adjacent* if they share no vertices.

Call a component *compact* if it contains just a single non-adjacent suspect square, that is, consists of at most four suspect squares sharing a vertex (see Fig. 8).

Observation 17.2. *(See Fig. 2.) The σ-covers of two non-adjacent suspect squares are separated by at least the distance $a\Delta$ for the side length Δ and* separation coefficient

$$a := 2 - \sigma\sqrt{2} > 0 \text{ for } 1 < \sigma < \sqrt{2}.$$

Lemma 17.1. *Let $|z_g - z_h| \geq \sigma\Delta$, for a pair of roots z_g and z_h, the softness σ of e/i tests, and the side length Δ of a suspect square, and let $i > 0.5 + \log_2(m)$. Then in at most i subdivision steps the roots x and y lie in distinct components.*

Proof. Two roots z_g and z_h lying in the same component at the ith subdivision step are connected by a chain of σ-covers of at most m suspect squares, each having diameter $\sigma\Delta/2^{i-0.5}$. Hence

$$|z_g - z_h| \leq m\sigma\Delta/2^{i-0.5},\ i \leq \log_2\left(\frac{m\sigma\Delta}{|z_g - z_h|}\right) + 0.5.$$

Substitute the bound $|z_g - z_h| \geq \sigma\Delta$ and deduce that $m \leq 2^{i-0.5}$ or equivalently $i \leq 0.5 + \log_2(m)$ if the roots z_g and z_h are connected by a chain of suspect squares, that is, lie in the same component. This implies the lemma. □

Corollary 17.1. *A component is broken in at most $1 + \lfloor 0.5 + \log_2(m) \rfloor$ successive subdivision steps unless it is compact.*

17.3 Isolation of compact components

Observation 17.3. *(i) A compact component is surrounded by a frame of discarded squares (see Fig. 8) and hence is covered by a 2-isolated square. (ii) If it stays compact in h successive subdivision steps, then its minimal covering square and disc become 2^{h+1}-isolated and $2^h\sqrt{2}$-isolated, respectively.*

Proof. Readily verify claim (i). Claim (ii) follows because a subdivision step can only thicken the frame of discarded squares while decreasing at least by twice the diameter of any component that stays compact. □

18 ϵ-compression: algorithm and analysis

18.1 Compression algorithm: high level description

Recall that the classical subdivision iterations converge slowly where they successively process only compact components (cf. Fig. 7), but then we accelerate subdivision process by interrupting it with *compression* steps in at most three iterations.

Next we specify compression for a 3.5-isolated unit disc $D = D(0,1)$ covering a component; map (2.2) extends compression to any 3.5-isolated disc $D(c, \rho)$; the map $\rho\epsilon \mapsto \epsilon$ adjusts the error tolerance ϵ to scaling by ρ.

Algorithm 18.1. ϵ-compression (high level description).

INPUT: a positive $\epsilon \le 1$ and a black box polynomial $p(x)$ of a degree d such that[36]

$$i(D(0,1)) \ge 3.5 \text{ and } \#(D(0,1)) \ge 1. \tag{18.1}$$

OUTPUT: an integer m, a complex c', and a positive ρ' such that $m = \#(D(0,1))$, (i) $X(D(c',\rho')) = X(D(0,1))$, and (ii) $\rho' \le \epsilon$ or $8|z_g - z_h| > 5\rho'$ for a pair of roots $z_g, z_h \in D(c',\rho')$.

INITIALIZATION: Fix $\delta \ge 0$ such that $8\delta \le \epsilon \le 1$.

COMPUTATIONS: Compute $m = \#(D(0,1))$, c', ρ' such that

$$|c' - c| \le \delta \text{ and } \rho \le \rho' \le \max\{\epsilon, \rho + 2\delta\} \tag{18.2}$$

for $\rho := r_{d-m+1}(c')$ and some $c \in CH(X(D(0,1)))$.

18.2 Correctness of compression algorithm

First prove correctness in a **simple case** where $\epsilon = \delta = 0$, $c' = c$, $\rho' = \rho$.

In this case $c' = c \in CH(X(D(0,1))) \subset D(0,1)$, and so $\rho' = \rho = r_{d+1-m}(c') \le 2$. Hence $D(c',\rho') \subseteq D(0,3)$.

Furthermore, $\#(D(c',\rho')) = m$ since $\rho' = \rho := r_{d+1-m}(c')$, while $\#(D(0,1)) = m$ by assumption.

Now (18.1) implies that $\#(D(0,3)) = m$, which supports claim (i) for the output of Alg. 18.1.

Next let $|c' - z_g| = \rho := r_{d+1-m}(c')$ for a zero z_g of $p(x)$. Then all m roots in $D(0,1)$ also lie in the disc $D(c',\rho) = D(c',\rho')$ by virtue of claim (i), just proved.

Since $c' = c \in CH(X(D(0,1))$, however, there is a root, z_h, lying in $D(c',\rho')$ but not in $D(z_g,\rho)$, such that $|z_g - z_h| > \rho \ge \rho' - 2\delta$ (cf. (18.2)).

Hence $|z_g - z_h| > 3\rho'/4$ (implying claim (ii) for the output of Alg. 18.1) because $2\delta \le \epsilon/4 < \rho'/4$.

Next extend this correctness proof to the **general case** based on bounds (18.2).

Notice that $c' \in D(0, 1+\delta)$ because $c \in CH(X(D(0,1))) \subset D(0,1)$ and $|c' - c| \le \delta$.

Hence $\rho = r_{d-m+1}(c') \le 2+\delta$, and so $\rho' \le \rho + 2\delta \le 2 + 3\delta$ (cf. (18.2)) and $D(c',\rho') \subseteq D(0, 3+4\delta)$.

Therefore $D(c',\rho') \subseteq D(0,3.5)$ because $8\delta \le \epsilon \le 1$ by assumption.

Now recall that $\#(D(0,3.5)) = \#(D(0,1)) = m$, while $\#(D(c',\rho')) \ge \#(D(c',\rho)) = m$ for $\rho' \ge \rho$ and deduce claim (i) for the output of Alg. 18.1.

Next wlog assume that $c = 0$, $c' = \delta$, and $z_g = c' + \rho$ for a root z_g.

[36]Newton's and Schröder's iterations of [114, 88] accelerate convergence of subdivision iterations to a root cluster in a disc if the disc is $5d^2$-isolated, while (18.1) only requires 3.5-isolation.

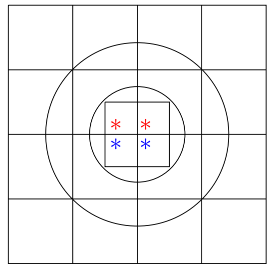


Figure 8: Roots (asterisks) define compact components. A subdivision step halves their diameters and at least doubles isolation of their minimal covering squares and discs.

Then $\Re(z_g) \ge \rho - \delta \ge \rho' - 3\delta$ because $|c'| = |c' - c| \le \delta$ and $\rho' \le \rho + 2\delta$ by assumptions.

Furthermore, $\Re(z_h) < 0$ for some root z_h because $c = 0 \in CH(X(D(0,1)))$. (Actually, $\Re(z_h)$ reaches its maximum $-\frac{z_h}{m-1}$ where z_h is a simple root and $-\frac{z_h}{m-1}$ is a root of multiplicity $m-1$.)

Therefore, $|z_g - z_h| \ge \Re(z_g) - \Re(z_h) > \rho' - 3\delta \ge \rho' - \frac{3}{8}\epsilon$.

This implies claim (ii) for the output of Alg. 18.1 since $\rho' > \epsilon$.

18.3 Elaboration upon the compression algorithm

We can readily compute $m = \#(D(0,1))$ for the isolated input disc $D(0,1)$ of Alg. 18.1, e.g., by applying Alg. 3.1, and we can extend this recipe to any disc $D(c,\rho)$ by applying map (2.2) and Thm. 3.2.

Furthermore, based on Thm. 3.3 we can compute close approximations $\tilde{s}_1$ to the sum s_1 of the m zeros of p lying in that disc and c' to their average s_1/m, lying in the convex hull $CH(D(c,\rho))$, Finally, we can approximate the mth smallest root radius $r_{d-m+1}(c',p)$ by applying our root radius estimation of Sec. 6, based on ℓ-tests for $\ell = 1$ and $\ell = m$, for which we can apply ℓ-tests of Parts II or combine e/i tests of Sec. 11 with reduction of m-test to e/i tests in Sec. 12.

Computation of the sufficiently close approximation of the sum s_1 of the zeros of p and the average s_1/m involves a little higher HLA cost and Boolean time than we need to support Thm. 1.7 (HLA cost bound would get an extraneous term $O(b\log(d)$ and similarly for Boolean time bound), but we fix this deficiency in Sec. 20, where we cover another variant of compression algorithm.

18.4 Counting ϵ-compression steps

c' approximates within ϵ all m roots lying in the disc $D(0,1)$ if $\rho' \le \epsilon$ for the output value ρ' of Alg. 18.1. Then call the compression *final* and stop computations.

Otherwise call the square $S = S(c', \rho')$ suspect and apply to it subdivision iterations.

Extend Lemma 17.1 to the case where $|z_g - z_h| \ge \frac{5}{8}\rho'$ for a pair of roots $z_g, z_h \in D(c', \rho')$, combine it with claim (ii) for the output of Alg. 18.1, and obtain that at most $2.5 + \log_2(8\sigma m/5)$ subdivision steps break the square S into more than one component.

Until that moment apply no new compression steps and hence apply less than $2m$ such steps overall – no more than there are nodes in the partition tree $\mathbb{T}$.

Now elaborate upon Observation 17.1 as follows.

Theorem 18.1. *For a fixed $\epsilon \ge 0$ let subdivision and compression steps be applied to a component C of suspect squares containing at most m roots. Then at most m final compression steps among at most $2m-1$ compression steps overall output isolated discs of radii at most ϵ that cover all roots lying in C.*

Corollary 18.1. *Under the assumptions of Thm. 18.1 all subdivision steps process at most $6m-3$ compact components, made up of at most $24m-12$ suspect squares; this involves at most $24m-12$ e/i tests.*

Proof. A compact component stays unbroken and uncompressed in at most 3 consecutive subdivision steps, which end with a compression step or follow a non-final compression step, that is, such components are processed in at most $6m-3$ subdivision , which involve at most $24m-12$ e/i tests overall. □

19 Counting e/i tests

19.1 An upper estimate

The *upper estimate* $2m-1$ of Thm. 18.1 is *sharp* – reached in the case of the complete binary tree $\mathbb{T}$ and implying that our accelerated subdivision process can involve $2m-1$ suspect squares (e/i steps). This reaches lower estimate up to a constant factor:

Theorem 19.1. *Let subdivision iterations with compression be applied to a suspect square containing at most m roots. Then they involve $O(m)$ e/i tests overall.*

19.2 The first lemma

Our proof of Thm. 19.1 begins with

Lemma 19.1. *In every component C made up of $\sigma(C)$ suspect squares we can choose $n(C) \geq \sigma(C)/5$ non-adjacent suspect squares.*

Proof. Among all leftmost squares in C the uppermost one has at most four neighbors – none on the left or directly above. Include this square into the list of non-adjacency squares and remove it and all its neighbors from C. Then apply this recipe recursively until you remove all $\sigma(C)$ suspect squares from C. Notice that at least 20% of them were non-adjacent. □

19.3 The second lemma

Lemma 19.2. *Fix two integers $i \geq 0$ and $g \geq 3$ a value σ of softness, for $1 < \sigma < \sqrt{2}$, define the separation fraction $a := 2 - \sigma\sqrt{2}$ of Observation 17.2. For all j let j-th subdivision step process v_j components made up of σ_j suspect squares, n_j of which are non-adjacent, let Δ_j denote their side length, and let the j-th steps perform no compression for $j = i, i+1, \dots, i+g$. Then*

$$\sigma_{i+g} \geq (n_i - v_i)a2^{g-0.5} \text{ if } \Delta_i \geq 2^g \Delta_{i+g}. \tag{19.1}$$

Proof. At the ith step a pair of non-adjacent suspect squares of the same component is separated by at least the gap $a\Delta_i$, and there are at least $n_i - v_i$ such gaps with overall length $(n_i - v_i)a\Delta_i$. Neither of these gaps disappears at the $(i+g)$th step, and so they must be covered by σ_{i+g} suspect squares, each of diameter $\sqrt{2}\ \Delta_{i+g} = \Delta_i/2^{g-0.5}$ under (19.1). Hence $(n_i - v_i)a\Delta_i \leq \sigma_{i+g}\Delta_i/2^{g-0.5}$. □

By virtue of Cor. 18.1 all compact components together involve $O(m)$ suspect squares and e/i tests; it remains to prove that we only involve $O(m)$ other e/i tests, for which $n_j \geq 2$ for all j.

Next we extend Lemma 19.2 by allowing compression steps. We only need to extend the equation $2\Delta_{j+1} = \Delta_j$ to the case where the jth subdivision step for $i \leq j \leq i+g$ may perform compression, after which we apply subdivision process to a single suspect square with side length at most $4\Delta_j$.

The next two subdivision steps perform only five e/i tests, that is, at most $10m-5$ e/i tests are performed at all pairs of subdivision steps following all (at most $2m-1$) compression steps.

Then again, we can exclude all these e/i tests and bound by $O(m)$ the number of the e/i tests in the remaining subdivision process, which is not interrupted with compression steps anymore.

Furthermore, we resume subdivision process after a compression step with a suspect square whose side length decreased (in two discounted subdivision steps) from at most $4\Delta_j$ to at most Δ_j.

Hence we can apply to it Lemma 19.2, because now it accesses the same number n_j of non-adjacent suspect squares and an increased number σ_j of all suspect squares (cf. (19.1)).

19.4 Counting e/i tests at partition steps

Let h be the height of our partition tree $\mathbb{T}$ and count suspect squares processed in subdivision steps that partition components, temporarily ignoring the other steps.

Clearly, bounds (19.1) still hold and can only be strengthened. Notice that $v_i \geq 1$ for all i, sum v_{i+g} as well as bounds (19.1) for a fixed $g > 0$ and $i = 0, 1, \dots, h-g$, and obtain

$$\sum_{i=0}^{h-g} n_i - \lambda \sum_{i=0}^{h-g} \sigma_{i+g} \leq \sum_{i=0}^{h-g} v_{i+g} \text{ for } \lambda := \frac{\sqrt{2}}{a2^g}. \tag{19.2}$$

Notice that

$$\sum_{i=0}^{h-g} v_{i+g} \leq \sum_{i=0}^{h} v_i - \sum_{i=0}^{g-1} v_i < 2m - g,$$

Combine the above estimate and the bounds $n_i \leq m$ and (cf. Lemma 19.1) $\sigma_i \leq 5n_i$, for all i, write $n' := \sum_{i=0}^{h} n_i$, obtain

$$\sum_{i=0}^{h-g} n_i \leq n' - \sum_{i=h-g+1}^{h} n_i \leq n' - mg, \ \sum_{i=0}^{h-g} \sigma_{i+g} \leq \sum_{i=0}^{h} \sigma_i \leq 5n',$$

and so

$$(1-5\lambda)n' \leq 2m - 1 - g + mg \text{ for } \lambda \text{ of (19.2)}.$$

Obtain that

$$0.5n' \leq 2m - 1 - g + mg \text{ for } g = 1 + \lceil \log_2(\frac{5\sqrt{2}}{a}) \rceil.$$

Hence

$$n' \leq 4m - 2 - 2g + 2mg, \text{ and so } n' = O(m) \text{ for } g = O(1). \tag{19.3}$$

19.5 Extension to counting all e/i steps

Next consider a non-compact component, count all suspect squares processed in all subdivision steps applied until it is partitioned, and then prove in Lemma 19.3 that up to a constant factor the overall number of these suspect squares is given by their number at the partition step. Such a result will extend to all subdivision steps our count $n' = O(m)$ of (19.3), so far restricted to partition steps. Together with Lemma 19.1 this will complete our proof of Thm. 19.1.

Lemma 19.3. *Let $k+1$ successive subdivision steps with σ-soft e/i tests be applied to a non-compact component for σ of our choice in the range $1 < \sigma < \sqrt{2}$. Let the j-th step output n_j non-adjacent suspect squares for $j = 0, 1, \dots, k$. Then $\sum_{j=0}^{k-1} n_j \leq n_k \frac{10\sqrt{2}}{a}$ for the separation fraction $a := 2 - \sigma\sqrt{2}$ (cf. Observation 17.2.)*

Proof. Substitute $g = k - i$ into (19.3) and obtain $\sigma_k \geq (n_i - v_i)a2^{k-i-0.5}$, for $i = 0, 1, \dots, k-1$.

Recall that $\sigma_j \leq 5n_j$ (cf. Lemma 19.1) and that in our case $v_i = 1$ and $n_i \geq 2$ and obtain that $n_i - v_i \geq n_i/2$ for all i.

Hence $5n_k \geq n_i a 2^{k-i-1.5}$ for all i, and so

$$\sum_{i=0}^{k-1} n_i \leq n_k \frac{10\sqrt{2}}{a} \sum_{j=1}^{k} 2^{-j} < n_k \frac{10\sqrt{2}}{a}.$$

$\square$

Remark 19.1. *The overhead constants hidden in the "O" notation of our estimates are fairly large but overly pessimistic. E.g., (i) the roots x and y involved in the proof of Lemma 19.2 are linked by a path of suspect squares, but we deduce the estimates of the lemma by using only a part of this path, namely, the part that covers the gaps between non-adjacent suspect squares. (ii) Our proof of that lemma covers the worst case where a long chains between the roots x and y can consist of up to $4m$ suspect squares, but for typical inputs such chains are much shorter and can consist of just a few suspect squares. (iii) In the proof of Lemma 19.3 we assume that $n_i - 1 \geq n_i/2$; this holds for any $n_i \geq 2$ but the bounds become stronger as n_i increases. (iv) In a component made up of five suspect squares at least two are non-adjacent, and so the bound of Lemma 19.1 can be strengthened at least in this case.*

Remark 19.2. *Alg. 18.1 compresses an input disc $D(0,1)$ into η-rigid disc for $\eta = 5/8$, but our proof of Thm. 19.1 only involves η the argument in Sec. 18.4, and this argument and therefore the whole proof of the theorem can be immediately extended if it relies on compression into any rigid disc.*

20 A variant of compression algorithm

Recall Schröder's iterations [118], celebrated for fast convergence to a root of multiplicity m [70, Sec. 5.4]:

$$z = y - m\ \mathrm{NR}(y). \tag{20.1}$$

Due to map (2.2) we only need to consider the case where the input disc D is the unit disc $D(0,1)$ and first assume that it contains all d zeros of p and readily verify

Observation 20.1. *Schröder's single iteration (20.1) for $m = d$ and $p(x) = p_d(x - z_1)^d$ outputs the zero z_1 of $p(x)$ unless $y = z_1$.*

Next we extend this observation by proving that the output z of Schröder's iteration (20.1) lies near the convex hull $CH(X(D))$ provided that $m = d$, a disc D contain all d zeros of p, and point y lies for from the disc D. Here are our specific estimates.

Theorem 20.1. *Write $D := D(0,1)$, $CH := CH(D)$, and $\Delta := \Delta(D)$, fix y such that $|y| > 2$, let $z := y - d\mathrm{NR}(\mathrm{y})$, which is (20.1) for $m = d$. Let $\bar{x} \in CH$. Then $|z - \bar{x}| \leq \alpha\ \Delta$ for $\alpha := \frac{2y|-1)|y|}{(y|-2)^3}|y - \bar{x}|$.*

Proof. Let $\bar{x} \in CH$ and then $|z_j - \bar{x}| \leq \Delta$ for $j = 1, \dots, d$. Hence (cf. (2.1))

$$\left|\mathrm{NIR}(y) - \frac{d}{y - \bar{x}}\right| \leq \sum_{j=1}^{d} \left|\frac{1}{y - z_j} - \frac{1}{y - \bar{x}}\right| \leq \sum_{j=1}^{d} \frac{|\bar{x} - z_j|}{|y - \bar{x}| \cdot |y - z_j|} \leq \frac{d\Delta}{(|y| - \Delta)^2},$$

and so

$$|u-w|\le\beta \text{ for } u:=\frac{\mathrm{NIR}(y)}{d},\ w:=\frac{1}{y-\bar{x}},\ \text{and } \beta:=\frac{\Delta}{(|y|-\Delta)^2}. \tag{20.2}$$

Hence

$$|z-\bar{x}|=\Big|\bar{x}-y+\frac{d}{\mathrm{NIR}(y)}\Big|=\Big|\frac{1}{w}-\frac{1}{u}\Big|=\frac{|u-w|}{|uw|}\le\frac{\beta d}{|\mathrm{NIR}(y)|}|y-\bar{x}|.$$

Now write

$$\bar{z}_j:=\frac{z_j}{y} \text{ for } j=1,\dots,d$$

and deduce from (2.1) that

$$|y|\ |\mathrm{NIR}(y)|=\Big|\sum_{j=1}^{d}\frac{1}{1-\bar{z}_j}\Big|=\Big|\sum_{h=0}^{\infty}\sum_{j=1}^{d}\bar{z}_j^h\Big|\ge d-\Big|\sum_{h=1}^{\infty}\sum_{j=1}^{d}\bar{z}_j^h\Big|=d-\sum_{j=1}^{d}\Big|\frac{\bar{z}_j}{1-\bar{z}_j}\Big|.$$

Recall that $\max_{j=1}^{d}|z_j|\le 1<|y|$, and so $|\frac{\bar{z}_j}{1-\bar{z}_j}|\le\frac{1}{|y|-1}$. Hence

$$|y|\ |\mathrm{NIR}(y)|\ge d-d\frac{1}{|y|-1}=d\frac{|y|-2}{|y|-1}\ge 0,$$

$$\frac{d}{|\mathrm{NIR}(y)|}\le\frac{(|y|-1)|y|}{|y|-2}.$$

Combine this bound with the above bound on $|z-\bar{x}|$ and obtain

$$|z-\bar{x}|\le\frac{(|y|-1)|y|}{|y|-2}\beta\ |y-\bar{x}|$$

for β of (20.2). Substitute $\beta:=\frac{\Delta}{(|y|-\Delta)^2}$ and obtain

$$|\bar{x}-z|\le\frac{(|y|-1)|y|}{(|y|-\Delta)^2(|y|-2)}\Delta|y-\bar{x}|.$$

Substitute $\Delta\le 2$ and obtain the theorem. □

Notice that $|z-\bar{x}|\ge r_d(z)$, while $\alpha\to 1$ as $|y|\to\infty$ and in particular $|z-\bar{x}|\le 1.1\Delta$ if, say, $|y|\ge 100$.

Now notice that

Observation 20.2. *Let a point c lie in a disc D and let $\Delta:=\Delta(CH(X(D)))$ denote the diameter of the convex hull of the root set $X((D))$. Write Then $r-\rho\le\Delta\le 2r$ for $r:=r_{d-m+1}(c',p)$ and $\rho:=r_d(c',p)$.*

Corollary 20.1. *If $\rho\le\nu r$ under the notations of Observation 20.2, then the disc D is $(1-\nu)/2$-rigid.*

It follows that the disc $\bar{D}:=D(z,r_1(z))$ is rigid, and so we can compress into it the 100-isolated unit disc $D(0,1)$ at HLA cost 2.

Now deduce from Eqn. (2.1) that similar argument can be applied if $m<d$, Schröder's iterations (20.1) is applied at a point y such that $|y|^2=\theta$ form a sufficiently large θ, the unit disc $D(0,1)$ contains precisely m zeros of pa and is θ-isolated, that is, $i(D(0,1))\ge\theta$ and $\#(D(0,1))=m$. Furthermore, verify that this can be achieved already for θ of order $d^{O(1)}$, and we can ensure this by

delaying compression and performing instead $o(\log(d))$ subdivision steps until either *theta* increases to that level or the input compact component is broken into more than one component.

Then the proof of Thm. 19.1 still holds if its bound increases by a factor of $\log(d)$.

Part IV: Deflation

21 Deflation: when and how?

21.1 Comments and contents

In this section we cover deflation of a factor of a polynomial. In Sec. 21.2 we compare the benefits and shortcomings of explicit and implicit deflation. We conclude that explicit deflation of of p is numerically safe if the degree of the output is low, but otherwise deflation tends to destroy sparsity and to blow up the coefficient size. We cover deflation of factors of any degree, but realistically one should avoid deflation in subdivision process until the degree of the factor decreases to a sufficiently low level. In Sec. 23.2 we extend deflation to complete factorization of p.

We can readily verify correctness of both explicit and implicit deflation by means of root-finding for f (see Sec. 21.6).

We study deflation in two cases.

Case (i). Assume that some roots are tame, that is, have been approximated, and deflate a wild factor f of p having root set made up of the remaining w wild roots, with application in Sec. 5. Delve and Lyness in [30] and more recently Schleicher and Stoll in [128] apply *Newton's identities* (see Alg. 21.3) to recover f from the power sums of wild roots, computed as the differences between the power sums of all roots and tame roots. The algorithm has been presented for a polynomial p given in monomial basis and has complexity of $O(wd)$ arithmetic operations (see Sec. 21.3).

The precision of these computations can be readily estimated a posteriori but is not known a priori and is unbounded for the worst case input, although this caused no problems in the tests in [128].

Our alternative Alg. 21.2 of Sec. 21.4, based on evaluation and interpolation, avoids this problem, and we extend it to deflation of a wild factor of a black box polynomial p within a fixed error bound $\epsilon > 0$. Its *complexity* is dominated by the cost of the evaluation of a polynomial p at $O(w\log(1/\epsilon))$ points.

Case (ii). In Sec. 21.5, within the latter cost bound under error bound ϵ, we can explicitly deflate a factor f of a black box polynomial p over an isolated disc, such that f and p share their root set in it (see the next subsection). In this case we can readily approximate the power sums by Cauchy sums and then recover the factor f by using Newton's identities or Newton's iterations (see Algs. 21.3 and 21.4, respectively). We observe strong numerical stability of the former algorithm.

Both algorithms can approximate the factor f quite closely at a low arithmetic cost; in Appendix B we outline Schönhage's algorithms of [120, Secs. 10 –12] that fast refine such approximations at a low cost.

In Sec. 21.7 we cover a simple deflation by means of Taylor's shift (we call it Laser deflation), which worked empirically in the paper [62] by Kobel, Rouillier, and Sagraloff for a narrow input class, and we show that it is numerically unstable for a worst case input.

21.2 When should we deflate?

Consider pros and cons for choosing or skipping deflation.

With *implicit deflation* of a factor $f = f(x)$ of a degree $w < d$, we apply the same or another root-finder to the input polynomial $p(x)$ but only approximate the w roots of the factor $f(x)$. In Remark 21.1 we elaborate upon such deflation in case (i) of deflation of a wild factor.

With *explicit deflation* we explicitly compute a factor f and then approximate its roots faster because it has a lower degree w (see below). Furthermore, the computational cost tend to decrease because the root set shrinks. In the case of explicit deflation we can first compute a crude approximation of the factor $f(x)$ and then refine it fast, by applying Schönhage's advanced algorithm s of [120, Secs. 10 – 12] (see our Appendix B).

We should, however, weigh the benefits of explicit deflation against its computational cost and 'side effects", that is, the coefficient growth and the loss of sparsity. For example, consider explicit deflation of factors of degree k of $p(x) = x^{3k+1} + 1$ for a large k. There is no such "side effects" for the factors of the form $(x - c)^m$, but they can appear in factorization of $(x - c)^m - \delta$. The 'side effects" steadily decrease as the degree w of the factor decreases:[37] $f(x)$ has at most $w+1$ non-zero coefficients, and if $f(x)$ is monic and has all its roots lying in the unit disc $D(0, 1)$, then the norm $|f(x)|$ reaches its maximum 2^w for $f = (x - 1)^w$.

Hence if w is a small positive integer, then the "side effects" are minor or disappear, and we can safely deflate f explicitly. In particular deflation tends to be efficient in the highly important case of real root-finding (see Sec. 21.6) because real roots tend to make up a small fraction of all d complex roots.

In some cases we can avoid or mitigate the "side effects" by means of avoiding or delaying deflation.

Example 21.1. *Let $p = \prod_{j=1}^{d}(x - 1 + 2^{-j})$ for a large integer d. In this case isolation of the root $1 - 2^{-j}$ weakens as j increases. Subdivision would recursively approximate the roots $1 - 2^{-j}$ for $j = 1, 2, \dots,$ but should we deflate the factors $x - 1 + 2^{-j}$ and continue computations with the factors $g_k(x) = \prod_{j=k}^{d}(x - 1 + 2^{-j})$ of degree $d - k$ for $k = 2, 3, \dots$? Approximation of such a factor of p for $2k \ge d$ involves at least $\frac{b}{2}d^2$ bits and at least $\frac{b}{8}d^2$ Boolean operations. Indeed, the root set of p_i consists of at least $0.5d$ roots, and we must operate with them with a precision of at least $\frac{b}{4}d$ bits (see Cor. 1.1). Hence approximation of these factors for $i = 1, \dots, d-1$ involves at least $\frac{b}{8}d^3$ bits and at least $\frac{b}{16}d^3$ Boolean operations. We can decrease this cubic lower bound to quadratic level if we skip explicit deflation at the kth step unless $\frac{\deg(g_k)}{\deg(g_{k+1})} \ge \nu$ for a fixed $\nu b > 1$, e.g., $\nu = 2$. Under such a degree bound we apply deflation not more than $\lceil \log_\nu(d) \rceil$ times.*

Remark 21.1. *We can yield the benefits of contracting root set and avoid the "side effects" of explicit deflation in* implicit deflation*, that, is application of a selected root-finder to the wild factor $f(x)$ available only implicitly. Assume that close approximations $z_1, \dots, z_t$ to tame roots $x_1, \dots, x_t$ are available such that we can identify x_j with z_j and $t(x)$ with $\prod_{j=1}^{d-w}(x - z_j)$. Then equations*

$$f(x) = p(x)/t(x), \tag{21.1}$$

$$\mathrm{NR}_f(x) = \frac{f'(x)}{f(x)} = \mathrm{NR}_p(x) - \sum_{j=1}^{d-w} \frac{1}{x - x_j} \tag{21.2}$$

enable us to evaluate $f(x)$ and $\mathrm{NR}_f(x) = \frac{f'(x)}{f(x)}$ without explicit deflation. Moreover, we can accelerate simultaneous numerical approximation of the sum $\sum_{j=1}^{d-w} \frac{1}{x - x_j}$ at many points x by applying FMM.

[37] Boolean complexity of explicit deflation of high degree factors can be bounded, and such deflation was the basis for theoretical study of the complexity of high accuracy root-finding in [90] (cf. Part I of our Appendix), but the implied growth of the coefficients and working precision has made these algorithms quite hard to implement.

21.3 Cauchy-Delves-Lyness-Schönhage deflation of a wild factor

Algorithm 21.1. A wild factor via the power sums of the roots *[30]*.[38]

Given $w+1$ leading coefficients of a polynomial p of (1.1) and its tame roots $x_{w+1},\dots,x_d$, first successively compute the power sums

(i) $s_h = \sum_{j=1}^{d} x_j^h$ of p of all d roots of p,

(ii) all $s_{h,\mathrm{tame}} = \sum_{j=w+1}^{d} x_j^h$ of its $d-w$ tame roots, and

(iii) $s_{\mathrm{wild}} = s_h - s_{h,\mathrm{tame}}$ of its wild roots.

In all cases (i)-(iii) do this for $h = 1,\dots,w$. Then

(iv) compute the wild factor $f(x) = \prod_{j=1}^{w}(x-x_j)$ by applying Alg. 21.3, that is, by using Newton's identities (3.15).

Stages (i), (ii), (iii), and (iv) involve $O(w\log(w))$, $O(wd)$, w, and $O(w^2)$ arithmetic operations, respectively. The *overall arithmetic complexity* is dominated by order of wd operations at stage (ii).

Remark 21.2. *We would bound the output errors by $\epsilon > 0$ if we perform the computations with the precision of order* $\log(1/\epsilon)$), *except that in subtractions at stage (iii) we would lose about*

$$b_{\mathrm{loss}} = \log_2\left(\left|\frac{\sum_{j=1}^{d} x_j^h}{\sum_{j=1}^{w} x_j^h}\right|\right) \tag{21.3}$$

bits of precision. For the worst case input all wild roots have much smaller magnitude than some tame roots, and so generally the value b_{loss} is unbounded a priori, and so are the output errors of Alg. 21.1, under any fixed bound on working precision*, although a posteriori precision bounds are readily available. Furthermore, we can verify correctness of deflation at a low cost in the case where w is small, by means of root-finding for $f(x)$ (cf. Alg. 21.5), which runs at a low cost where* $\deg(f)$ *is small. For many inputs the value b_{loss} is reasonably well bounded; in particular no numerical problems have been reported in numerical tests in [126], based on Newton's identities at both stages (i) and (iv).*

Remark 21.3. *Stage (i) involves the coefficients of p, but we can modify it by approximating the power sums of the roots of p within $\epsilon > 0$ by Cauchy sums at the cost of the evaluation of the ratio p'/p at $O(w\log(1/\epsilon))$ points. This would dominate the* overall arithmetic complexity *of the resulting algorithm, said to be* **Alg. 21.1a** *and handling a black box input polynomial $p(x)$.*

Remark 21.4. *At stage (iv) Alg. 21.3 runs at a dominated arithmetic cost and has very strong numerical stability. We can decrease this cost bound significantly if instead we apply Alg. 21.4, less stable numerically.*

21.4 Deflation of a wild factor by means of evaluation and interpolation

Algorithm 21.2. Deflation of a wild factor by means of evaluation and interpolation.

Fix K distinct complex points $\zeta_0,\zeta_1,\dots,\zeta_{K-1}$ for $w < K \le 2w$, not being tame roots or lying close to them, write $t(x) := \prod_{g=w+1}^{d}(x-x_g)$ and $f(x) := \frac{p(x)}{t(x)}$, and compute

(i) $p(\zeta_j)$ for $j = 0,\dots,K-1$,

[38][30] approximates the power sums of the roots of $p(x)$ at stage (i) below by adjusting some customary algorithms of that time for numerical integration over a circle, but we prefer to apply Newton's identities (3.15) at that stage.

(ii) $f(\zeta_j) = \frac{p(\zeta_j)}{t(\zeta_j)}$ $j = 0, \ldots, K-1$, *and*

(iii) the coefficients of the polynomial $f(x)$.

The computation is reduced to evaluation of $p(x)$ at $K \leq 2w$ points at stage (i), which dominates *overall complexity*, $(d-w)K$ subtractions and as many divisions at stage (ii), and interpolation to the degree w polynomial $f(x)$ at stage (iii).

We can scale the variable x by a fixed or random $a \in C(0,1)$, avoiding coefficient growth and division by 0 at stage (ii) with probability 1, then write

$$\zeta_j := a\zeta^j, \text{ for } j = 0, 1, \ldots, K-1, \tag{21.4}$$

for $\zeta := \exp\left(\frac{2\pi\mathbf{i}}{K}\right)$ of (1.3), and perform stage (iii) by applying inverse DFT at the Kth roots of unity to $f(ax)$, which amounts to multiplication by unitary matrix (up to scaling by a constant) and is numerically stable. In this case the computations involve $O(K\log(K))$ arithmetic operations at stage (iii).

For a polynomial $p(x)$ given in monomial basis, stage (i) is reduced to numerically stable DFT applied to the polynomial $p(ax)$ and involves $O(K\log(K))$ arithmetic operations; in this case the *overall arithmetic complexity* is dominated by $2(d-w)K = O(wd)$ operations at stage (ii).

Alg. 21.2 has no problem of numerical stability, unlike Algs. 21.1 and 21.1a.

21.5 Deflation of a factor over an isolated disc

Given an isolated disc on the complex plane, we can deflate a factor f of p where f and p share the root set in that disc as follows.

First approximate the power sums s_h of all w roots of f by Cauchy sums $s_{h,q}$ for $h = 1, 2, \ldots, w = \deg(f)$. Ensure an error bound $\epsilon > 0$ by computing the Cauchy sums within *complexity* bound at the cost dominated by the cost of the evaluation of the ratio p'/p at $O(w\log(1/\epsilon))$ points.

Then approximate the coefficients of f within the error bound $O(\epsilon)$ by means of the following algorithm, which has strong numerical stability and can be also incorporated into Alg. 21.1, at its stage (iv).

Algorithm 21.3. The coefficients from the power sums via Newton's identities.

Given the power sums $s_1, \ldots, s_k$ *of the roots of a monic polynomial* $f(x) = x^w + \sum_{i=0}^{w-1} f_i x^i$, *for* $k \leq w = \deg(f)$, *recursively compute its* k *leading coefficients as the* k *trailing coefficients of the reverse polynomial*

$$f_{\text{rev}}(x) = x^w f\left(\frac{1}{x}\right) = 1 + \sum_{i=1}^{w} f_{w-i}x^i$$

by solving the linear system of k *Newton's identities (3.15), that is,*

$f_{w-1} + s_{1,f} = 0,$

$s_{1,f}f_{w-1} + 2f_{w-2} + s_{2,f} = 0,$

$s_{2,f}f_{w-1} + s_{1,f}f_{w-2} + 3f_{w-3} + s_{3,f} = 0,$

. . . .

We can solve such a triangular linear system of k equations in k^2 arithmetic operations by means of substitution. Furthermore, we can ensure numerical stability of these computations if we make linear system diagonally dominant by moving all roots into the disc $D(0, \frac{1}{2k})$ by means of scaling the variable x.

Numerical stability of transition from power sums to coefficients with Newton's identities. The following theorem links the absolute values of the coefficients, the roots, and their power sums.

Theorem 21.1. *Let*

$$f_{\text{rev}} := x^w + \sum_{i=1}^{w} f_i x^{w-i} = \prod_{i=1}^{w}(x - x_j)$$

denote a polynomial whose all w roots lie in the unit disc $D(0,1)$ and let $s_0, s_1, \dots, s_w$ denote the power sums of these roots.

(i) If $|s_h| \le \nu^h$ for $h = 1, \dots, w$ and $\nu \ge 0$, then $|f_h| \le \nu^h$ for $h = 1, \dots, w$, while $|f_1| \le \alpha$ and $|f_h| < \alpha$ for $h = 2, 3, \dots, w$ if $|s_h| \le \alpha < 1$ for $h = 1, \dots, w$.

(ii) If $|f_h| \le \nu^h$ for $h = 1, \dots, w$, then $|s_h| \le (2\nu)^h$ for $h = 1, \dots, w$.

(iii) The bounds $\nu|x_j| < 1$ hold for $j = 1, \dots, w$ if and only if

$$|f_h| \le \binom{w}{h} \nu^h \text{ for } h = 1, \dots, w.^{39} \tag{21.5}$$

Proof. Define w Newton's identities (3.15) for $d = w$ and w_i replacing p_i. They form two triangular linear systems expressing $f_1, \dots, f_w$ via $s_1, \dots, s_w$ and vice versa. Solve them by applying back substitution and deduce claims (i) and (ii). Claim (iii) follows because the w-multiple root $-1/(2\nu)$ of the polynomial $(1 + 2\nu x)^w$ is absolutely smaller than every root of $f(x)$ under (21.5). ☐

This theorem supports numerically stable implementation of stage (iii) of Alg. 21.2.

We lose this benefit but decrease the arithmetic cost if at stage (iii) of Alg. 21.2 we replace Alg. 21.3 with the following algorithm of [120, Sec. 13]; it involves $2k$ first power sums in order to output k leading coefficients of $f(x)$ for $k \le w$ and decreases the dominated arithmetic cost $O(n^2)$ of stage 3 to $O(n \log(n))$.

Algorithm 21.4. Recovery of a polynomial from the power sum of its roots by means of Newton's iterations.

Write $f_{\text{rev}}(x) := 1 + g(x) = \prod_{j=1}^{w}(1 - xx_j)$ and deduce that

$$(\ln(1 + g(x)))' = \frac{g'(x)}{1 + g(x)} = -\sum_{j=1}^{d_f} \frac{x_j}{1 - xx_j} = -\sum_{j=1}^{h} s_j x^{j-1} \mod x^h \tag{21.6}$$

for $h = 1, 2, \dots, w + 1$.

Write $g_r(x) := g(x) \mod x^{r+1}$, observe that $g_1(x) = -s_1 x$ and $g_2(x) = -s_1 x + (s_1^2 - s_2)x^2$, and express the polynomial $g_{2r}(x)$ as follows:

$$1 + g_{2r}(x) = (1 + g_r(x))(1 + h_r(x)) \mod x^{2r+1} \tag{21.7}$$

for an unknown polynomial

$$h_r(x) = h_{r+1} x^{r+1} + \dots + h_{2r} x^{2r}. \tag{21.8}$$

[39] Recall that $\sum_{h=0}^{w} \binom{n}{h} \nu^h = 2^w$.

Equation (21.7) implies that

$$\frac{h'_r(x)}{1+h'_r(x)} = h'_r(x) \mod x^{2r+1}.$$

Hence

$$\ln(1+g_{2r}(x))' = \frac{g'_{2r}(x)}{1+g_{2r}(x)} = \frac{g'_r(x)}{1+g_r(x)} + \frac{h'_r(x)}{1+h_r(x)} \mod x^{2r}.$$

Combine these identities with equation (21.6) for $w = 2r+1$ and obtain

$$\frac{g'_r(x)}{1+g_r(x)} + h'_r(x) = -\sum_{j=1}^{2r} s_j x^{j-1} \mod x^{2r}. \tag{21.9}$$

Having the power sums $s_1, \ldots, s_{2d_f}$ and the coefficients of the polynomials $g_1(x)$ and $g_2(x)$ available, recursively compute the coefficients of the polynomials $g_4(x), g_8(x), g_{16}(x), \ldots$ by using identities (21.7) – (21.9).

Namely, having the polynomial $g_r(x)$ available, compute the polynomial $\frac{1}{1+g_r(x)} \mod x^{2r}$ and its product with $g'_r(x)$. Then obtain the polynomials $h'_r(x)$ from (21.9), $h_r(x)$ from (21.8), and $g_{2r}(x)$ from (21.7).

Notice that $\frac{1}{1+g_r(x)} \mod x^r = \frac{1}{1+g(x)} \mod x^r$ for all r and reuse the above polynomials for computing the polynomial $\frac{1}{1+g_r(x)} \mod x^{2r}$. Its coefficients approximate the coefficients $f_1, \ldots, f_k$ of $f(x)$ and its reverse polynomial, and we can stop at any positive k not exceeding $w-1$.

The algorithm performs $\lceil \log_2(k) \rceil$ iterations; the ith iteration amounts to performing single polynomial division modulo x^{2^i} and single polynomial multiplication modulo x^{2^i}, $i = 1, 2, \ldots, \lceil \log_2(k \rceil + 1$, at the overall arithmetic cost in $O(k \log(k))$, and so performing $O(w \log(w))$ arithmetic operations is sufficient in order to approximate all coefficients of the reverse polynomial $f_{\rm rev}(x)$, which it shares with $f(x)$. See the estimates for working precision and Boolean complexity of this algorithm in [120, Lemma 13.1].

21.6 Correctness verification of deflation by means of root-finding

Algorithm 21.5. Correctness verification by means of root-finding.

> *Closely approximate all d_f roots of the polynomial f and estimate the smallest root radius centered at each of these approximation points. See Sec. 15 and some relevant error and precision estimates in Part I of the Appendix.*

21.7 Laser Deflation

Suppose that a small and $\frac{1}{\rho}$-isolated disc $D(c, \rho)$ for a small positive ρ covers a cluster of m roots of p and write

$$\tilde{f} := p(x) \mod (x-c)^{w+1}. \tag{21.10}$$

Clearly $\tilde{f}$ converges to a factor of p with its root set converging to this cluster as $\rho \to 0$, and Kobel et al [62, Sec. 3.2] propose to compute the polynomial $\tilde{f}$ by means of shifting the variable $x \to x - c$. We call the computation of $\tilde{f}$ *Laser deflation* because, unlike all other deflation algorithms of this section, it is only defined under the assumption that a very close approximation c to the cluster is given. The algorithm involves the coefficients of $p(x)$, but we extend it below to a black box polynomial; it is not clear, however, if it has any advantages versus the algorithms of Sec. 21.5, although in [62, Sec. 3.2] Kobel et al report empirical success of Laser deflation in the case where a fixed small segment of the real axis contains a small number of roots and is very strongly isolated from the external roots.

(i) Laser deflation for a black box polynomial.

Reduction to convolution enables us to deduce the following result (cf. [89, Sec. 2.5]), whose supporting algorithm is numerically unstable for $d > 50$.

Theorem 21.2. *Given the coefficients of a polynomial $p(x)$ of (1.1) and a number c one can compute the coefficients of the polynomial $s(x) = p(x-c)$ by applying $O(d\log(d))$ arithmetic operations.*

We can avoid numerical problems and approximate the trailing coefficients of a black box polynomial p by divided differences.

> **Algorithm 21.6.** *Fix a small positive ϵ, compute $\tilde{p}_0 := p_0 = p(0)$, recursively compute approximations $\tilde{p}_k \approx p_k = \frac{1}{\epsilon^k}(p(\epsilon) - \sum_{i=0}^{k-1}\tilde{p}_i\epsilon^i)$ for $k = 1, 2, \dots, w$, and observe that $\tilde{p}_k \to p_k$ as $\epsilon \to 0$.*

Correctness of the algorithm follows from the link of the coefficients $p_1, \dots, p_d$ of p to the values of the derivative and the higher order derivatives at the origin:

$$p_k = k!p^{(k)}(0),\ \ k = 0, 1, \dots, d. \tag{21.11}$$

The algorithm evaluates $p(x)$ at $w+1$ points and also involves $4w-3$ additional arithmetic operations, $2w$ of them for the computation of the values $\epsilon^k p_k$ for $k = 1, \dots, w$ by means of Horner's algorithm.

All this can be readily extended to the computation of the w trailing coefficients of the polynomial $s(x) = p(x-c)$, and we can determine the integer w (if it is unknown) as the minimal order of the derivative of p that neither vanishes nor nearly vanishes at c implying similar property for the approximations by divided differences.

(ii) Some hard inputs.

Next we prove that the roots of $\tilde{f}$ can greatly deviate from the cluster of the roots unless the disc $D(c, 1)$ is isolated exponential in $d-w$.

Assume that $c = 0$. If furthermore, $f = x^w$, so that $x_1 = \dots = x_w = 0$ is a w-multiple root, then the approximation of f by $\tilde{f}$ has no error, but let

$$p := fg \text{ for } f = (x+\rho)^w,\ \ g = 1 + x + x^2 + \dots + x^{d-w} \text{ for } \rho \le 1. \tag{21.12}$$

In this case polynomial p has a w-multiple root ρ and $d-w$ simple roots ζ_{d-w+1}^j, $j = 1, 2, \dots, d-w$, being the $(d-w+1)$st roots of unity except for the point 1. Deduce from (21.10) and (21.12) that

$$p_w = 1,\ \ |p| = |f|\ |g| = (1+\rho)^w(d-w+1), \tag{21.13}$$

$$\tilde{f} - f = \sum_{k=1}^{w}\sum_{i=0}^{w-k}\binom{w}{i}\rho^{w-i}x^{i+k}.$$

Therefore

$$|\tilde{f} - f| \ge \sum_{i=0}^{w-1}\binom{w}{i}\rho^{w-i} = (1+\rho)^w - 1 \ge w\rho. \tag{21.14}$$

Now suppose that we seek approximation of the roots x_j for $j > w$ within an error bound 2^{-b} for $b \geq 1$, say. Then we must approximate g and consequently f within error norm about $2^{-(d-w)b}|p|$ (see Observation 1.2).

In view of (21.13) and (21.14) this would require that

$$\rho w \leq (1+\rho)^w - 1 \leq 2^{-(d-w)b}(1+\rho)^w(d-w+1).$$

Recall that $\rho \leq 1 \leq b$, and hence, as we claimed,

$$\rho \leq \frac{(1+\rho)^w(d-w+1)}{w2^{(d-w)b}} \leq \frac{d-w+1}{w}2^{2w-d}.$$

PART V: FUNCTIONAL ITERATIONS AND FACTORIZATION

In Secs. 22 and 23 we outline some new research directions, explain their promise, and provide initial motivation, but leave their detailed analysis and formal and experimental support as a challenge.

In Sec. 22 we recall and strengthen three functional iterations, namely, Newton's, Weierstrass's, or Ehrlich's. Given d reasonably close approximations to the d roots, they refine them with at least quadratic convergence rate, but empirically they also very fast converge to the roots globally – right from the start, under proper customary initialization.

In Sec. 23 we recall Kirrinnis's algorithm of [60]. It refines with quadratic convergence rate a reasonably accurate numerical factorization of a polynomial (see footnote [2] on some important applications of the factorization).

Such a refinement of the factorization can be also achieved by means of root-refining, followed by transition to factorization, but this way generally requires a higher precision of computations (recall Observation 1.2).

At the end of the section we extend our numerical factorization of a polynomial to its multipoint evaluation.

22 Newton's, Weierstrass's, and Ehrlich's iterations

22.1 Contents

In the next subsection we comment on convergence of the three listed iterative root-finders and on their stopping criterion. In Sec. 22.3 we discuss the policy of doubling precision, the resulting dynamic partition of the roots into tame and wild, and the deflation of a wild factor. In Sec. 22.4 we accelerate these iterations by incorporating FMM. In Secs. 22.5 – 22.7 we cover their initialization. In Sec. 22.5 we approximate the d root radii, which defines d narrow annuli whose union contains all d roots, which has also significant independent importance. In Sections 22.6 and 22.7 we further narrow root search to the intersection of a small number of such unions.

22.2 Definition, convergence and stopping

1. Definition. Consider Newton's, Weierstrass's (aka Durand-Kerner's), and Ehrlich's (aka Aberth's) iterations [63, 50, 116, 128, 133, 28, 59, 34, 3, 21]):

$$z \leftarrow z - N(z),\ z_j \leftarrow z_j - W(z_j),\ z_j \leftarrow z_j - E_j(z_j),\ j = 1, \dots, d, \text{ where} \tag{22.1}$$

$$N(x) = W(x) = E_j(x) = 0 \text{ for all } j \text{ if } p(x) = 0 \tag{22.2}$$

and otherwise

$$N(z) := \frac{p(z)}{p'(z)},\ W(x) := \frac{p(x)}{l'(x)},\ \frac{1}{E_j(x)} := \frac{1}{N(x)} - \sum_{i=1, i\neq j}^{d} \frac{1}{x - z_i} \tag{22.3}$$

for $l(x) := \prod_{j=1}^{d}(x - z_j)$.

Newton's iterations can be initiated at a single point z or at many points and can approximate a single root, many or all d roots at the cost decreasing as the number of roots decreases [63, 50, 116, 128].

In contrast, Weierstrass's and Ehrlich's iterations begin at d points for a dth degree polynomial and approximate a subset of its d roots about as fast and almost as slow as they approximate all its d roots.

2. Convergence. Near the roots Newton's and Weierstrass's iterations converge with quadratic rate, Ehrlich's iterations with cubic rate [130], but empirically all of them consistently converge very fast globally, right from the start, assuming a proper initialization, which involves the coefficients (see [14, 21], and our Sec. 22.5).[40] Because of their excellent empirical global convergence, the iterations are currently at least competitive with, if not superior to, subdivision iterations according to user's choice. This, however, is conditioned to proper initialization, which can be ensured at a low cost for polynomials given with coefficients – in monomial basis, but extension to a black box polynomial remains a challenge.

3. Stopping. For a *stopping criterion* one can first verify readily whether the residuals, that is, the differences between the current and previous approximations are within a fixed tolerance, and if so, then one can estimate whether the output errors of the computed approximations z_j to the roots, that is, the smallest root radii $r_d(z_j, p)$, are within a fixed tolerance value.

22.3 Doubling precision and the deflation of a wild factor

Computational (working) precision sufficient in order to control the relative error of an output approximation to a root can be defined by root's condition number (see the relevant discussion in [14, 21]). This number is not readily available, however, and instead one can recursively double precision of computing, until a fixed output accuracy is verified: first apply a fixed root-finding iterations with the IEEE double precision of 53 bits and then recursively double it as long as this is feasible and enables more accurate approximation of the roots, stopping where the required output accuracy is ensured.

This policy for root-finding can be traced back to [14], where it was proposed for MPSolve. It turned out to be highly efficient and was adopted in [38] for Eigensolve, in [21] for the next version of MPSolve, and in [116, 128] for approximation of all roots by means of Newton's iterations.

According to recipe (22.2) the computed approximation z is only updated until $p(z) = 0$ or, in numerical implementation, until the roots become tame, for a fixed stopping criterion. In the latter case, one should restrict updating to the wild roots (cf. Sec. 21.2) and should deflate the wild factor, if it has small degree. When a root is tamed the minimal working precision is exceeded by less than a factor of 2. Recall from [120] and [122, Thm. 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.

[40] Formal support for such global convergence is much weaker for Newton's iterations (see [63, 50, 128], and the references therein) and is completely missing for Weierstrass's and Ehrlich's iterations.

For every working precision the d roots of $p(x)$ are partitioned into $d-w$ tame and w wild roots, as above, and then one can apply the techniques of Secs. 21.2–21.5 for explicit or implicit deflation.

At some point practical limitation on the computational precision may allow no further progress in the approximation of the wild roots of a polynomial of a very high degree. Typically, at this moment the number of the wild roots, that is, the degree of a wild factor, is relatively small for Ehrlich's and Newton's iterations [11, 128],[41] and then explicit deflation of the wild factor can be a natural way out. Otherwise implicit deflation should be applied.

22.4 Acceleration with Fast Multipole Method (FMM)

Given $3d$ complex values w_j, x_i, z_j such that $x_i \neq z_j$ for all $i, j = 1, \dots, d$, **Fast Multipole Method (FMM)** by Greengard and Rokhlin [44] and by Carrier, Greengard, and Rokhlin [26] computes the sums $\sum_{j=1}^{d} \frac{w_j}{x_i - z_j}$ as well as $\sum_{j=1, j\neq i}^{d} \frac{w_j}{z_i - z_j}$ for all i by using $O(db)$ arithmetic operations, with the precision of order b, versus $2d^2$ arithmetic operations in the straightforward algorithm. The paper [107] proposes and extensively tests its acceleration to *Superfast Multipole Method*. The papers [92, 94] list the recent bibliography on FMM and related computations with Hierarchical Semiseparable (HSS) matrices. Relevant software libraries developed at the Max Planck Institute for Mathematics in the Sciences can be found in HLib,

The acceleration based on FMM is practically valuable in spite of the limitation of the working and output precision to the order of $\log(d)$ and the resulting limitation on the decrease of the Boolean computational complexity.

Next we incorporate FMM into Newton's, Weierstrass's, and Ehrlich's iterations. First relate the roots of a polynomial $p(x)$ to the eigenvalues of a diagonal-plus-rank-one matrix

$$A = D + \mathbf{v}\mathbf{e}^T \text{ for } D = \operatorname{diag}(z_j)_{j=1}^{d},\ \mathbf{e} = (1)_{j=1}^{d},\quad \mathbf{w} = (w_j)_{j=1}^{d},\ w_j = W(z_j),$$

$W(x)$ of (22.3) and all j (cf. [15]). Let I_d denote the $d \times d$ identity matrix, apply the Sherman-Morrison-Woodbury formula, and obtain

$$xI_d - A = (xI_d - D)(I_d - (xI_d - D)^{-1}\mathbf{w}\mathbf{e}^T).$$

Hence

$$\frac{1}{p_d} p(x) = \det(xI_d - A) = -\sum_{i=1}^{d} w_i \prod_{j\neq i}(x - z_j) + \prod_{j=1}^{d}(x - z_j). \tag{22.4}$$

Indeed, the monic polynomials of degree d on both sides have the same values for $x = z_j$, $j = 1, \dots, d$ (cf. [15])).[42] Equations (22.4) imply that

$$p(z_i) = -w_i \prod_{j\neq i}(z_i - z_j),\ p'(z_i) = \prod_{j\neq i}(z_i - z_j)\Big(1 - \sum_{j\neq i} \frac{w_i + w_j}{z_i - z_j}\Big),$$

and therefore

$$\frac{w_i}{N(z_i)} = \sum_{j\neq i} \frac{w_i + w_j}{z_i - z_j} - 1. \tag{22.5}$$

Hence, given the $2d$ values of z_j and $w_j = W(z_j)$ for all j, one can fast approximate $N(z_j)$ and consequently $E_j(z_j)$ for all j (cf. (22.3)) by applying FMM.

[41] $w \approx 0.001\ d$ in Newton's iterations of [128] applied to a polynomial of degree $d = 2^{17}$.

[42] Gerschgorin's theorem (cf. [42]) for the matrix $xI_d - A$ defines neighborhoods of its eigenvalues (the roots of p), which helps to analyze and to modify the iterations (cf. [21]).

Every iteration but the first one begins with some values z_j, w_j, and updated values $z_j^{(\text{new})}$ of (22.1), for $j = 1, \dots, d$, and outputs updated values $w_j^{(\text{new})} = W(z_j^{(\text{new})})$ for all j. Then equations (22.3) and (22.5) define $N(z_j^{(\text{new})})$ and $E_j(z_j^{(\text{new})})$ for all j. To update the values w_j, apply [21, equation (17)]:

$$w_i^{\text{new}} = (z_i^{\text{new}} - z_i) \prod_{j \neq i} \frac{z_i^{\text{new}} - z_j}{z_i^{\text{new}} - z_j^{\text{new}}} \Big(\sum_{k=1}^{d} \frac{w_k}{z_i^{\text{new}} - z_k} - 1 \Big), \ i = 1, \dots, d.$$

In the case where $z_i^{\text{new}} = z_i$ if and only if $i > k$ this updating is still used for $i = 1, \dots, k$ but is simplified to

$$w_i^{\text{new}} = w_i \prod_{j=1}^{k} \frac{z_i - z_j}{z_i^{\text{new}} - z_j} \text{ for } i = k+1, \dots, d.$$

Now fast compute the values $\sum_{k=1}^{d} \frac{w_k}{z_i^{\text{new}} - z_k}$ for all i by applying FMM.

Also compute the values

$$R_i = (z_i^{\text{new}} - z_i) \prod_{j \neq i} \frac{z_i^{\text{new}} - z_j}{z_i^{\text{new}} - z_j^{\text{new}}}$$

fast in two ways.

(i) Use the following customary trick (see [29, Remark 3]): approximate the values $\ln(R_i)$ for $i = 1, \dots, d$ fast by applying FMM and then readily approximate $R_i = \exp(\ln(R_i))$ for all i. [29] proves numerical stability of the reduction to FMM of a similar albeit a little more involved rational expressions.

(ii) Observe that

$$\ln \Big(\frac{R_i}{z_i^{\text{new}} - z_j} \Big) = \sum_{j \neq i} \ln \Big(1 - \frac{z_j^{\text{new}} - z_j}{z_i^{\text{new}} - z_j^{\text{new}}} \Big) \text{ for } i = 1, \dots, d$$

and that

$$\ln \Big(1 - \frac{z_j^{\text{new}} - z_j}{z_i^{\text{new}} - z_j^{\text{new}}} \Big) \approx \frac{z_j - z_j^{\text{new}}}{z_i^{\text{new}} - z_j^{\text{new}}} \text{ if } |z_j^{\text{new}} - z_j| \ll |z_i^{\text{new}} - z_j^{\text{new}}|.$$

Let the latter bound hold for $j \in \mathbb{J}_i$ and let $\mathbb{J}_i'$ denote the set of the remaining indices $j \neq i$. Then

$$R_i \approx \exp(\sigma_i) \prod_{j \in \mathbb{J}_i'} \Big(1 - \frac{z_j^{\text{new}} - z_j}{z_i^{\text{new}} - z_j^{\text{new}}} \Big), \ \sigma_i = \sum_{j \in \mathbb{J}_i} \frac{z_j - z_j^{\text{new}}}{z_i^{\text{new}} - z_j^{\text{new}}}, \ i = 1, \dots, d.$$

By applying FMM fast approximate σ_i for all i and compute the values $\prod_{j \in \mathbb{J}'} \frac{z_i^{\text{new}} - z_j}{z_i^{\text{new}} - z_j^{\text{new}}}$ for all i in at most $4 \sum_{i=1}^{d} |\mathbb{J}_i'| - d$ arithmetic operations for $|\mathbb{J}_i'|$ denoting the cardinality of the set $\mathbb{J}_i'$. Compare the latter cost bound with order of d^2 arithmetic operations in the straightforward algorithm.

Clearly the above updating computation is simpler and more stable numerically for $W(z_j)$ versus $N(z_j)$ and even more so versus $E_j(z_j)$, for $j = 1, \dots, d$.

Let us comment on the changes where one incorporates FMM into Newton's, Weierstrass's, and Ehrlich's iterations implicitly applied to the wild factor $f(x)$ of $p(x)$ (see Remark 21.1). The same expression (21.2) defines Ehrlich's iterations for both $f(x)$ and $p(x)$ by virtue of Thm. 10.1. Newton's iterations to a wild factor one additionally subtract the sum $\sum_{j=1}^{d-w} \frac{1}{x - x_j}$ (cf. (21.2)) and can apply FMM to accelerate numerical computation of this sum. To apply Weierstrass's iterations to a wild factor one performs additional division by $t(x)$ (cf. (21.1)) and decreases the degree of the divisor $l'(x)$ from $d - 1$ to $w - 1$.

22.5 d root radii and initialization

The algorithms of [63, 50, 128] approximate all d roots of a polynomial p by applying Newton's iterations at $f(d)$ equally spaced points on a large circle containing inside all d roots for $f(d)$ of order at least $d\log(d)$. Bounds (10.12) immediately define such a circle for a polynomial given by its coefficients.

MPSolve further elaborates upon initialization. It first applies Bini's heuristic recipe of [10], reused in [14, 21], for the approximation of the d root radii by involving all coefficients of p. Given such approximations MPSolve defines k concentric annuli $A(c, \rho_i, \rho'_i)$ and k integers m_i such that $\#(A(c, \rho_i, \rho'_i)) = m_i$ for $i = 1, \dots, k$, $k \le d$, and $\sum_{i=1}^{k} m_i = d$. MPSolve initializes Ehrlich's iterations at d points z_j, $j = 1, 2, \dots, d$, exactly m_i of them being equally spaced on the circle $C(c, (\rho_i + \rho'_i)/2)$. This heuristic initialization recipe turns out to be highly efficient, and one is challenged to test whether it can also help accelerate convergence of Newton's iterations.

The algorithm supporting the next theorem also approximates the d root radii but is certified, deterministic, and faster than Bini's. It was first outlined and briefly analyzed in [120] for fast approximation of a single root radius and then extended in [79, Sec. 4] (cf. also [80, Sec. 5]) to approximation of all root radii of a polynomial. As in [10, 14, 21] it uses all d coefficients of p. A variation of this algorithm in [53] supports the following theorem.

Theorem 22.1. *[53, Prop. 3]. Given the coefficients of a polynomial $p = p(x)$, one can approximate all the d* root radii *$|x_1|, \dots, |x_d|$ within the relative error bound $4d$ at a Boolean cost in $O(d\log(||p||_\infty))$.*

[53] extends the supporting algorithm to approximation of all root radii within a positive relative error bound Δ by performing $\lceil \log_2 \log_{1+\Delta}(2d) \rceil$ DLG root-squaring iterations (10.9).

Combine Thm. 22.1 with estimates for the Boolean cost of performing these iterations implicit in the proof of [120, Cor. 14.3] and obtain

Corollary 22.1. *Given the coefficients of a polynomial $p = p(x)$ and a positive Δ, one can approximate all the d* root radii *$|x_1|, \dots, |x_d|$ within a relative error bound Δ in $\frac{1}{d^{O(1)}}$ at a Boolean cost in $O(d\log(||p||_\infty) + d^2\log^2(d))$.*

The algorithm supporting the corollary and said to be **Alg. 22.1a** computes at a low cost d *suspect annuli* $A(0, r_{j,-}, r_{j,+})$ for $j = 1, \dots, d$ (some of them may pairwise overlap), whose union $\mathbb{U}$ contains all d roots.

We conclude this subsection with a list of **some benefits of the approximation of the d root radii.**

(i) In a subdivision step one can skip testing exclusion and discard a square unless it intersects $\mathbb{U}$.

(ii) The search for real roots can be narrowed to the intersection $\mathbb{I}$ of $\mathbb{U}$ with the real axis (see Fig. 10). [112] has tested a simple real root-finder that approximates the roots in $\mathbb{I}$. According to these tests the algorithm is highly efficient for the approximation of all isolated real roots. Clearly, the well-developed root-finding techniques, e.g., ones using derivatives and higher order derivatives (see [90, 57], and the references therein) can further strengthen these algorithms. By complementing the root radii approach with some advanced novel techniques (see more in Remark 22.2), the paper [53] has noticeably accelerated the real root-finder [62], currently of user's choice.

(iii) Initialization based on Alg. 22.1a should improve global convergence of Ehrlich's, Weierstrass's, and other functional iterations for root-finding as well as of Kirrinnis's factorization algorithm of [60], revisited in the next section. In particular Ehrlich's iterations of MPSolve are

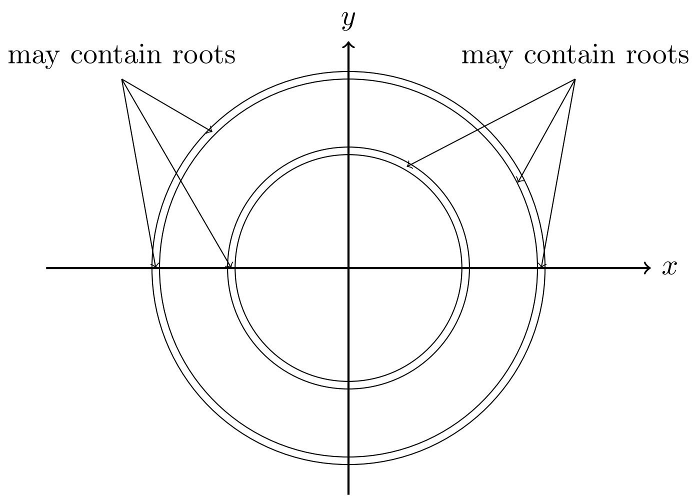


Figure 9: Two approximate root radii define two narrow suspect annuli and four suspect segments of the real axis.

celebrated for their very fast global empirical convergence (right from the start), but it has only been observed with such initialization; moreover, they diverge without it in an example of [115].

(iv) One can readily compute the power sums of the roots lying in any computed annulus isolated from the external roots. By applying the algorithms of Sec. 21.5 to such an annulus that contains a small number of roots, one can compute a low degree factor of p that shares with p the small root set in this annulus; then one can closely approximate these roots at a lower cost. Finally, one can approximate the remaining wild roots by applying the recipes of Sec. 21.2 for explicit or implicit deflation.

22.6 Root-finding in the intersection of the unions of d narrow annuli

Overview and some remarks. Fix an integer $k > 1$ and by applying **Alg. 22.1a** compute the d root radii for the polynomials $t_j(x) = p(x - c_j)$ for k distinct complex numbers c_j, $j = 1, \dots, k$. This defines k unions $\mathbb{U}_j$, each containing all d roots, and clearly, the above benefits (i) – (iv) are accentuated if we confine root-finding to the intersection of all these unions.

The cost of computing such an intersection grows because already for $k = 2$ it may consist of d^2 components, but this can be mitigated. E.g., in case (i) we can avoid computing the intersection: we skip testing exclusion and discard a suspect square unless it intersects all k unions $\mathbb{U}_j$ for $j = 1, \dots, k$, and we can test this property of a square by using $O(kd)$ comparisons.

If the roots form a cluster (cf. Example 21.1), then Alg. 22.1a with shifts can help locate it, but apparently at the computational cost not lower than by means of subdivision with compression. Alg. 22.1a with shifts, however, helps if the d roots do not form a single cluster but lie on or near a circle centered at the origin, in which case Alg. 22.1a without shift outputs just a single circle.

In the next subsection we outline and strengthen a randomized root-finder proposed and analyzed in [95]. It fast approximates all isolated complex roots by relying on application of Alg. 22.1a with three centers that lie far from the roots.

The case of a black box polynomial. Combination of Alg. 22.1a with Taylor's shift does not blow up Boolean cost but was considered non-practical because it destroys sparsity and requires significant increase of the precision of causing numerical problems.

22.7 Root-finding in the intersection of three unions

Next we outline and strengthen the algorithm of [95].

First fix two long shifts of the variable $x \to x - c_j$, for

$$c_j = \gamma\ r_+ \exp(\phi_j \mathbf{i}),\ j = 1, 2,\ \phi_1 = 0,\ \phi_2 = \frac{\pi}{2},\ r_+ \geq r_1 = r_1(0, p),$$

and a sufficiently large γ, e.g., $\gamma = 100$. Then apply Alg. 22.1a to the two polynomials $t(x) = p(x - c_j)$, for $j = 1, 2$ and c_1 and c_2 above, and obtain two families, each made up of d narrow and possibly pairwise overlapping annuli, centered at c_1 and c_2, respectively.

Let ρ denote the maximal width of the annuli of these families; already for a reasonably large γ it is close to the width of every annulus of both families. The d^2 intersections of the d annuli of family 1 with the d annuli of family 2 are close to squares having vertical and horizontal sides of length ρ.

Now collapse all overlapping squares into larger rectangles with the maximal side length $\tilde{\rho} \leq d\rho$ and let their centers form a grid $\mathbb{G}$ of $d_1 d_2$ vertices where $d_i \leq d$ for $i = 1, 2$. The $d_1 d_2$ nodes of the grid $\mathbb{G}$ approximate the d roots within about $\frac{\tilde{\rho}}{\sqrt{2}} \leq \frac{d\rho}{\sqrt{2}}$ or less.

Given a suspect square we can discard it without testing exclusion unless both real and imaginary parts of a node of the grid $\mathbb{G}$ lie within $\tilde{\rho}$ from the real and imaginary parts of the sides of that suspect square. We can verify whether this property holds by performing $2d_1 + 2d_2 \leq 4d$ comparisons.

In [95] we extended the approach to the approximation within $\tilde{\rho}\sqrt{2}$ of all roots by at most d vertices of the grid. We obtained this extension whp by means of application of Alg. 22.1a to the polynomial $t_3(x) = p(x - c_3)$, for $c_3 = \gamma r_+ \exp(\phi_3 \mathbf{i})$, the same r_+ and γ as before, and random ϕ_3 sampled under the uniform probability distribution from the range $[\frac{\pi}{8}, \frac{3\pi}{8}]$.

Namely, in [95] we assumed some separation among the annuli of each family and then proved that whp each of the $d_3 \leq d$ narrow annuli output in the third application of Alg. 22.1a intersects a single square having its center in the grid $\mathbb{G}$; thus whp these d_3 centers approximate all the d roots. Furthermore, the index of an annulus of the third family shows the number of the roots lying in the small rectangle centered at the associated node of the grid $\mathbb{G}$.

In [95] we specified the probability estimate of at least $1 - \epsilon$ for the success of the algorithm provided that the annuli of each family are separated from each other by an annulus with relative width at least $v > 0$ such that $\frac{4\tilde{\rho}\sqrt{2}}{\pi v}(d_1 d_2 - 1)d_1 d_2 \leq \epsilon$, where $\frac{4\sqrt{2}}{\pi} = 1.8006\ldots$.

By virtue of Cor. 22.1 we can ensure a proper bound on the relative width of the output annuli of Alg. 22.1a (defined by the relative error Δ of Cor. 22.1) in order to satisfy the latter estimate.[43]

Remark 22.1. *It would be more costly to compute a specific angle ϕ_3 which would ensure that the intersection of the three families consists of at most d domains. Indeed, there are $M = \binom{d_1 d_2}{2} = \frac{1}{2}(d_1 d_2 - 1)d_1 d_2$ straight lines that pass through the pairs of the $d_1 d_2$ vertices of the grid $\mathbb{G}$, and the desired property of the intersection of the three families would only hold if neither of these straight lines forms an angle close to $\phi_3 + \frac{\pi}{2}$ with the real axis. Verification of this property requires at least M arithmetic operations, that is, $\frac{1}{2}(d^2 - 1)d^2$ where $d_1 = d_2 = d$. Empirically we should have good chances to succeed even if we fix any angle ϕ_3 in the range $[\frac{\pi}{8}, \frac{3\pi}{8}]$ instead of choosing it at random. One can try various heuristic choices of* $\exp(\phi_3 \mathbf{i}) = \frac{a + b\mathbf{i}}{a^2 + b^2}$ *where $|a|$ and $|b|$ are pairs of small positive integers.*

[43] The estimate is stated in [95] in the case where each family consists of precisely d annuli with no overlap, but the proof supports the more general statement above.

Remark 22.2. *Based on some advanced novel techniques the paper [53] applies Alg. 22.1a to the polynomials $t_j(x) = p(x - c_j)$, for $j = 0, 1, 2$, where a polynomial $p(x)$ has integer coefficients, $c_0 = 0$, $c_i = a_i + b_i\mathbf{i}$, and $|a_i|$ and $|b_i|$ for $i = 1, 2$ are bounded non-negative integers. In spite of the limitations on the shift values c_j, the resulting real and complex subdivision root-finders for polynomials given in monomial basis have noticeably accelerated the known algorithms according to a series of extensive tests.*

23 Fast polynomial factorization and multipoint evaluation

Given close initial approximations to the d roots of p, one can first refine them by applying Newton's, Schröder's, Weierstrass's, Ehrlich's, or subdivision iterations and then compute complete factorization (1.2). Alternatively, one can first compute factorization (1.2) based on the initial approximations to the roots and then apply Kirrinnis's refinement of factorization of [60], revisited in this section. The latter way tends to be superior in the case of multiple roots or clusters of roots, where root-finding and therefore root-refining computations may require much higher precision than factorization (see Observation 1.2). Such application of factorization can be initiated at the late stages of subdivision process if they define factors of small degree (see the end of Sec. 23.6).

Factorization into s factors can be viewed as simultaneous explicit deflation of these factors. As in the case of explicit deflation, Boolean complexity can be nicely bounded, but numerical stability problems caused by blowing up the coefficients and the loss of sparsity are likely unless the factors have low degree or are of the form $(x - c)^m$ (cf. Sec. 21.2).

23.1 Contents

In this section we recall some known algorithms for polynomial factorization, outline a direction to advancing them, link them to subdivison iterations and other root-finders, and extend this advance to numerical multipoint polynomial evaluation (MPE).

In the next subsection we link polynomial root-finding and factorization. In Sec. 23.3 we recall our algorithms of 1994 and 1995 for polynomial factorization and propose to combine them with Kirrinnis's refinement. In Sec. 23.4 we cover (a simplified version of) Kirrinnis's refinement of polynomial factorization of [60]. In Sec. 23.5 we recall its complexity estimates. In Sec. 23.6 we discuss its bottleneck stage of initialization and at the end provision for combining subdivision with factorization.

In Sec. 23.7 we extend factorization to numerical MPE, based on our novel combination of FMM with polynomial factorization, and in Sec. 23.8 we briefly recall Moenck–Borodin's algorithm for non-numerical MPE.

23.2 Linking polynomial root-finding and factorization

Problem 1 of root-finding for $p(x)$ and Problem 2 of its complete factorization into the product of linear factors are closely related to one another.

Thm. 1.5 and Remark A.3 specify recovery of the roots from linear factors.

Conversely, suppose a set of pairwise isolated discs $D(c_j, \rho_j)$, together with their indices m_j summed to d has been computed. E.g., such discs can be output by a subdivision iteration; alternatively they can be computed from d approximations to the d roots output by another root-finder or from the output of the algorithm of Sec. 22.7. Then we can immediately define approximate factorization as follows:

$$p(x) \approx q(x) = \prod_{j:\ |c_j| \leq 1} (x - c_j)^{m_j} \prod_{j:\ |c_j| > 1} (xc_j - 1)^{m_j}$$

(cf. (1.2) where $u_j = 1$, $v_j = c_j$ if $|c_j| \leq 1$ and where $u_j = c_j$, $v_j = 1$ if $|c_j| > 1$).

To obtain a closer approximate factorization one can first compute Cauchy sums and then apply Alg. 21.3 or 21.4 (see Sec. 21.5). In this case, however, the factors $(x - c_j)^{m_j}$ and $(xc_j - 1)^{m_j}$ for large integers m_j can lose their compact sparse representation and can turn into unstructured dense polynomials with coefficients of large size.

We omit estimation of a relative error bound $\epsilon = |p(x) - q(x)|/|p|$. Clearly, it converges to 0 as so does the relative maximal error of the root approximation $\delta = |1 - \max_j |x_j/z_j||$, which converges exponentially fast when Newton's, Weierstrass's, or Ehrlich's iterations are applied.

23.3 Divide and conquer factorization based on higher order derivatives or on resultant inequalities

The algorithm of [120], outlined in Appendix A, computes complete polynomial factorization by means of recursive divide and conquer splitting of a polynomial $p(x)$ into the product of non-constant factors. The degrees of the factors could decrease slowly, in up to $d - 1$ recursive splitting steps, but in the algorithm of [90] the maximal degree of the factor in every splitting is at most $\frac{11}{12}$-th of an input degree. Consequently, the overall complexity of complete factorization exceeds the cost of splitting by at most a constant factor and is smaller than in [120] by a factor of d (see further comments in Appendix A.3). This is achieved based on combining factorization of $p(x)$ with factorization of its higher order derivatives, making the algorithm quite involved in spite of its low asymptotic complexity.

The papers [81, 82] proposed randomized and deterministic algorithms, respectively, based on some resultant inequalities. The algorithms output approximate factorization of a polynomial $p(x)$ of degree d into the product of factors of degrees less than $d/2$ by applying $O(d^2)$ arithmetic operations. Recursive application of these algorithms to the computed factors would also numerically decompose $p(x)$ into the product of linear factors.

This alternative approach is more transparent than that [90] and is much simpler to implement, except that the papers [81, 82] have not elaborated upon extension of their error estimates from a single recursive step to the entire factorization. Such extension is tedious, but seems to be rather straightforward.

The work on it, however, has lost motivation and stopped in 1995, when the divide and conquer randomization of [90] turned out to be nearly optimal. Looking back now, we can see that Kirrinnis's algorithm of [60] greatly simplifies, if not trivializes, the error estimation of the algorithms of [81, 82].

This study is interesting theoretically but the algorithms of [81, 82] share with [120, 90] the implementation problems where high degree factors appear in recursive splitting processes.

23.4 Kirrinnis's refinement of polynomial factorization

Newton's refinement of factorization of a polynomial p into any number s of factors proposed by Kirrinnis in [60] extends Schönhage's algorithm of [120] for $s = 2$ and together with factorization of $p(x)$ refines the partial fraction decomposition *(pfd)* of the reciprocal $1/p(x)$. Here is a simplified presentation of his algorithm, and we discuss extension of factorization to pfd in Sec. 23.6.

Given a monic polynomial p of (1.1), we seek pairwise prime monic polynomials $f_1, \ldots, f_s$ and polynomials $h_1, \ldots, h_s$, deg $h_j <$ deg $f_j = d_j$, $j = 1, \ldots, s$, defining the factorization and the pfd, such that

$$p = f_1 \cdots f_s \text{ and } \frac{1}{p} = \frac{h_1}{f_1} + \cdots + \frac{h_s}{f_s}. \tag{23.1}$$

Suppose that initially $2s$ polynomials $f_{j,0}$ and $h_{j,0}$, $j = 1, \ldots, s$, are given (see Sec. 23.6) such that

$$f_0 = f_{1,0} \cdots f_{s,0} \approx p, \ f_{j,0} \approx f_j \text{ for all } j, \tag{23.2}$$

$$\frac{1}{f_0} = \frac{h_{1,0}}{f_{1,0}} + \cdots + \frac{h_{s,0}}{f_{s,0}} \ \text{ and } \deg(h_{j,0}) < \deg(f_{j,0}) \text{ for all } j. \tag{23.3}$$

Then define Kirrinnis's refinement of both initial approximation (23.2) and pfd (23.3) by performing the following computations:

$$q_{j,k} = \frac{f_k}{f_{j,k}}, \ h_{j,k+1} = (2 - h_{j,k} q_{j,k}) h_{j,k} \bmod f_{j,k}, \ j = 1, \ldots, s, \tag{23.4}$$

$$f_{j,k+1} = f_{j,k} + (h_{j,k+1} p \bmod f_{j,k}), \ j = 1, \ldots, s, \tag{23.5}$$

$$f_{k+1} = f_{1,k+1} \cdots f_{s,k+1} \tag{23.6}$$

for $k = 0, 1, \ldots$ Substitute equations (23.4) to rewrite equations (23.5):

$$q_{j,k} = \frac{f_k}{f_{j,k}}, \quad f_{j,k+1} = f_{j,k} + ((2 - h_{j,k} q_{j,k}) h_{j,k} p \bmod f_{j,k}), \ j = 1, \ldots, s,$$

The refinement iterations are simplified where the factors $l_{j,k}$ have small degrees, e.g.,

$$h_{j,k+1} = (2 - h_{j,k} f_k'(z_{j,k})) h_{j,k}$$

and both $h_{j,k}$ and $h_{j,k+1}$ are constants for all k where $f_{j,k} = x - z_{j,k}$ is a monic linear factor and $f_k'(x)$ denotes the derivative of the polynomial $f_k(x)$.

23.5 The overall complexity of Kirrinnis's refinement

Three assumptions. Analyzing his algorithm, Kirrinnis assumes that (i) all roots have been moved into the unit disc $D(0,1)$, e.g., by means of scaling the variable, (ii) the s root sets of the s factors $f_1, \ldots, f_s$ as well as the s root sets of the initial approximations $f_{1,0}, \ldots, f_{s,0}$ to these factors are pairwise well isolated from each other, and (iii) a given initial factorization (23.2) and pfd (23.3) are sufficiently close to those of (23.1).

Assumption (ii) is satisfied for the factors $f_{j,0}$ of the form $(x - c_j)^{m_j}$ where c_j is a point in the component C_j formed by suspect squares at a subdivision step, m_j is the number of roots in that component, and s is the number of components. Likewise, we can fulfill property (ii) when we define factors $f_{j,0}$ from approximations z_j to the roots x_j computed by another root-finder or from the intersection of the unions in Sec. 22.7 provided that we cover every intersections by a nearly minimal isolated disc and compute its index.

Assumption (iii) is satisfied where the factors are defined by sufficiently close approximations to the roots.

Under these assumptions (i) –(iii), **Kirrinnis's theorem** below states that k iterations of his algorithm for $f_k = f_{1,k} \cdots f_{s,k}$, sufficiently large k, and $\mu(u) = O((u \log(u))$, ensure the approximation error norm bounds

$$\delta_k = \frac{|p - f_k|}{|p|} \le 2^{-b}, \quad \sigma_k = \left|1 - h_{1,k} \frac{f_k}{f_{1,k}} - \ldots - h_{s,k} \frac{f_k}{f_{s,k}}\right| \le 2^{-b}$$

at the Boolean cost in $O(d\mu(b'+d)\log(d))$. By applying the algorithms of [120] $s-1$ times one can obtain similar estimates, but Kirrinnis's algorithm streamlines the supporting computations. Here are his detailed estimates.

Theorem 23.1. *(See [60, Cor. 1.8 and Def. 3.16].) Let s, d, $d_1,\dots,d_s$ be fixed positive integers such that*

$$s \ge 2,\ \ d_1+\dots+d_s = d.$$

Let p, $f_{i,0}$, $h_{i,0}$, $i=1,\dots,s$, be $2s+1$ fixed polynomials such that p, $f_{1,0},\dots,f_{s,0}$ are monic and

$$\deg(f_{i,0}) = d_i > \deg(h_{i,0}),\ \ i=1,\dots,s; \deg(p) = d.$$

Let all roots of the polynomial p lie in the disc $D(0,1)$. Furthermore, let

$$|p|\delta_0 = |p - f_{1,0}\cdots f_{s,0}| \le \min\Big\{\frac{2^{-9d}}{(sh)^2}, \frac{2^{-4d}}{(2sh^2)^2}\Big\},$$

$$\sigma_0 = \Big|1 - \frac{f_0 h_{1,0}}{f_{1,0}} - \dots - \frac{f_0 h_{s,0}}{f_{s,0}}\Big| \le \min\{2^{-4.5d}, \frac{2^{-2d}}{h}\},$$

$$f^{(0)} = \prod_{j=1}^{s} f_{j,0} \text{ and } h = \max_{i=1,\dots,s} |h_i|$$

for h_i of (23.1). Let

$$l = l(d_1,\dots,d_s) = \sum_{i=1}^{s} \frac{d_i}{d}\log\Big(\frac{d_i}{d}\Big)$$

(which implies that $l \le \log_2(d)$ for all choices of $s, d_1, d_2, \dots, d_s$ and that $l = O(1)$ for $s = 2$ and all choices of d_1 and d_2).

Finally let $b \ge 1$, $b_1 \ge 1$, and k in $O(\log(b+b_1))$ be sufficiently large. Then in k steps Algorithm (23.4)-(23.6) computes the polynomials $f_{1,k}$, $h_{1,k}$, $\dots$, $f_{s,k}$, $h_{s,k}$ such that $f_{1,k},\dots,f_{s,k}$ are monic,

$$\deg(h_{i,k}) < \deg\ \ (f_{i,k}) = d_i,\ \ i=1,\dots,s,\ \ \delta_k|p| < 2^{-b},\ \text{ and }\ \ \sigma_k < 2^{-b_1}.$$

These steps involve $O((dl\log(d))\log(b+b_1))$ arithmetic operations in $O(b+b_1)$-bit precision; they can be performed by using $O(\mu((b+b_1)dl))$ Boolean operations, that is, $O(\mu((b')dl))$ for $b' = b+b_1$. Moreover,

$$\max_{1\le i\le s} |f_{i,k} - f_i| < 2^{3d} h\delta_0|p|$$

where $p = f_1\cdots f_s$ and $f_{1,k},\dots,f_{s,k}$ are the computed approximate factors of p.

23.6 Computation of an initial pfd

1. Transition from factorization to pfd. Given factorization $p = f_1\dots f_s$ one can compute pfd (23.1) by applying some well-known techniques of computer algebra [41] or can reduce the problem to solving an associated structured linear system of equations; see [136, 92] for fast numerical solution algorithms. If $s = d$, $f_j = x - x_j$, for $j = 1,\dots,d$, and all roots x_j are distinct, we specify pfd as follows:

$$\frac{1}{f(x)} = \sum_{j=1}^{d} \frac{h_j}{x - z_j},\ \text{ for } h_j = \frac{1}{f'(z_j)},\ j = 1,\dots,d. \tag{23.7}$$

We can compute the vales $h_1, \dots, h_s$ by using $O(d \log(d))$ arithmetic operations but may prefer to increase the cost to $O(d^2)$ and to ensure numerical stability.

2. Numerical transition from factors to pfd. Given a set of isolated discs with their indices summed to d, one can compute a quite close approximate factorization $p \approx f_1 \cdots f_s$ by applying the recipes of Sec. 23.2, although the output may still violate assumption (iii) of Thm. 23.1.

In [60] Kirrinnis refines this factorization by extending from the case $s = 2$ the algorithm of [120, Secs. 10 and 11], which we recall in Appendix B. He first observes that

$$\frac{h_{i,0}(x)}{f_{i,0}(x)} = \frac{1}{2\pi \mathbf{i}} \int_{C(c_i,\rho_i)} \frac{dz}{f_0(z)}$$

provided that (23.3) holds and all roots of a factor $f_{i,0}$ of a polynomial f_0 lie in a θ_i-isolated disc $D(c_i, \rho_i)$, for $\theta_i > 1$. If this holds for $i = 1, \dots, s$ and if the values $\theta_i - 1$ exceed a fixed positive tolerance bound for all i, then one can closely approximate these s integrals by means of their discretization at reasonably many equally spaced points on the boundary circles $C(c_i, \rho_i)$.

The algorithm is quite involved and tends to be the bottleneck of the entire factorization of [60], but this is because it has been devised to cover the case where the root sets of the factors are $(1 + 1/d)$-isolated. [82, Appendix] specifies dramatic simplification in the case where the root sets is isolated, and this is always the case for the factorization of [81, 82] as well as for one defined from the isolated components computed in subdivision process.

One should not forget about the “side effects" of Sec. 21.2, that is, potential blow up of the coefficients and loss of sparsity of the numerators h and denominators f in the pfd computed in the process of refinement of factorization.

One can avoid these “side effects" by initializing factorization only where all roots are covered with isolated discs D_j having small indices $\#(D_j)$. In particular, one can delay transition from subdivision to factorization until this provision is fulfilled (cf. Example 21.1).

23.7 Fast Numerical Multipoint Polynomial Evaluation (MPE)

Multipoint polynomial evaluation (MPE) with FMM. In context of polynomial root-finding we further recall that the algorithm of [92, 94] evaluates a polynomial of degree d at $O(d)$ points at the same asymptotic cost of performing FMM, that is, by using $O(d \log^2(d))$ arithmetic operations with the precision $O(\log(d))$. The algorithm successively reduces MPE to multiplication by a vector of Vandermonde, Cauchy and finally HSS matrices; the latter operation is performed fast with FMM. The above reduction steps also require $O(d \log^2(d))$ arithmetic operations with the precision $O(\log(d))$, but the overhead constant hidden in the "O" notation of the estimated overall complexity of MPE increases versus that of FMM in such a 3-step reduction.

Next we propose an alternative reduction of MPE to FMM by means of factorization of a monic polynomial, possibly computed by Kirrinnis's algorithm of Sec. 23.4 with our initialization outlined in Sec. 23.6.

Algorithm 23.1. Reduction 1 of MPE to FMM via factorization.

Given a monic polynomial $p(x)$ of degree d and m complex points $v_1, \dots, v_m$,
(i) fix a sufficiently large scalar ν and compute an approximate factorization

$$\tilde{q}(x) = \prod_{j=1}^{d} (x - \tilde{y}_j) \approx q(x) := p + \nu - p(0); \tag{23.8}$$

(ii) evaluate the polynomial $\tilde{q}(x)$ at m given points $v_1, \dots, v_m$, and

(iii) compute $\tilde{q}(v_j) + p_0 - \nu \approx p(v_j)$ for $j = 1, \dots, m$.

At stage (i) $q(x)$ is closely approximated by the polynomial $p_d \prod_{j=1}^{d}(x - \nu^{\frac{1}{d}} \zeta_d^j)$ for $\zeta_d = \exp(\frac{2\pi \mathbf{i}}{d})$ and large ν; hence its refined factorization can be computed very fast, e.g., with MPSolve, and possibly even for moderately large values of ν. One can perform stage (ii) by applying an FMM variation of [29] and only needs to perform m subtractions at stage (iii); one can compensate for numerical cancellation in these subtractions by adding about $\log_2(\nu)$ bits of working precision throughout the algorithm.

For a large class of inputs, limited precision increase is sufficient, and one can further limit it by applying homotopy continuation techniques (cf. [63]), that is, by recursively moving the point ν towards the origin 0 and updating the polynomial $q(x)$ and factorization (23.8).

Recursive processes in the above papers continue until the roots of $p(x)$ are approximated closely, but our process can stop much earlier, that is, as soon as the incurred precision increase $\log_2(\nu)$ becomes reasonably small.

Next consider three other alternative reductions of MPE to FMM.[44]

Algorithm 23.2. Reduction 2 of MPE to FMM via factorization.

INPUT: a monic polynomial $p(x)$ of degree d and m complex points $v_1, \dots, v_m$,

COMPUTATIONS:

(i) Fix a sufficiently large scalar ν and compute an approximate factorization

$$q(x) \approx \tilde{q}(x) = d \prod_{j=1}^{d} (x - \tilde{y}_j) \text{ for } q'(x) = p(x) \text{ and } q(0) = \nu - p(0).$$

(ii) Compute the values $\tilde{q}(v_i)$ for $i = 1, \dots, m$.

(iii) Compute the values $r(v_i) = \frac{\tilde{q}'(v_i)}{\tilde{q}(v_i)}$ for $i = 1, \dots, m$.

(iv) Compute and output the values $\tilde{p}(v_i) = r(v_i)\tilde{q}(v_i)$ for $i = 1, \dots, m$.

This variation does not require adding order of $\log(\nu)$ bits to working precision. Notice that $r(x) = \frac{\tilde{q}'(x)}{\tilde{q}(x)} = \sum_{j=1}^{d} \frac{1}{x - \tilde{y}_j}$, and so at stage (iii) we apply the classical FMM of [44, 26] for evaluation of $r(v_i)$, but we still need the FMM variant of [29] for the evaluation of $\tilde{q}(v_i)$ at stage (ii). In the next two algorithms we do not use this variation. We again require order of $\log(\nu)$ extra bits of working precision in the first but not in the second of these algorithms.

Algorithm 23.3. Reduction 3 of MPE to FMM via factorization.

INPUT: a monic polynomial $p(x)$ of degree d and m complex points $v_1, \dots, v_m$,

COMPUTATIONS:

(i) Fix a sufficiently large scalar ν and compute approximate factorization

$$q_+(x) \approx \tilde{q}_+(x) = d \prod_{j=1}^{d} (x - \tilde{y}_j), \ q_-(x) \approx \tilde{q}_-(x) = d \prod_{j=1}^{d} (x - \tilde{z}_j)$$

for $q'_+(x) = q'_-(x) = p(x)$, $q_+(0) = \nu - p(0)$, and $q_-(0) = -\nu - p(0)$.

[44] One can also apply Alg. 23.1 to a polynomial $t(x)$ with subsequent evaluation of $p(v_j) = t'(v_j)$ for all j by means of the algorithm of [L76], [BS83].

(ii) Compute $r_+(v_i) \approx \frac{\tilde{q}'_+(v_i)}{\tilde{q}_+(v_i)}$ *and* $r_-(v_i) \approx \frac{\tilde{q}'_-(v_i)}{\tilde{q}_-(v_i)}$ *for* $i = 1, \ldots, m$.

(iii) Compute and output $\tilde{p}(v_i) = \frac{2\nu r_+(v_i) r_-(v_i)}{r_-(v_i) - r_+(v_i)}$ *for* $i = 1, \ldots, m$.

Algorithm 23.4. Reduction 4 of MPE to FMM via factorization.

INPUT as in Alg. 23.3.

COMPUTATIONS:

(i) Fix a sufficiently large scalar ν *and compute approximate factorization*

$$p_+(x) \approx \tilde{p}_+(x) = p_d \prod_{j=1}^{d} (x - \tilde{y}_j), \ p_-(x) \approx \tilde{q}_-(x) = p_d \prod_{j=1}^{d} (x - \tilde{z}_j)$$

where $p_+(x) - p(x) + p(0) = p(x) - p(0) - p_-(x) = \nu$.

(ii) Compute $r_+(v_i) = \frac{\tilde{p}'_+(v_i)}{\tilde{p}_+(v_i)}$ *and* $r_-(v_i) = \frac{\tilde{p}'_-(v_i)}{\tilde{p}_-(v_i)}$ *for* $i = 1, \ldots, m$.

(iii) Compute and output $\tilde{p}(v_i) = \frac{r_+(v_i) + r_-(v_i)}{(r_-(v_i) - r_+(v_i))\nu} + p(0)$ *for* $i = 1, \ldots, m$.

If we seek the roots of p, then the above algorithms enable us to accelerate the key stage of approximation of p based on fast numerical factorization of $q(x)$ or of a pair of $q_+(x)$ and $q_-(x)$ or $p_+(x)$ and $p_-(x)$.

Initial numerical tests of Algorithm 23.4 have been performed by Qi Luan at CUNY for random input polynomials of degree 40,000. Their coefficients were complex numbers with real and imaginary parts uniformly generated at random in the ranges $[-30, 30]$ and $[-30\,\mathbf{i}, 30\,\mathbf{i}]$, respectively. The 40,000 knots of evaluation were also randomly generated in the same way as the coefficients. All computations were done by using 424 bits of precision, and the output values have been obtained within the relative error bound $2.62e{-}8$.

The evaluation with Horner's algorithm took 1,256 seconds, while the MPSolve+FMM method took 2,683 seconds, with the break down into 2,179 seconds for the MPSolve part and 503 seconds for the FMM part.

For MPSolve the publicly released version was used. A significantly faster version has been developed by Dario Bini and Leonardo Robol [11] but not released and can be further accelerated by means of our recipes in Section 22.4.

The tests also show that the ratio of the time used for Horner's algorithm and for FMM increases as the degree d of an input polynomial increases.

Further tests are planned with the algorithms of this subsection, accelerated MPSolve, and the accelerated factorization of $p(x)$ of Secs. 23.4 and 23.6.

23.8 Fast non-numerical MPE

For an alternative, **Moenck-Borodin's algorithm for MPE** of [73] was the first one with arithmetic complexity $O(d \log^2(d))$ for the evaluation of p at $O(d)$ points.[45] The Boolean cost of performing that algorithm with a high precision is nearly optimal [60, 109], but the algorithm fails when it is applied numerically, with double precision, for $d > 50$. One can counter this deficiency by applying some tools from Computer Algebra. Namely, we can evaluate the polynomial over Gaussian integers (by scaling we can reduce to this case approximations with binary numbers defined by finite number of bits to the real and imaginary parts of complex numbers) and perform most

[45]See its extensions to multivariate MPE in [65, 48].

of computations with a reasonably low precision. To do this we first apply the algorithm of [73] modulo a number of fixed primes of moderate size and then recover the integer output by means of applying the Chinese Remainder algorithm.

Finally, **Horner's algorithm** performs at a much higher arithmetic cost but avoids the growth of working precision, ends with a slightly superior Boolean cost bound, and so far has been remaining the method of choice when the designers of polynomial root-finders use MPE.

PART VI

24 Conclusions

Our study leads to some challenging projects.

1. Elaboration upon and implementation of our black box subdivision root-finders and their simplification for a polynomial given in monomial basis – with its coefficients. [The implementation work has began in [51] with incorporation of our exclusion tests into the algorithm of [22], continued in [53, 54], and has already showed that our progress is for real.

2. Further analysis, comparison, combination, and extension of our technical novelties for devising efficient polynomial root-finders, such as Cauchy sum techniques, exclusion tests, root-squaring, compression of a disc, randomized root-counting, error detection and correction, and root radii approximation.

3. Estimation of precision guarantees for a fixed tolerance to output errors or "empirical precision guarantees", e.g., by means of doubling the working precision recursively, possibly with implicit deflation.

4. Analysis, implementation and testing of Ehrlich's iterations of MPSolve, Newton's, Weierstrass's, and Schröder's iterations incorporating our novelties of initialization based on root radii approximation, our deflation techniques, and reduction to FMM.

5. Enhancing the power of these and other known efficient root-finders by properly combining them with various known and our novel techniques such as FMM and our root radii approximation algorithms.

6. Further study of our novelties and their extensions, e.g., exploration of the links of our root-finding algorithms to matrix eigenvalue computation.[46]

7. Elaboration upon, testing, and implementation of our outline of fast algorithms for polynomial factorization and multipoint evaluation in Sec. 23.

8. New attempts of synergistic combination of the known and novel techniques and possibly devising efficient heuristic sub-algorithms for polynomial root-finders.

9. Proof of fast global convergence of Weierstrass's and Ehrlich's iterations initialized based on the techniques of Secs. 22.5 – 22.7 or extension of the divergence results of [117, 115] under that initialization.

10. Estimation of Boolean complexity of root-finding and factorization for black box, sparse, and shifted sparse polynomials, including the complexity of semi-certified solution algorithms.

[46]Remark 8.1 points out such links. The pioneering papers [15, 16] and the subsequent advanced works in [12, 5] explore the structure of the companion matrix of a polynomial. On root-finding via eigen-solving also see our papers by ourselves and with co-authors [85, 15, 16, 91, 102, 104, 103, 110, 111, 113]. For example, the paper [113] proposed, analyzed, and supported with the results of initial tests a number of non-trivial real polynomial root-finders of this kind.

APPENDIX

A Recursive factorization of a polynomial and extensions to root-finding and root isolation

In this section we briefly recall some algorithms and complexity estimates from [120, 122]. We first recursively decompose a polynomial into the product of linear factors and then extend the factorization to approximation and isolation of the roots.

A.1 Auxiliary norm bounds

Recall two simple lemmas.

Lemma A.1. *It holds that* $|u+v| \le |u|+|v|$ *and* $|uv| \le |u|\ |v|$ *for any pair of polynomials* $u = u(x)$ *and* $v = v(x)$.

Lemma A.2. *Let* $u = \sum_{i=0}^{d} u_i x^i$ *and* $|u|_2 = (\sum_{i=0}^{d} |u_i|^2)^{\frac{1}{2}}$. *Then* $|u|_2 \le |u|$.

The following lemma relates the norms of a polynomial and its factors.

Lemma A.3. *If* $p = p(x) = \prod_{j=1}^{k} f_j$ *for polynomials* $f_1, \dots, f_k$ *and* $\deg p \le d$, *then* $\prod_{j=1}^{k} |f_j| \le 2^d |p|_2 \le 2^d |p|$.

Proof. The leftmost bound was proved by Mignotte in [69]. The rightmost bound follows from Lemma A.2. $\square$

Remark A.1. *[122, Lemma 2.6] states with no proof a little stronger bound* $\prod_{j=1}^{k} |f_j| \le 2^{d-1}|p|$ *under the assumptions of Lemma A.3. From various factors of the polynomial* $p(x) = x^d - 1$ *such as* $\prod_{j=1}^{\frac{d}{2}} (x - \zeta_d^j)$ *for even* d *and* ζ *of (1.3), one can see some limitations on strengthening this bound further.*

A.2 Errors and complexity

Suppose that we split a polynomial p into a pair of factors over an isolated circle and recursively apply this algorithm to the factors until they become linear of the form $ux + v$; some or all of them can be repeated. Finally, we arrive at complete approximate factorization

$$p \approx p^* = p^*(x) = \prod_{j=1}^{d} (u_j x + v_j). \tag{A.1}$$

Next, by following [120, Sec. 5], we estimate the norm of the residual polynomial

$$\Delta^* = p^* - p. \tag{A.2}$$

We begin with an auxiliary result.

Theorem A.1. *Let*

$$\Delta_k = |p - f_1 \cdots f_k| \leq \frac{k\epsilon|p|}{d}, \tag{A.3}$$

$$\Delta = |f_1 - fg| \leq \epsilon_k |f_1|, \tag{A.4}$$

for some non-constant polynomials $f_1, \ldots, f_k, f$ and g such that

$$\epsilon_k d 2^d \prod_{j=1}^{k} |f_j| \leq \epsilon. \tag{A.5}$$

Then

$$|\Delta_{k+1}| = |p - fgf_2 \cdots f_k| \leq \frac{(k+1)\epsilon|p|}{d}. \tag{A.6}$$

Proof. $\Delta_{k+1} = |p - f_1 \cdots f_k + (f_1 - fg)f_2 \cdots f_k|$. Apply Lemma A.1 and deduce that $\Delta_{k+1} \leq \Delta_k + |(f_1 - fg)f_2 \cdots f_k|$ and, furthermore, that

$$|(f_1 - fg)f_2 \cdots f_k| \leq |f_1 - fg| \; |f_2 \cdots f_k| = \Delta |f_2 \cdots f_k|.$$

Combine the latter inequalities and obtain $\Delta_{k+1} \leq \Delta_k + \Delta|f_2 \cdots f_k|$. Combine this bound with (A.3)-(A.5) and Lemmas A.1 and A.3 and obtain (A.6). □

Write $f_1 := f$ and $f_{k+1} = g$. Then (A.6) turns into (A.3) for k replaced by $k+1$. Now compute one of the factors f_j as in (A.4), apply Thm. A.1, then recursively continue splitting the polynomial p into factors of smaller degrees, and finally arrive at factorization (A.1) with

$$|\Delta^*| \leq \epsilon|p|$$

for Δ^* of (A.2). Let us call this computation *Recursive Splitting Process* provided that it begins with $k = 1$ and $f_1 = p$ and ends with $k = d$.

Theorem A.2. *In order to support (A.3) for $0 < \epsilon \leq 1$ and $j = 1, 2, \ldots, d$ in the Recursive Splitting Process it is sufficient to choose ϵ_k in (A.4) satisfying*

$$\epsilon_k \; d \; 2^{2d+1} \leq \epsilon \text{ for all } k. \tag{A.7}$$

Proof. Prove bound (A.3) by induction on j. Clearly, the bound holds for $k = 1$. It remains to deduce (A.6) from (A.3) and (A.7) for any k. By first applying Lemma A.3 and then bound (A.3), obtain

$$\prod_{i=1}^{k} |f_i| \leq 2^d |\prod_{i=1}^{k} f_i| \leq 2^d \left(1 + \frac{k\epsilon}{d}\right) |p|.$$

The latter bound cannot exceed $2^{d+1}|p|$ for $k \leq d$, $\epsilon \leq 1$. Consequently (A.7) ensures (A.5), and then (A.6) follows by virtue of Thm. A.1.

□

Remark A.2. *The theorem implies that we can ensure the output precision bound b' by working with the precision of*

$$\bar{b} \geq \bar{b}_{\text{inf}} = 2d + 1 + \log_2 d + b'$$

bits throughout the Recursive Splitting Process.

A.3 Overall complexity of recursive factorization

The overall complexity of recursive factorization is dominated by the sum of the bounds on the complexity of all splittings into pairs of factors. The algorithm of [90] adopts recursive factorization of [120] but decreases the overall complexity bound by roughly a factor of d. This is due to additional techniques, which ensure that in every recursive splitting of a polynomial into the product of two factors the ratio of their degrees is at most 11. Consequently $O(\log(d)$ recursive splitting steps are sufficient in [90], and so the overall complexity of root-finding is proportional to that of the first splitting. In contrast [120] allows a linear factor in any splitting and thus allows up to $d-1$ splittings, with each of the first $0.5d$ of them almost as expensive as the very first one (cf. Example 21.1).

A.4 From factors to roots

Theorem A.3. *[122, Thm. 2.7]. Suppose that*

$$p = p_d \prod_{j=1}^{d} (x - x_j),\ \tilde{p} = \tilde{p}_d \prod_{j=1}^{d} (x - y_j),\ |\tilde{p} - p| \le \epsilon |p|,\ \epsilon \le 2^{-7d}$$

and

$$|x_j| \le 1 \text{ for } j = 1, \dots, d' \le d,\ |x_j| \ge 1 \text{ for } j = d'+1, \dots, d.$$

Then up to reordering the roots it holds that

$$|y_j - x_j| < 9\epsilon^{\frac{1}{d}} \text{ for } j = 1, \dots, d';\ \left|\frac{1}{y_j} - \frac{1}{x_j}\right| < 9\epsilon^{\frac{1}{d}} \text{ for } j = d'+1, \dots, d.$$

By virtue of Thm. A.3 for $b' = O(bd)$ we can bound the Boolean complexity of the solution of Problem 1 by increasing the estimate for the complexity of factorization in Sec. A.3 by a factor of d.

Corollary A.1. Boolean complexity of the solution of Problem 1. *Given a polynomial p of degree d and a real $b' \ge 7d$, one can approximate all roots of that polynomial within the error bound $2^{-b'}$ at a Boolean cost in $O(\mu(b'+d)d^2 \log(d)(\log^2 d + \log(b'))) = \tilde{O}((b'+d)d^2)$.*

Remark A.3. *The study in this appendix covers the solution of Problem 1 for the worst case input polynomial p. If its roots are isolated from each other, then the upper estimates of Thm. A.3 and Cor. A.2 can be decreased by a factor of d (cf. [90]). It seems that by extending [90] one can prove that the complexity bound of the corollary can be decreased by a factor of d/m if the roots lie in s pairwise isolated discs D_i such that $\#(D_i) = m_i$ for $i = 1, \dots, s$ and $\max_{i=1}^{s}\{m_i\}$.*

Based on his results recalled in this appendix Schönhage in [120, Sec. 20] estimates the Boolean complexity of the following well-known problem.

Problem 5. *Polynomial root isolation.* Given a polynomial p of (1.1) that has integer coefficients and only simple roots, compute d disjoint discs on the complex plane, each containing exactly one root.

Corollary A.2. Boolean complexity of polynomial root isolation. *Suppose that a polynomial p of (1.1) has integer coefficients and has only simple roots. Let σ_p denotes its* root separation*, that is, the minimal distance between a pair of its roots. Write $\epsilon := 0.4\sigma_p$ and $b' := \log_2(\frac{1}{\epsilon})$. Let $\epsilon < 2^{-d}$ and let $m = m_{p,\epsilon}$ denote the maximal number of the roots of the polynomial $p(x)$ in ϵ-clusters of its roots. Then Problem 5 can be solved in Boolean time $\tilde{O}(bdm)$ for $b = \frac{b'}{m}$.*

B Newton's refinement of splitting a polynomial into the product of two factors

B.1 A refinement algorithm

By computing Cauchy sums and then applying Alg. 21.3 or 21.4 (see Sec. 21.5) one can approximate splitting of a polynomial p into the product of two polynomials $f_{1,0}$ and $f_{2,0}$,

$$p \approx f_{1,0} f_{2,0}, \tag{B.1}$$

at a nearly optimal Boolean cost $O(d\log(d)\mu(b+d))$ (cf. [120, 90]). The cost increases proportionally to $\mu(b+d)$ as b exceeds d and grows, but for large b one can save a factor of $\log(d)$ by applying Kronecker's map for multiplication of polynomials with integer coefficients (see [39], [78, Sec. 40]).

Schönhage in [120, Secs. 10–12] has elaborated upon this saving. In particular he devised advanced algorithms for Newton's refinement of splitting a polynomial into factors, which enabled him to decrease the overall Boolean cost of splitting (B.1) to $O(d\mu(b+d))$. The resulting decrease of the complexity bound of root-finding in [120] was only by a factor of $\log(d)$, versus a factor of d in [90]. Besides Kirrinnis in [60], Gourdon in [40], and ourselves, the researchers little used the techniques of [120]. Important progress on root-finding with extremely high precision by means of distinct techniques have been reported in [49]. In our present paper, however, we efficiently combine some techniques of [120] with ours. Next we outline some of Schönhage's algorithms and estimates.

Given an initial approximate splitting (B.1) one can update it as follows:

$$p \approx f_{1,1} f_{2,1},\ f_{1,1} = f_{1,0} + h_{1,0}\ f_{2,1} = f_{2,0} + h_{2,0}, \tag{B.2}$$

where the polynomials $h_{1,0}$ and $h_{2,0}$ satisfy

$$p - f_{1,0} f_{2,0} = f_{1,0} h_{2,0} + h_{1,0} f_{2,0},\ \deg(h_{i,0}) < \deg(f_{i,0}) \text{ for } i = 1, 2. \tag{B.3}$$

This is Newton's iteration. Indeed, substitute (B.2) into exact splitting $p = f_{1,1} f_{2,1}$ and arrive at (B.3) up to a single term $h_{1,0} h_{2,0}$ of higher order. We can complete this iteration by computing polynomials $h_{1,0}$ and $h_{2,0}$ and then can continue such iterations recursively.

From equation (B.3) we obtain that $p = h_{1,0} f_{2,0} \mod f_{1,0}$, and so $h_{1,0} = p\bar{h}_1 \mod f_{1,0}$ where the multiplicative inverse $\bar{h}_1$ satisfies $\bar{h}_1 f_{2,0} \mod f_{1,0} = 1$. Having computed polynomials $\bar{h}_1$ and $h_{1,0}$ we can obtain the polynomial $h_{2,0}$ from equation (B.3) within a dominated cost bound by means of approximate polynomial division (cf. [121, 9, 105, 106, 60]).

B.2 Fast initialization of Newton's refinement

Next we recall Schönhage's algorithm of [120, Secs. 10 – 12] for Newton's refinement of a close initial approximate splitting (B.1) (see its extension and an alternative in Sec. 23.6).

Given an initial approximation $\bar{h}_{1,0}$ to $\bar{h}_1$ the algorithm recursively updates it[47] by computing the polynomials

$$j_{i,0} = 1 - \bar{h}_{i,0} f_{2,0} \mod f_{1,0} \text{ and } \bar{h}_{i+1,0} = \bar{h}_{i,0}(1 + j_{i,0}) \mod f_{1,0} \text{ for } i = 1, 2, \ldots$$

[47] We can compute the multiplicative inverse $\bar{h}_1$ by applying the extended Euclidean algorithm (cf., e.g., [89]), but it is not reliable in numerical implementation and produces an extraneous factor of $\log^2(d)$ in the Boolean complexity bound. Such an alternative computation of the multiplicative inverse, however, does not require initialization and can be replaced by the equivalent task of solving a Sylvester linear system of equation, which has displacement structure of Toeplitz type. See [89, Chapter 5] and [136] for highly efficient solution of such linear systems.

For any $b > d$ the computations ensure the bound $|\bar{h}_{i,0}| \le 2^{-b}$ by using $O(d\mu(b+d))$ Boolean operations provided that

$$m^2 2^{2d}\ |\bar{h}_{1,0}| \le \nu^2 \tag{B.4}$$

where $m = \deg(f_1) \le d$, $\nu = \min_{x:\ |x|=1} |p(x)|$ (see [120, Lemma 11.1]), and $\frac{1}{\nu} \le 2^{cn}$ for a constant c (see [120, equation (16.7)]). If in addition

$$w^4 2^{3d+m+1}\ |p - f_{1,0}f_{2,0}| \le \nu^4, \tag{B.5}$$

then the new factors $f_{1,1}$ and $f_{2,1}$ can be computed by using $O(d\mu(b+d))$ Boolean operations such that $|p - f_{1,1}f_{2,1}| \le |p|\ 2^{-b}$.

One can ensure bound (B.5) by combining Cauchy sum computation and Alg. 21.3 or 21.4 (see Sec. 21.5). This completes the computation of splitting at the overall Boolean time in $O(d\mu(b+d))$ provided that within this cost bound one can also compute an initial polynomial $\bar{h}_{1,0}$ satisfying (B.4).

One can do this based on the following expression of [120, equation (12.19)]:

$$\bar{h}_{1,0} = \sum_{i=0}^{m-1} \Big(\sum_{j=i+1}^{m} u_{m-j} v_{j-i} \Big) x^i$$

where u_k and v_k are the coefficients of the polynomial $f_{1,0}(x) = x^m + u_1 x^{m-1} + \cdots + u_m$ and of the Laurent expansion $\frac{1}{f_{1,0}(x)f_{2,0}(x)} = \sum_k v_k x^k$, respectively.

References

[1] Ahlfors, L.: Complex Analysis. McGraw-Hill series in Mathematics, McGraw-Hill (2000), ISBN 0-07-000657-1

[2] Abbott, J.: Quadratic interval refinement for real roots.
arXiv:1203.1227, Submitted on 6 Mar 2012.
Originally presented as a poster at ISSAC 2006

[3] Aberth, O.: Iteration methods for finding all zeros of a polynomial simultaneously. Mathematics of Computation **27** (122), 339–344 (1973)

[4] Atanassova, L., Kyurkchiev, N., Yamamoto, T.: Methods for computing all roots of a polynomial simultaneously - known results and open Problems. Computing [Suppl] **16**, 23–35 (2002)

[5] Aurentz, J.L., Mach, T., Vandebril, R., Watkins D.S.: Fast and backward stable computation of roots of polynomials. SIAM Journal on Matrix Analysis and Applications, 36(3): 942–973 (2015). Part II in arXiv:1611.02435, July 2018

[6] Alman, J., Williams, V. V.: A refined laser method and faster matrix multiplication. In ACM-SIAM SODA 2021, 522–539. SIAM (2021)

[7] Bell, E.T.: The Development of Mathematics, McGraw-Hill, New York (1940)

[8] Barnett, S.: Polynomial and Linear Control Systems. Marcel Dekker, New York (1983) doi: 10.1112/blms/16.3.331

[9] Bini, D.A.: Parallel solution of certain Toeplitz Linear systems. SIAM J. Comput. **13** (2), 268–279 (1984) doi: 10.1137/0213019

[10] Bini, D.A.: Numerical computation of polynomial zeros by means of Aberth's method. Numerical Algorithms **13** (3–4), 179–200 (1996)

[11] Bini, D.A.: private communication (2019)

[12] Bini, D.A., Boito, P., Eidelman, Y., Gemignani, L., Gohberg, I.: A fast implicit QR eigenvalue algorithm for companion matrices. Linear Algebra and its Applications, 432 (2010), 2006-2031

[13] Blum, Lenore; Cucker, Felipe; Shub, Mike; Smale, Steve: Complexity and Real Computation. Springer New York (1998). doi:10.1007/978-1-4612-0701-6. ISBN 978-0-387-98281-6. S2CID 12510680

[14] Bini, D.A., Fiorentino, G.: Design, analysis, and implementation of a multiprecision polynomial rootfinder. Numerical Algorithms **23**, 127–173 (2000) doi: 10.1023/A:1019199917103

[15] 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) doi: 10.1016/S0898-1221(04)90037-5

[16] Bini, D.A., Gemignani, L., Pan, V.Y.: Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations. Numer. Math. 100, 373–408 (2005). https://doi.org/10.1007/s00211-005-0595-4

[17] Box, G.E.P., Jenkins, G.M.: Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco, California, fifth edition (2015) doi: 10.1111/jtsa.12194

[18] Borodin, A., Munro, I.: The Computational Complexity of Algebraic and Numeric Problems, Elsevier computer science library (1975)

[19] Bini, D.A., Noferini, V.: Solving polynomial eigenvalue problems by means of the Ehrlich-Aberth method. Linear Algebra and its Applications, **439**(4), 1130-1149 (2013) DOI:10.1016/J.LAA.2013.02.024

[20] Bini, D.A., Pan, V.: Polynomial and Matrix Computations, **1**, Fundamental Algorithms (XVI + 415 pages), Birkhäuser, Boston (1994)

[21] Bini, D.A., Robol, L.: Solving secular and polynomial equations: a multiprecision algorithm. J. Comput. Appl. Math. **272**, 276–292 (2014) doi: 10.1016/j.cam.2013.04.037

[22] Becker, R., Sagraloff, M., Sharma, V., Xu, J., Yap, C.: Complexity analysis of root clustering for a complex polynomial. In: Intern. Symp. Symb. Algebraic Computation (ISSAC 2016), pp. 71–78. ACM Press, New York (2016) doi: 10.1145/2930889.2930939

[23] 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. Symbolic Computation **86**, 51–96 (2018) doi: 10.1016/j.jsc.2017.03.009

[24] Cramér, H.: Mathematical Methods of Statistics. Princeton Mathematical Series, 8, pp. xvi+575 (Princeton, N.J.: Princeton University Press, 1946)

[25] Cipra, B. A.: The best of the 20th century: Editors name top 10 algorithms. SIAM News **33, 4**, page 1 (May 2000).

[26] Carrier, J., Greengard, L., Rokhlin, V.: A fast adaptive multipole algorithm for particle simulations. SIAM Journal on Scientific and Statistical Computing **9**, 669-686 (1988)

[27] Cucker, F., Smale, S.: Complexity estimates depending on condition and round-off error. J. ACM, 46(1):113–184 (1999)

[28] Durand, E.: Solutions numériques des équations algébriques. Tome 1: Equations du type $F(X) = 0$; Racines d'un polynome. Masson, Paris (1960)

[29] Dutt, A., Gu, M., Rokhlin, V.: Fast Algorithms for polynomial interpolation, integration and differentiation. SIAM J. Numer. Anal. **33(5)**, 1689–1711 (1996)

[30] Delves, L.M., Lyness, J.N.: A numerical method for locating the zeros of an analytic function. Math. Comput. **21**, 543–560 (1967) doi: 10.1090/S0025-5718-1967-0228165-4

[31] Demeure, C.J., Mullis, C.T.: The Euclid algorithm and the fast computation of cross-covariance and autocovariance sequences. IEEE Trans. Acoust, Speech, Signal Processing **37**, 545–552 (1989) doi: 10.1109/29.17535

[32] Demeure, C.J., Mullis, C.T.: A Newton–Raphson method for moving-average spectral factorization using the Euclid algorithm. IEEE Trans. Acoust, Speech, Signal Processing **38**, 1697–1709 (1990) doi: 10.1109/29.60101

[33] Duan, R., Wu, H., Zhou, R.: Faster Matrix Multiplication via Asymmetric Hashing, arXiv:2210.10173 (2022)

[34] Ehrlich, L.W.: A modified Newton method for polynomials. *Communs. ACM* **10**, 107–108 (1967) doi: 10.1145/363067.363115

[35] Edelman, A., Kostlan, E.: How many zeros of a random polynomial are real? *Bull. Amer. Math. Soc.* **32**, 1–37 (1995)

[36] Emiris, I.Z., Pan, V.Y., Tsigaridas, E.: Algebraic Algorithms. In: Tucker, A.B., Gonzales, T., Diaz-Herrera, J.L. (eds.) Computing Handbook (3rd edition), Vol. **I**: Computer Science and Software Engineering, Ch. 10, pp. 10-1 – 10-40. Taylor and Francis Group (2014)

[37] Fiduccia, C. Polynomial evaluation via the division algorithm the fast fourier transform revisited. In Proceedings of the Fourth Annual ACM Symposium on Theory of Computing, STOC '72, page 88–93, New York, NY, USA, 1972. ACM Press, NY (1972)

[38] Fortune, S.: An iterated eigenvalue algorithm for approximating roots of univariate polynomials. In: J. of Symbolic Computation, **33, 5**, 627–646 (2002) doi: 10.1006/jsco.2002.0526. Proc. version in Proc. Intern. Symp. on Symbolic and Algebraic Computation (ISSAC'01), 121–128, ACM Press, New York (2001)

[39] Fischer, M. J., Paterson, M. S.: String matching and other products. SIAM-AMS Proceedings **7**, 113–125 (1974)

[40] Gourdon, X.: Combinatoire, Algorithmique et Géométrie des Polynomesole Polytechnique. PhD Thesis, Paris 1996, 152 pages (in French).

[41] von zur Gathen, J., Gerhard, J., Modern Computer Algebra. Cambridge University Press, Cambridge, UK, 2003 (second edition), 2013 (third edition).

[42] Golub, G.H., Van Loan, C.F.: Matrix Computations. The Johns Hopkins University Press, Baltimore, Maryland (2013) (fourth edition).

[43] Go, Soo; Pan, V.Y., Soto, P.: Root-squaring for root-finding, preprint (2023)

[44] Greengard, L., Rokhlin, V.: A fast algorithm for particle simulation, *Journal of Computational Physics*, **73**, 325–348 (1987)

[45] Householder, A.S.: Dandelin, Lobachevskii, or Graeffe? *Amer. Math. Monthly* **66**, 464–466 (1959)

[46] Henrici, P.: Applied and Computational Complex Analysis. Vol. 1: Power Series, Integration, Conformal Mapping, Location of Zeros. Wiley, NY (1974)

[47] Hristov, V., Kyurkchiev, N., Iliev, A.: Global convergence properties of the SOR-Weierstrass method. In: Poc, of Numerical Methods and Applications - 7th International Conference, NMA 2010, Borovets, Bulgaria, August 20-24, 2010, pages $437-444$ (2010)

DOI: 10.1007/978-3-642-18466-6_52 Issn: 0302-9743

[48] van der Hoeven, J., Lecerf, G.: Fast multivariate multi-point evaluation revisited (2018) hal: 01848571

[49] van der Hoeven, J., Lecerf, G.: Ultimate complexity for numerical algorithms. *ACM Commun. Comput. Algebra* **54(1)**, $1-13$ (2020)

[50] Hubbard, J., Schleicher, D., Sutherland, S.: (2001) How to find all roots of complex polynomials by Newton's method. Invent. Math. **146**, 1–33 (2001)

[51] Imbach, R., Pan, V.Y.: New progress in univariate polynomial root-finding. In: Procs. of ACM-SIGSAM ISSAC 2020, pages $249-256$, July 20-23, 2020, Kalamata, Greece, ACM Press (2020). ACM ISBN 978-1-4503-7100-1/20/07

[52] Imbach, R., Pan, V.Y.: New practical advances in polynomial root clustering. LNCS **11989**, Ch. 11, pp. 1–16 (2020) In Book: Mathematical Aspects of Computer and Information Sciences (MACIS 2019), D. Salmanig et al (Eds.), Springer Nature Switzerland, DOI:10.1007/$978-3-030-43120-4$_11

[53] Imbach, R., Pan, V.Y.: Root radii and subdivision for polynomial root-finding. In: Computer Algebra in Scientific Computing (CASC'21), Springer Nature Switzerland AG 2021, F. Boulier et al. (Eds.): CASC 2021, LNCS 12865, pp. 1–21, 2021.

https://doi.org/10.1007/$978-3-030-85165-1$_9

Also arXiv:2102.10821 (22 Feb 2021)

[54] Imbach, R., Pan, V.Y.: Accelerated subdivision for clustering roots of polynomials given by evaluation oracles. In: CASC'22, Springer Nature Switzerland AG 2022, F. Boulier et al. (Eds.): LNCS **13366**, 143–164 (2022)

[55] Imbach, R., Pan, V.Y., Yap, C.: Implementation of a near-optimal complex root clustering algorithm. In: Proc. Int. Congress Math Software (ICMS 2018), Davenport, J.H., Kaers, M., Labahn, G., Urban, G. (eds.), LNCS **10931** 235–244, Springer, Cham (2018)

[56] Imbach, R., Pan, V.Y., Yap, C., Kotsireas, I.S., Zaderman, V.: Root-finding with implicit deflation. In: Proceedings of the 21st International Workshop on Computer Algebra in Scientific Computing (CASC'2019), (M. England, W. Koepf, T.M. Sadikov, W.M. Seiler, and E. V. Vorozhtsov, editors), Lecture Notes in Computer Science **11661**, 236–245, Springer, Nature Switzerland (2019) doi: 10.1007/978-3-030-26831- arXiv:1606.01396, submitted on 21 May (2019)

[57] Jindal, G., Sagraloff, M.: Efficiently computing real roots of sparse polynomials. In: Proc. of ISSAC 2017, pages 229-236, ACM Press, NY (2017) DOI: 10.1145/3087604.3087652 Also in arXiv: 1704.06979, 23 Apr 2017

[58] Kac, M.: On the average number of real roots of a random algebraic equation. *Bull. Amer. Math. Soc.* **49**, 314–320 and 938 (1943)

[59] Kerner, I. O.: Ein Gesamtschrittverfahren zur Berechung der Nullstellen von Polynomen, *Numerische Math.* **8**, 290–294 (1966)

[60] Kirrinnis, P.: Polynomial factorization and partial fraction decomposition by simultaneous Newton's iteration. Journal of Complexity **14**, 378–444 (1998)

[61] Knuth, D.E.: The Art of Computer Programming, volume 2, Addison-Wesley, 1981 (second edition), 1997 (third edition)

[62] Kobel, A., Rouillier, F., Sagraloff, M.: Computing real roots of real polynomials ... and now for real! In: Intern. Symp. Symb. Algebraic Computation (ISSAC 2016), pp. 301 – 310. ACM Press, New York (2016) doi: 10.1145/2930889.2930937

[63] Kim, M.-H., Sutherland, S.: Polynomial root-finding algorithms and branched covers. SIAM Journal of Computing **23**, 415–436 (1994)

[64] Karstadt, E., Schwartz, O.: Matrix multiplication, a little faster. *J. ACM* **67**(1), 1–31 (2020) doi:10.1145/3364504, MR 4061328, S2CID 211041916

[65] Kedlaya, K. S., Umans, C.: Fast polynomial factorization and modular composition. SIAM J. Comput. **40, 6**, 1767–1802 (2011)

[66] Lorentz, G. G.: Bernstein Polynomials, University of Toronto Press (1953)

[67] Luan, Q., Pan, V.Y., Kim, W., Zaderman, V.: Faster numerical univariate polynomial root-finding by means of subdivision iterations. In: CASC 2020, F. Boulier et al. (Eds.): LNCS 12291, pp. 431–446. Springer (2020)

[68] Louis, A., Vempala, S. S.: Accelerated Newton iteration: roots of black box polynomials and matrix eigenvalues. IEEE FOCS 2016, **1**, 732-740 (2016) DOI: 10.1109/FOCS.2016.83; also arXiv:1511.03186, January 2016

[69] Mignotte, M.: An inequality about factors of polynomials. Math. of Computation **28**, 1153–1157 (1974) doi: 10.2307/2005373

[70] McNamee, J.M.: Numerical Methods for Roots of Polynomials, Part I, XIX+354 pages. Elsevier (2007) ISBN: 044452729X; ISBN13: 9780444527295

[71] McNamee, J.M., Pan, V.Y.: Numerical Methods for Roots of Polynomials, Part II, XXI+728 pages. Elsevier (2013) ISBN: 9780444527301

[72] Merzbach, U.C., Boyer, C.B.: A History of Mathematics. Wiley, New York (2011), fifth edition, doi: 10.1177/027046769201200316

[73] Moenck, R., Borodin, A.: Fast modular transform via divison, *Proceedings of 13th Annual Symposium on Switching and Automata Theory*, 90–96, IEEE Computer Society Press, Washington, DC (1972)

[74] Malajovich, G., Zubelli, J.P.: On the geometry of Graeffe iteration. Journal of Complexity **17(3)**, 541–573 (2001)

[75] Neff, C.A., Reif, J.H.: An $o(n^{1+\epsilon})$ algorithm for the complex root problem. In: Proc. 35th Ann. IEEE Symp. on Foundations of Comput. Sci. (FOCS '94), pp. 540–547. IEEE Computer Society Press (1994) doi: 10.1109/SFCS.1994.365737

[76] Pan, V. Y.: On methods of computing the values of polynomials, Uspekhi Matematicheskikh Nauk **21, 1 (127)**, 103–134 (1966). (Transl. Russian Mathematical Surveys **21, 1 (127)**, 105–137 (1966).)

[77] Pan, V.,Y.: Trilinear aggregating with implicit canceling for a new acceleration of matrix multiplication, Computers and Mathematics (with Applications), **8**(1), 23–34 (1982)

[78] Pan, V. Y.: How to Multiply Matrices Faster. Lecture Notes in Computer Science, vol. 179 (XI + 212 pages), Springer, Berlin (1984) ISBN:3-387-13866-8

[79] Pan, V. Y.: Sequential and parallel complexity of approximate evaluation of polynomial zeros. Computers and Mathematics (with Applications), **14, 8,** 591–622 (1987)

[80] Pan, V. Y.: Fast and efficient parallel evaluation of the zeros of a polynomial having only real zeros. Computers and Mathematics (with Applications), **17, 11**, 1475–1480 (1989)

[81] V. Y. Pan, New resultant inequalities and complex polynomial factorization. *SIAM Journal on Computing*, **23, 5**, 934–950, 1994.

[82] V. Y. Pan, Deterministic improvement of complex Polynomial factorization based on the properties of the associated resultant. *Computers and Math. (with Applications)*, **30, 2**, 71–94, 1995.

[83] Pan, V.Y.: Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros. ACM STOC'95, 741–750 (1995)

[84] Pan, V.Y.: Optimal and nearly optimal algorithms for approximating polynomial zeros. Computers and Mathematics (with Applications), bf 31, 12, 97–138 (1996).

[85] V. Y. Pan, Computing $x^n \mod p(x)$ and an application to splitting a polynomial into factors over a fixed disc. *Journal of Symbolic Computations*, **22**, 377–380, 1996.

[86] Pan, V.Y.: Solving a polynomial equation: some History and recent progress, SIAM Review, **39, 2**, 187–220 (1997)

[87] Pan, V.Y.: Solving polynomials with cComputers, American Scientist, **86**, 62-69 (January-February 1998)

[88] Pan, V.Y.: Approximation of complex polynomial zeros: modified quadtree (Weyl's) construction and improved Newton's iteration. J. Complexity **16**(1), 213–264 (2000) doi: 10.1006/jcom.1999

[89] Pan, V.Y.: Structured Matrices and Polynomials: Unified Superfast Algorithms. Birkhäuser/Springer, Boston/New York (2001) doi: 10.1007/978-1-4612-0129-8

[90] Pan, V.Y.: Univariate polynomials: nearly optimal algorithms for factorization and rootfinding. J. Symb. Comput. **33**(5), 701–733 (2002) doi: 10.1006/jsco.2002.0531. Procs. version in ACM ISSAC'01, 253–267 (2001)

[91] V. Y. Pan, Amended DSeSC Power method for polynomial root-finding. *Computers and Math. (with Applications)*, **49**, **9–10**, 1515–1524, 2005.

[92] Pan, V.Y.: Transformations of matrix structures work again, Linear Algebra and Its Applications, **465**, 1–32 (2015)

[93] Pan, V.Y.: Fast approximate computations with Cauchy matrices and polynomials, Mathematics of Computation, **86**, 2799–2826 (2017)
DOI: https://doi.org/10.1090/mcom/3204

[94] Pan, V.Y.: Fast matrix multiplication and its algebraic neighborhood, SB MATH (Mathematical Sbornik), **208 (11)**, 1161–1704 (in English), (2017) DOI:10.1070/SM8833

[95] Pan, V.Y.: Simple and nearly optimal polynomial root-finding by means of root radii approximation. In: Kotsireas, I.S., Martinez-Moro, E. (eds.) Springer Proc. in Math. and Statistics, Ch. 23: Applications of Computer Algebra **198** AG. Springer International Publishing (2017). Chapter doi: 10.1007/978-3-319-56932-1 23

[96] Pan, V.Y.: Fast feasible and unfeasible matrix multiplication, arxiv 1804.04102 (11 Apr 2018)

[97] Pan, V.Y.: Old and new nearly optimal polynomial root-finding. In: Proceedings of the 21st International Workshop on Computer Algebra in Scientific Computing (CASC'2019), (M. England, W. Koepf, T.M. Sadikov, W.M. Seiler, and E. V. Vorozhtsov, editors), Lecture Notes in Computer Science **11661**, 393–411, Springer, Nature Switzerland (2019) doi: 10.1007/978-3-030-26831-2

[98] Pan, V.Y.: Acceleration of subdivision root-finding for sparse polynomials. In: Computer Algebra in Scientific Computing, Springer Nature Switzerland AG 2020, F. Boulier et al. (Eds.): CASC 2020, LNCS 12291, pp. 1 – 17 (2020)

[99] Pan, V.Y.: New progress in polynomial root-finding. arXiv 1805.12042,
submitted on May 30, 2018, last revised on August 31, 2022.

[100] Pan, V.Y.: New progress in classic area: polynomial root-squaring and root-finding, arXiv 2206.01727 (2022)

[101] Pan, V.Y.; Go, Soo; Luan, Qi; Zhao, Liang: Fast approximation of polynomial zeros and matrix eigenvalues, Procs. of 13th International Symposium on Algorithms and Complexity (CIAC 2023), M. Mavronicolas (Ed.), Lecture Notes in Computer Science, 13898, pp. 1–17 Springer Nature, Switzerland AG (2023), arXiv 2301.11268 (submitted 2022, revised 2023).

[102] Pan, V.Y., Ivolgin, D., Murphy, B., Rosholt, R.E., Tang, Y., Wang, X., Yan, X.: Root-finding with eigen-solving. Chapter 14, pages 219–245 in *Symbolic-Numeric Computation*, (Dongming Wang and Lihong Zhi, editors), Birkhäuser, Basel/Boston, (2007)

[103] Pan, V.Y., Qian, G., Zheng, A.: Real and complex polynomial root-finding via eigen-solving and randomization. *Proceedings of CASC 2012* (V. P. Gerdt et al. editors), *Lecture Notes in Computer Science*, **7442**, 283–293, Springer, Heidelberg (2012)

[104] Pan, V.Y., Qian, G., Zheng, A., Chen, Z.: Matrix computations and polynomial root-finding with preprocessing. Linear Algebra and Its Applications, 434, 854–879 (2011)

[105] Pan, V.Y., Sadikou, A., Landowne, E.: Univariate polynomial division with a remainder by means of evaluation and interpolation. In: Proc. 3rd IEEE Symposium on Parallel and Distributed Processing, pp. 212–217. IEEE Computer Society Press, Los Alamitos, California (1991) doi: 10.1109/SPDP.1991.218277

[106] 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) doi: 10.1016/0020-0190(92)90055-Z

[107] Pan, V.Y., Svadlenka, J.: On the superfast Multipole method. arXiv: 1901.00377, 2019.

[108] Pan, V.Y., Tsigaridas, E.P.: Nearly optimal refinement of real roots of a univariate polynomial. J. Symb. Comput. **74**, 181–204 (2016) doi: 10.1016/j.jsc.2015.06.009
Proceedings version in: Kauers, M. (ed.) Proc. ISSAC 2013, pp. 299–306. ACM Press, New York (2013)

[109] Pan, V.Y., Tsigaridas, E.P.: Nearly optimal computations with structured matrices. Theor. Comput. Sci. **681**, 117–137 (2017)
doi: 10.1016/j.tcs.2017.03.030.

[110] Pan, V.Y., Zheng, A.: Real and complex polynomial root-finding with eigen-solving and preprocessing. In *Proc. International Symp. on Symbolic and Algebraic Computation (ISSAC 2010)*, pages 219–226, ACM Press, New York (2010)

[111] Pan, V.Y., Zheng, A.: New progress in real and complex polynomial root-finding. *Computers and Math. (with Applications)*, **61**, 1305–1334, 2011.

[112] Pan, V.Y., Zhao, L.: Real root isolation by means of root radii approximation. In: Proceedings of the 17th International Workshop on Computer Algebra in Scientific Computing (CASC'2015), (V. P. Gerdt, W. Koepf, and E. V. Vorozhtsov, editors), Lecture Notes in Computer Science **9301**, 347–358, Springer, Heidelberg, (2015) doi: 10.1007/978-3-319-24021-3 26

[113] Pan, V.Y., Zhao, L.: Real polynomial root-finding by means of matrix and polynomial iterations. Theor. Comput. Sci. **681**, 101–116 (2017) doi: 10.1016/j.tcs.2017.03.032.

[114] Renegar, J.: On the worst-case arithmetic complexity of approximating zeros of polynomials. J. Complex. **3**(2), 90–113 (1987)

[115] Reinke, B.: Diverging orbits for the Ehrlich-Aberth and the Weierstrass root finders. arXiv 2011.01660 (November 20, 2020)

[116] Randig, M., Schleicher, D., Stoll, R.: Newton's method in practice II: The iterated refinement Newton method and near-optimal complexity for finding all roots of some polynomials of very large degrees. arXiv:1703.05847 (31 Dec 2017)

[117] Reinke, B., Schleicher, D., Stoll, M.: The Weierstrass root-finder is not generally convergent, arXiv:2004.04777 (9 Apr 2020)

[118] Schröder, E.: Uber unendliche viele Algoritmen zur Auflosung der Gleichungen. Math. Ann. **2**, 317–365 (1870)

[119] Smale, S.: The fundamental theorem of algebra and complexity theory, Bulletin of the American Math. Society, **4 (1)**, 1–36 (1981)

[120] Schönhage, A.: The fundamental theorem of algebra in terms of computational complexity. Math. Dept., University of Tübingen, Tübingen, Germany (1982)

[121] Schönhage, A.: Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients. In: Proc. European Computer Algebra Conference (EUROCAM 1982), pp. 3–15. Computer Algebra (1982) doi: 10.1007/3-540-11607-9 1

[122] Schönhage, A.: Quasi-gcd computations. Journal of Complexity **1**, 118–137 (1985)

[123] Storjohann, A.: The shifted number system for fast linear algebra on integer matrices, Journal of Complexity **21**(4), 609–650 (2005)

[124] Stewart, G. W.: Matrix Algorithms, Vol. II: Eigensystems, SIAM (1998)

[125] Strassen, V.: Some results in a complexity theory, Proceedings of the International Congress of Mathematicians, Vancouver, 1974 (Ralph D. James, editor), Volume 1, Canadian Math. Society, pages 497–501 (1974)

[126] Schleicher, D.: private communication (2018).

[127] Schönhage, A., Strassen, V.: Schnelle Multiplikation grosser Zahlen. Computing **7**, 281–292 (1971)

[128] 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) doi: 10.1016/j.tcs.2017.03.025; arXiv:1508.02935

[129] Toom, A. L.: The complexity of a scheme of functional elements realizing the multiplication of integers. Soviet Mathematics Doklady **3**, 714–716 (1963)

[130] Tilli, P.: Convergence conditions of some methods for the simultaneous computation of polynomial zeros. In: Calcolo **35**, 3–15 (1998) doi: 10.1007/s100920050005

[131] Trefethen, Lloyd N., Weideman, J.A.C., The exponentially convergent trapezoidal rule, SIAM Review, 385–458 **56(3)**, (2014) DOI: 10.1137/130932132

[132] Van Dooren, P.M.: Some numerical challenges in control theory. In: Linear Algebra for Control Theory, IMA Vol. Math. Appl. **62**, 177 – 189 (1994) doi: 10.1007/978-1-4613-8419-9 12

[133] Weierstrass, K.: Neuer Beweis des Fundamentalsatzes der Algebra. Mathematische Werke, Bd. **III**, 251–269. Mayer und Mueller, Berlin (1903)

[134] Weyl, H.: Randbemerkungen zu Hauptproblemen der Mathematik. II. Fundamentalsatz der Algebra und Grundlagen der Mathematik. Mathematische Zeitschrift **20**, 131–151 (1924)

[135] Wilson, G.T.: Factorization of the covariance generating function of a pure moving-average. SIAM J. Numer. Anal. **6**, 1–7 (1969) doi: 10.1137/0706001

[136] Xia, J., Xi, Y., Gu, M.: A superfast structured solver for Toeplitz linear systems via randomized sampling, SIAM J. Matrix Anal. Appl. **33**, 837–858 (2012) doi:10.1137/110831982

[137] Yokota, R., Barba, L.A.: A tuned and scalable fast multipole method as a preeminent algorithm for exascale system, The International Journal of High Performance Computing Applications, **26, 4**, 337–346 (2012)