Accelerated Subdivision for Clustering Roots of Polynomials Given by Evaluation Oracles

Rémi Imbach[1] and Victor Y. Pan[2(✉)]

[1] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
remi.imbach@laposte.net
[2] Lehman College and Graduate Center of City University of New York, New York, USA
victor.pan@lehman.cuny.edu

Abstract. In our quest for the design, the analysis and the implementation of a subdivision algorithm for finding the complex roots of univariate polynomials given by oracles for their evaluation, we present subalgorithms allowing substantial acceleration of subdivision for complex roots clustering for such polynomials. We rely on approximation of the power sums of the roots in a fixed complex disc by Cauchy sums, each computed in a small number of evaluations of an input polynomial and its derivative, that is, in a polylogarithmic number in the degree. We describe root exclusion, root counting, root radius approximation and a procedure for contracting a disc towards the cluster of root it contains, called ε-compression. To demonstrate the efficiency of our algorithms, we combine them in a prototype root clustering algorithm. For computing clusters of roots of polynomials that can be evaluated fast, our implementation competes advantageously with user's choice for root finding, `MPsolve`.

Keywords: Polynomial root finding · Subdivision algorithms · Oracle polynomials

1 Introduction

We consider the

ε-Complex Root Clustering Problem (ε-CRC)
Given: a polynomial $p \in \mathbb{C}[z]$ of degree d, $\varepsilon > 0$
Output: $\ell \leq d$ couples $(\Delta^1, m^1), \ldots, (\Delta^\ell, m^\ell)$ satisfying:
- the Δ^j's are pairwise disjoint discs of radii $\leq \varepsilon$,
- for any $1 \leq j \leq \ell$, Δ^j and $3\Delta^j$ contain $m^j > 0$ roots of p,
- each complex root of p is in a Δ^j for some j.

Victor's research has been supported by NSF Grant CCF 1563942 and PSC CUNY Award 63677 00 51.

F. Boulier et al. (Eds.): CASC 2022, LNCS 13366, pp. 143–164, 2022.
https://doi.org/10.1007/978-3-031-14788-3_9

Here and hereafter *root(s)* stands for *root(s) of* p and are counted with multiplicities, $3\Delta^j$ for the factor 3 concentric dilation of Δ^j, and p is a *Black box polynomial*: its coefficients are not known, but we are given *evaluation oracles*, that is, procedures for the evaluation of p, its derivative p' and hence the ratio p'/p at a point $c \in \mathbb{C}$ with a fixed precision. Such a black box polynomial can come from an experimental process or can be defined by a procedure, for example Mandelbrot's polynomials, defined inductively as

$$\mathtt{Man}_1(z) = z, \quad \mathtt{Man}_k(z) = z\,\mathtt{Man}_{k-1}(z)^2 + 1.$$

$\mathtt{Man}_k(z)$ has degree $d = 2^k - 1$ and d non-zero coefficients but can be evaluated fast, i.e., in $O(k)$ arithmetic operations. Any polynomial given by its coefficients can be handled as a black box polynomial, and the evaluation subroutines for p, p' and p'/p are fast if p is sparse or Mandelbrot-like. One can solve root-finding problems and in particular the ε-CRC problem for black box polynomials by first retrieving the coefficients by means of evaluation-interpolation, e.g., with FFT and inverse FFT, and then by applying the algorithms of [2,4,11,13,19]. Evaluation-interpolation, however, decompresses the representation of a polynomial, which can blow up its input length, in particular, can destroy sparsity. We do not require knowledge of the coefficients of an input polynomial, but instead use evaluation oracles.

Functional root-finding iterations such as Newton's, Weierstrass's (also known as Durand-Kerner's) and Ehrlich's iterations – implemented in `MPsolve` [4] – can be applied to approximate the roots of black box polynomials. Applying such iterations, however, requires initial points, which the known algorithms and in particular `MPsolve` obtain by computing root radii, and for that it needs the coefficients of the input polynomial.

Subdivision Algorithms. Let $\mathbf{i}$ stand for $\sqrt{-1}$, $c \in \mathbb{C}$, $c = a + \mathbf{i}b$ and $r, w \in \mathbb{R}$, r and w positive. We call *box* a square complex interval of the form $B(c,w) := [a - \frac{w}{2}, a + \frac{w}{2}] + \mathbf{i}[b - \frac{w}{2}, b + \frac{w}{2}]$ and *disc* $D(c,r)$ the set $\{x \in \mathbb{C} \mid |x - c| \le r\}$. The *containing disc* $D(B(c,w))$ of a box $B(c,w)$ is $D(c,(3/4)w)$. For a $\delta > 0$ and a box or a disc S, δS denotes factor δ concentric dilation of S.

We consider algorithms based on iterative subdivision of an initial box B_0 (see [2,3,12]) and adopt the framework of [2,3] which relies on two basic subroutines: an *Exclusion Test* (ET) – deciding that a small inflation of a disc contains no root – and a *Root Counter* (RC) – counting the number of roots in a small inflation of a disc. A box B of the subdivision tree is tested for root exclusion or inclusion by applying the ET and RC to $D(B)$, which can fail and return -1 when $D(B)$ has some roots near its boundary circle. In [2], ET and RC are based on the Pellet's theorem, requiring the knowlege of the coefficients of p and shifting the center of considered disc into the origin (*Taylor's shifts*); then Dandelin-Lobachevsky-Gräffe iterations, aka *root-squaring* iterations, enable the following properties for boxes B and discs Δ:

(p1) if $2B$ contains no root, ET applied to $D(B)$ returns 0,

(p2) if Δ and 4Δ contain m roots, RC applied to 2Δ returns m.

(p1) and (p2) bound the depth of the subdivision tree. To achieve quadratic convergence to clusters of roots, [2] uses a complex version of the Quadratic Interval Refinement iterations of J. Abbott [1], aka QIR Abbott iterations, described in details in Algorithm 7 of [3] and, like [12], based on extension of Newton's iterations to multiple roots due to Schröder. [8] presents an implementation of [2] in the `C` library `Ccluster`[1], which slightly outperforms `MPsolve` for initial boxes containing only few roots.

In [6] we applied an ET based on Cauchy sums approximation. It satisfies (p1) and instead of coefficients of p involves $O(\log^2 d)$ evaluations of p'/p with precision $O(d)$ for a disc with radius in $O(1)$; although the output of this ET is only certified if no roots lie on or near the boundary of the input discs, in our extensive experiments it was correct when we dropped this condition.

1.1 Our Contributions

The ultimate goal of our work is to design an algorithm for solving the ε-CRC problem for black box polynomials which would run faster in practice than the known solvers, have low and possibly near optimal Boolean complexity (aka bit complexity). We do not achieve this yet in this paper but rather account for the advances along this path by presenting several sub-routines for root clustering. We implemented and assembled them in an experimental ε-CRC algorithm which outperforms the user's choice software for complex root finding, `MPsolve`, for input polynomials that can be evaluated fast.

Cauchy ET and RC. We describe and analyze a new RC based on Cauchy sum computations and satisfying property (p2) which only require the knowledge of evaluation oracles. For input disc of radius in $O(1)$, it requires evaluation of p'/p at $O(\log^2 d)$ points with precision $O(d)$ and is based on our ET presented in [6]; the support for its correctness is only heuristic.

Disc Compression. For a set S, let us write $Z(S,p)$ for the set of roots in S and $\#(S,p)$ for the cardinality of $Z(S,p)$; two discs Δ and Δ' are said *equivalent* if $Z(\Delta,p) = Z(\Delta',p)$. We introduce a new sub-problem of ε-CRC:

ε-Compression into Rigid Disc (ε-CRD)
Given: a polynomial $p \in \mathbb{C}[z]$ of degree d, $\varepsilon > 0$, $0 < \gamma < 1$,
a disc Δ s.t. $Z(\Delta,p) \neq \emptyset$ and 4Δ is equivalent to Δ.
Output: a disc $\Delta' \subseteq \Delta$ of radius r' s.t. Δ' is equivalent to Δ and:
- either $r' \leq \varepsilon$,
- or $\#(\Delta,p) \geq 2$ and Δ' is at least γ-*rigid*, that is

$$\max_{\alpha,\alpha' \in Z(\Delta',p)} \frac{|\alpha - \alpha'|}{2r'} \geq \gamma.$$

[1] https://github.com/rimbach/Ccluster.

The ε-CRD problem can be solved with subdivision and QIR Abbott iteration, but this may require, for an initial disk of radius r, up to $O(\log(r/\max(\varepsilon', \varepsilon)))$ calls to the ET in the subdivision if the radius of convergence of the cluster in Δ for Schröder's iteration is in $O(\varepsilon')$.

Table 1. Runs of `CauchyQIR`, `CauchyComp` and `MPsolve` on Mignotte and Mandelbrot polynomials

		`CauchyQIR`			`CauchyComp`			`MPsolve`
d	$\log_{10}(\varepsilon^{-1})$	t	n	t_N	t	n	t_C	t
Mignotte polynomials, $a = 16$								
1024	5	1.68	30850	0.44	**0.96**	16106	0.27	1.04
1024	10	2.08	30850	0.58	**1.07**	16106	0.37	1.30
1024	50	**2.17**	30850	0.71	2.70	16105	1.96	4.84
2048	5	3.84	62220	0.90	**2.13**	32148	0.51	4.08
2048	10	4.02	62220	1.03	**2.36**	32148	0.70	5.09
2048	50	**4.51**	62220	1.25	5.62	32147	3.78	17.1
Mandelbrot polynomials								
1023	5	10.4	30877	0.86	**6.23**	18701	0.41	27.2
1023	10	10.1	30920	0.91	**6.45**	18750	0.59	30.0
1023	50	10.3	30920	1.06	**8.64**	18713	2.71	45.7
2047	5	24.3	62511	1.95	**15.2**	39296	1.39	229.
2047	10	26.4	62952	2.31	**15.5**	39358	1.71	246.
2047	50	26.1	62952	2.64	**20.4**	39255	6.22	380.

We present and analyze an algorithm solving the ε-CRD problem for $\gamma = 1/8$ based on Cauchy sums approximation and on an algorithm solving the following root radius problem: for a given $c \in \mathbb{C}$, a given non-negative integer $m \leq d$ and a $\nu > 1$, find r such that $r_m(c,p) \leq r \leq \nu r_m(c,p)$ where $r_m(c,p)$ is the smallest radius of a disc centered in c and containing exactly m roots of p. Our compression algorithm requires only $O(\log\log(r/\varepsilon))$ calls to our RC, but a number of evaluations and arithmetic operations increasing linearly with $\log(1/\varepsilon)$.

Experimental Results. We implemented our algorithms[2] within `Ccluster` and assembled them in two algorithms named `CauchyQIR` and `CauchyComp` for solving the ε-CRC problem for black box polynomials. Both implement the subdivision process of [2] with our heuristically correct ET and RC. `CauchyQIR` uses QIR Abbott iterations of [3] (with Pellet's test replaced by our RC), while `CauchyComp` uses our compression algorithm instead of QIR Abbott iterations.

[2] they are not publicly realeased yet.

We compare runs of `CauchyQIR` and `CauchyComp` to emphasize the practical improvements allowed by using compression in subdivision algorithms for root finding. We also compare running times of `CauchyComp` and `MPsolve` to demonstrate that subdivision root finding can outperform solvers based on functional iterations for polynomials that can be evaluated fast. `MPsolve` does not cluster roots of a polynomial, but approximate each root up to a given error ε. Below we used the latest version[3] of `MPsolve` and call it with: `mpsolve -as -Ga -j1 -oN` where `N` stands for $\max(1, \lceil \log_{10}(1/\varepsilon) \rceil)$.

All the timings given below have to be understood as sequential running times on a `Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz` machine with Linux. We highlight with boldface the best running time for each example. We present in Table 1 results obtained for Mandelbrot and Mignotte polynomials of increasing degree d for decreasing error ε. The Mignotte polynomial of degree d and parameter a is defined as

$$\mathtt{Mig}_{d,a}(z) = z^d - 2(2^{\frac{a}{2}-1}z - 1)^2.$$

In Table 1, we account for the running time t for the three above-mentionned solvers. For `CauchyQIR` (resp. `CauchyComp`), we also give the number n of exclusion tests in the subdivision process, and the time t_N (resp. t_C) spent in QIR Abbott iterations (resp. compression). Mignotte polynomials have two roots with mutual distance close to the theoretical separation bound; with the ε used in Table 1, those roots are not separated.

1.2 Related Work

The subdivision root-finders of Weyl 1924, Henrici 1974, Renegar 1987, [3,12], rely on ET, RC and root radii sub-algorithms and heavily use the coefficients of p. Design and analysis of subdivision root-finders for a black box p have been continuing since 2018 in [16] (now over 150 pages), relying on the novel idea and techniques of compression of a disc and on novel ET, RC and root radii sub-algorithms, and partly presented in [5,6,10,14,15], and this paper. A basic tool of Cauchy sum computation was used in [20] for polynomial deflation, but in a large body of our results only Thm. 5 is from [20]; we deduced it in [5,16] from a new more general theorem of independent interest. Alternative derivation and analysis of subdivision in [16] (yielding a little stronger results but presently not included) relies on Schröder's iterations, extended from [12]. The algorithms are analyzed in [10,14–16], under the model for black box polynomial root-finding of [9]. [5,6] complement this study with some estimates for computational precision and Boolean complexity. We plan to complete them using much more space (cf. 46 pages in each of [20] and [3]).[4] Meanwhile we borrowed from [3]

[3] `3.2.1` available here: https://numpi.dm.unipi.it/software/mpsolve.

[4] In [20, Sect. 2], called "The result", we read: "The method is involved and many details still need to be worked out. In this report also many proofs will be omitted. A full account of the new results shall be given in a monograph" which has actually never appeared. [3] deduced *a posteriori estimates*, depending on root separation and Mahler's measure, that is, on the roots themselves, not known a priori.

Pellet's RC (involving coefficients), Abbott's QIR and the general subdivision algorithm with connected components of boxes extended from [12,18]. With our novel sub-algorithms, however, we significantly outperform `MPsolve` for polynomials that can be evaluated fast; all previous subdivision root-finders have never come close to such level. `MPsolve` relies on Ehrlich's (aka Aberth's) iterations, whose Boolean complexity is proved to be unbounded because iterations diverge for worst case inputs [17], but divergence never occurs in decades of extensive application of these iterations.

1.3 Structure of the Paper

In Sect. 2, we describe power sums and their approximation with Cauchy sums. In Sect. 3, we present and analyze our Cauchy ET and RC. Section 4 is devoted to root radii algorithms and Sect. 5 to the presentation of our algorithm solving the ε-CRD problem. We describe the experimental solvers `CauchyQIR` and `CauchyComp` in Sect. 6, numeric results in Sect. 7 and conclude in Sect. 8. We introduce additional definitions and properties in the rest of this section.

1.4 Definitions and Two Evaluations Bounds

Troughout this paper, log is the binary logarithm and for a positive real number a, let $\overline{\log} a = \max(1, \log a)$.

Annuli, Intervals. For $c \in \mathbb{C}$ and positives $r \leq r' \in \mathbb{R}$, the annulus $A(c, r, r')$ is the set $\{z \in \mathbb{C} \mid r' \leq |z - c| \leq r'\}$.

Let $\square\mathbb{R}$ be the set $\{[a - \frac{w}{2}, a + \frac{w}{2}] \mid a, w \in \mathbb{R}, w \geq 0\}$ of real intervals. For $\square a = [a - \frac{w}{2}, a + \frac{w}{2}] \in \square\mathbb{R}$ the center $\mathrm{c}(\square a)$, the width $\mathrm{w}(\square a)$ and the radius $\mathrm{r}(\square a)$ of $\square a$ are respectively a, w and $w/2$.

Let $\square\mathbb{C}$ be the set $\{\square a + \mathbf{i}\square b | \square a, \square b \in \square\mathbb{R}\}$ of complex intervals. If $\square c \in \square\mathbb{C}$, then $\mathrm{w}(\square c)$ (resp. $\mathrm{r}(\square c)$) is $\max(\mathrm{w}(\square a), \mathrm{w}(\square b))$ (resp $\mathrm{w}(\square c)/2$). The center $\mathrm{c}(\square c)$ of $\square c$ is $\mathrm{c}(\square a) + \mathbf{i}\mathrm{c}(\square b)$.

Isolation and Rigidity of a Disc are defined as follows [12,16].

Definition 1. (Isolation) *Let $\theta > 1$. The disc $\Delta = D(c, r)$ has* isolation θ *for a polynomial p or equivalently is* at least θ-isolated *if $Z\left(\frac{1}{\theta}\Delta, p\right) = Z(\theta\Delta, p)$, that is, $Z(A(c, r/\theta, r\theta), p) = \emptyset$.*

Definition 2. (Rigidity) *For a disc $\Delta = D(c, r)$, define*

$$\gamma(\Delta) = \max_{\alpha, \alpha' \in Z(\Delta, p)} \frac{|\alpha - \alpha'|}{2r}$$

and remark that $\gamma(\Delta) \leq 1$. We say that Δ has rigidity γ or equivalently is at least γ-rigid *if $\gamma(\Delta) \geq \gamma$.*

Oracle Numbers and Oracle Polynomials. Our algorithms deal with numbers that can be approximated arbitrarily closely by a Turing machine. We call such approximation automata *oracle numbers* and formalize them through interval arithmetic.

For $a \in \mathbb{C}$ we call *oracle* for a a function $\mathcal{O}_a : \mathbb{N} \to \square\mathbb{C}$ such that $a \in \mathcal{O}_a(L)$ and $\mathrm{r}(\mathcal{O}_a(L)) \leq 2^{-L}$ for any $L \in \mathbb{N}$. In particular, one has $|\mathrm{c}(\mathcal{O}_a(L)) - a| \leq 2^{-L}$. Let $\mathcal{O}_{\mathbb{C}}$ be the set of oracle numbers which can be computed with a Turing machine. For a polynomial $p \in \mathbb{C}[z]$, we call *evaluation oracle* for p a function $\mathcal{I}_p : (\mathcal{O}_{\mathbb{C}}, \mathbb{N}) \to \square\mathbb{C}$, such that if $\mathcal{O}_a$ is an oracle for a and $L \in \mathbb{N}$, then $p(a) \in \mathcal{I}_p(\mathcal{O}_a, L)$ and $\mathrm{r}(\mathcal{I}_p(\mathcal{O}_a, L)) \leq 2^{-L}$. In particular, one has $|\mathrm{c}(\mathcal{I}_p(\mathcal{O}_a, L)) - p(a)| \leq 2^{-L}$.

Consider evaluation oracles $\mathcal{I}_p$ and $\mathcal{I}_{p'}$ for p and p'. If p is given by $d' \leq d+1$ oracles for its coefficients, one can easily construct $\mathcal{I}_p$ and $\mathcal{I}_{p'}$ by using, for instance, Horner's rule. However for procedural polynomials (*e.g.* Mandelbrot), fast evaluation oracles $\mathcal{I}_p$ and $\mathcal{I}_{p'}$ are built from procedural definitions.

To simplify notations, we let $\mathcal{I}_p(a, L)$ stand for $\mathcal{I}_p(\mathcal{O}_a, L)$. In the rest of the paper, $\mathcal{P}$ (resp. $\mathcal{P}'$) is an evaluation oracle for p (resp. p'); $\mathcal{P}(a, L)$ (resp. $\mathcal{P}'(a, L)$) will stand for $\mathcal{I}_p(\mathcal{O}_a, L)$ (resp. $\mathcal{I}_{p'}(\mathcal{O}_a, L)$).

Two Evaluation Bounds. The lemma below provides estimates for values of $|p|$ and $|p'/p|$ on the boundary of isolated discs. See [7, Appendix A.1] for a proof.

Lemma 3. *Let $D(c, r)$ be at least θ-isolated, $z \in \mathbb{C}$, $|z| = 1$ and g be a positive integer. Let $\mathrm{lcf}(p)$ be the leading coefficient of p. Then*

$$|p(c + rz^g)| \geq |\mathrm{lcf}(p)| \frac{r^d(\theta - 1)^d}{\theta^d} \quad \text{and} \quad \left|\frac{p'(c + rz^g)}{p(c + rz^g)}\right| \leq \frac{d\theta}{r(\theta - 1)}.$$

2 Power Sums and Cauchy Sums

Definition 4. (Power sums of the roots in a disc) *The h-th power sum of (the roots of) p in the disc $D(c, r)$ is the complex number*

$$s_h(p, c, r) = \sum_{\alpha \in Z(\Delta, p)} \#(\alpha, p)\, \alpha^h, \tag{1}$$

where $\#(\alpha, p)$ stands for the multiplicity of α as a root of p.

The power sums $s_h(p, c, r)$ are equal to Cauchy's integrals over the boundary circle $\partial D(c, r)$; by following [20] they can be approximated by Cauchy sums obtained by means of the discretization of the integrals: let $q \geq 1$ be an integer and ζ be a primitive q-th root of unity. When $p(c + r\zeta^g) \neq 0$ for $g = 0, \ldots, q-1$, and in particular when $D(c, r)$ is at least θ-isolated with $\theta > 1$, define the Cauchy sum $\widetilde{s_h}^q(p, c, r)$ as

$$\widetilde{s_h}^q(p, c, r) = \frac{r}{q} \sum_{g=0}^{q-1} \zeta^{g(h+1)} \frac{p'(c + r\zeta^g)}{p(c + r\zeta^g)}. \tag{2}$$

For conciseness of notations, we write s_h for $s_h(p,0,1)$ and $\widetilde{s_h}^q$ for $\widetilde{s_h}^q(p,0,1)$. The following theorem, proved in [6,20], allows us to approximate power sums by Cauchy sums in $D(0,1)$.

Theorem 5. *For $\theta > 1$ and integers h, q s.t. $0 \leq h < q$ let the unit disc $D(0,1)$ be at least θ-isolated and contain m roots of p. Then*

$$|\widetilde{s_h}^q - s_h| \leq \frac{m\theta^{-h} + (d-m)\theta^h}{\theta^q - 1}. \tag{3}$$

$$\textit{Fix } e > 0.\ \textit{If } q \geq \lceil \log_\theta(\frac{d}{e}) \rceil + h + 1 \textit{ then } |\widetilde{s_h}^q - s_h| \leq e. \tag{4}$$

Remark that $s_0(p,c,r)$ is the number of roots of p in $D(c,r)$ and $s_1(p,c,r)/m$ is their center of gravity when $m = \#(D(c,r),p)$.

Next we extend Theorm 5 to the approximation of 0-th and 1-st power sums by Cauchy sums in any disc, and define and analyze our basic algorithm for the computation of these power sums.

2.1 Approximation of the Power Sums

Let $\Delta = D(c,r)$ and define $p_\Delta(z)$ as $p(c+rz)$ so that α is a root of p_Δ in $D(0,1)$ if and only if $c + r\alpha$ is a root of p in Δ. Following Newton's identities, one has:

$$s_0(p,c,r) = s_0(p_\Delta,0,1), \tag{5}$$
$$s_1(p,c,r) = cs_0(p_\Delta,0,1) + rs_1(p_\Delta,0,1). \tag{6}$$

Next since $p'_\Delta(z) = rp'(c+rz)$, one has

$$\widetilde{s_h}^q(p,c,r) = \frac{1}{q}\sum_{g=0}^{q-1} \zeta^{g(h+1)} \frac{p'_\Delta(\zeta^g)}{p_\Delta(\zeta^g)} = \widetilde{s_h}^q(p_\Delta,0,1)$$

and can easily prove:

Corollary 6. (of theorm 5) *Let $\Delta = D(c,r)$ be at least θ-isolated. Let $q > 1$, $s_0^* = \widetilde{s_0}^q(p,c,r)$ and $s_1^* = \widetilde{s_1}^q(p,c,r)$. Let $e > 0$. One has*

$$|s_0^* - s_0(p,c,r)| \leq \frac{d}{\theta^q - 1}. \tag{7}$$

$$\textit{If } q \geq \lceil \log_\theta(1 + \frac{d}{e}) \rceil \textit{ then } |s_0^* - s_0(p,c,r)| \leq e. \tag{8}$$

Let Δ contain m roots.

$$|mc + rs_1^* - s_1(p,c,r)| \leq \frac{rd\theta}{\theta^q - 1}. \tag{9}$$

$$\textit{If } q \geq \lceil \log_\theta(1 + \frac{r\theta d}{e}) \rceil \textit{ then } |mc + rs_1^* - s_1(p,c,r)| \leq e. \tag{10}$$

2.2 Computation of Cauchy Sums

Next we suppose that $D(c,r)$ and q are such that $p(c+r\zeta^g) \neq 0\ \forall 0 \leq g < q$, so that $\widetilde{s_h}^q(p,c,r)$ is well defined. We approximate Cauchy sums with evaluation oracles $\mathcal{P}$, $\mathcal{P}'$ by choosing a sufficiently large L and computing the complex interval:

$$\square\widetilde{s_h}^q(p,c,r,L) = \frac{r}{q}\sum_{g=0}^{q-1}\mathcal{O}_{\zeta^{g(h+1)}}(L)\frac{\mathcal{P}'(c+r\zeta^g,L)}{\mathcal{P}(c+r\zeta^g,L)}. \tag{11}$$

$\square\widetilde{s_h}^q(p,c,r,L)$ is well defined for $L > \max_{0\leq g<q}(-\log_2(p(c+r\zeta^g)))$ and contains $\widetilde{s_h}^q(p,c,r)$. The following result specifies L for which we obtain that $\mathrm{r}(\square\widetilde{s_h}^q(p,c,r,L)) \leq e$ for an $e > 0$. See [7, Appendix A.2] for a proof.

Lemma 7. *For strictly positive integer d, reals r and e and $\theta > 1$, let*

$$L(d,r,e,\theta) := \max\left((d+1)\log\frac{\theta}{er(\theta-1)} + \log(26rd), 1\right)$$
$$\in O\left(d\left(\overline{\log}\frac{1}{re} + \log\frac{\theta}{\theta-1}\right)\right).$$

If $L \geq L(d,r,e,\theta)$ then $\mathrm{r}(\square\widetilde{s_h}^q(p,c,r,L)) \leq e$.

In the sequel let $L(d,r)$ stand for $L(d,r,1/4,2)$.

2.3 Approximating the Power Sums $s_0, s_1, \ldots, s_h$

Our Algorithm 1 computes, for a given integer h, approximations to power sums $s_0, s_1, \ldots, s_h$ (of p_Δ in $D(0,1)$) up to an error e, based on Eqs. (2) and (4).

Algorithm 1 satisfies the following proposition. See [7, Appendix A.3] for a proof.

Proposition 8. *Algorithm 1 terminates for an $L \leq L(d,r,e/4,\theta)$. Let* **ApproxShs**$(\mathcal{P},\mathcal{P}',\Delta,\theta,h,e)$ *return* $(success, [\square s_0,\ldots,\square s_h])$. *Let $\Delta = D(c,r)$ and $p_\Delta(z) = p(c+rz)$. If $\theta > 1$, one has:*

(a) If $A(c,r/\theta,r\theta)$ contains no root of p, then success = true and for all $i \in \{0,\ldots,h\}$, $\mathrm{w}(\square s_i) < e$ and $\square s_i$ contains $s_i(p_\Delta,0,1)$.
(b) If $e \leq 1$ and $D(c,r\theta)$ contains no root of p then success = true and for all $i \in \{0,\ldots,h\}$, $\square s_i$ contains the unique integer 0.
(c) If $e \leq 1$ and $A(c,r/\theta,r\theta)$ contains no root of p, $\square s_0$ contains the unique integer $s_0(p,c,r) = \#(\Delta,p)$.
(d) If success = false, then $A(c,r/\theta,r\theta)$ and $D(c,r\theta)$ contain (at least) a root of p.
(e) If success = true and $\exists i \in \{0,\ldots,h\}$, s.t. $\square s_i$ does not contain 0 then $A(c,r/\theta,r\theta)$ and $D(c,r\theta)$ contains (at least) a root of p.

Algorithm 1. **ApproxShs**$(\mathcal{P}, \mathcal{P}', \Delta, \theta, h, e)$

Require: $\mathcal{P}, \mathcal{P}'$ evaluation oracles for p and p', s.t. p is monic of degree d. $\Delta = D(c, r)$, $\theta \in \mathbb{R}, \theta > 1$, $h \in \mathbb{N}$, $h \geq 0$, $e \in \mathbb{R}$, $e > 0$.
Ensure: a flag $success \in \{true, false\}$, a vector $[\square s_0, \ldots, \square s_h]$.
1: $e' \leftarrow e/4$, $q \leftarrow \lceil \log_\theta(4d/e) \rceil + h + 1$
2: $\ell \leftarrow \frac{r^d(\theta-1)^d}{\theta^d}$, $\ell' \leftarrow \frac{d\theta}{r(\theta-1)}$
3: $L \leftarrow 1$
4: $[\square s_0, \ldots, \square s_h] \leftarrow [\mathbb{C}, \ldots, \mathbb{C}]$
5: **while** $\exists i \in \{0, \ldots, h\}$ s.t. $\mathrm{w}(\square s_i) \geq e$ **do**
6: $\quad L \leftarrow 2L$
7: $\quad$ **for** $g = 0, \ldots, q-1$ **do**
8: $\quad\quad$ Compute intervals $\mathcal{P}(c + r\zeta^g, L)$ and $\mathcal{P}'(c + r\zeta^g, L)$
9: $\quad$ **if** $\exists g \in \{0, \ldots, q-1\}$ s.t. $|\mathcal{P}(c + r\zeta^g, L)| < \ell$ **or** $\left|\frac{\mathcal{P}'(c+r\zeta^g, L)}{\mathcal{P}(c+r\zeta^g, L)}\right| > \ell'$ **then**
10: $\quad\quad$ **return** $false$, $[\square s_0, \ldots, \square s_h]$
11: $\quad$ **if** $\exists g \in \{0, \ldots, q-1\}$ s.t. $\frac{\ell}{2} \in |\mathcal{P}(c + r\zeta^g, L)|$ **or** $2\ell' \in \left|\frac{\mathcal{P}'(c+r\zeta^g, L)}{\mathcal{P}(c+r\zeta^g, L)}\right|$ **then**
12: $\quad\quad$ **continue**
13: $\quad$ **for** $i = 0, \ldots, h$ **do**
14: $\quad\quad \square s_i^* \leftarrow \square \widetilde{s}_i^{\,q}(p, c, r, L)$ *//as in Eq. (11)*
15: $\quad\quad \square s_i \leftarrow \square s_i^* + [-e', e'] + \mathbf{i}[-e', e']$
16: **return** $true$, $[\square s_0, \ldots, \square s_h]$

3 Exclusion Test and Root Counters

In this section we define and analyse our base tools for disc exclusion and root counting. We recall in Subsect. 3.1 and Subsect. 3.2 the RC and the ET presented in [6]. In Subsect. 3.3, we propose a heuristic certification of root counting in which the assumed isolation for a disc Δ is heuristically verified by applying sufficiently many ETs on the contour of Δ.

For $d \geq 1, r > 0$ and $\theta > 1$, define

$$C(d, r, e, \theta) := \log(L(d, r, e, \theta)) \log_\theta(d/e) \tag{12}$$

and $C(d, r) = C(d, r, 1/4, 2)$.

3.1 Root Counting with Known Isolation

For a disc Δ which is at least θ-isolated for $\theta > 1$, Algorithm 2 computes the number m of roots in Δ as the unique integer in the interval of width < 1 obtained by approximating 0-th cauchy sum of p_Δ in the unit disc within error $< 1/2$.

Proposition 9. *Let $\Delta = D(c, r)$.* **CauchyRC1**$(\mathcal{P}, \mathcal{P}', \Delta, \theta)$ *requires evaluation of $\mathcal{P}$ and $\mathcal{P}'$ at $O(C(d, r, 1, \theta))$ points and $O(C(d, r, 1, \theta))$ arithmetic operations, all with precision less than $L(d, r, 1/4, \theta)$. Let m be the output of the latter call.*

(a) If $A(c, r/\theta, r\theta)$ contains no roots of p then $m = \#(\Delta, p)$.
(b) If $m \neq 0$ then p has a root in the disc $\theta\Delta$.

Algorithm 2. CauchyRC1$(\mathcal{P}, \mathcal{P}', \Delta, \theta)$

Require: $\mathcal{P}$, $\mathcal{P}'$ evaluation oracles for p and p', s.t. p is monic of degree d. $\Delta = D(c, r)$, $\theta \in \mathbb{R}, \theta > 1$.
Ensure: An integer $m \in \{-1, 0, \ldots, d\}$.
1: $(success, [\square s_0]) \leftarrow$ **ApproxShs**$(\mathcal{P}, \mathcal{P}', \Delta, \theta, 0, 1)$
2: **if** $success = false$ or $\square s_0$ contains no integer **then**
3: **return** -1
4: **return** the unique integer in $\square s_0$

Proposition 9 is a direct consequence of Proposition 8: in each execution of the while loop in **ApproxShs**$(\mathcal{P}, \mathcal{P}', \Delta, \theta, 0, 1)$, $\mathcal{P}$ and $\mathcal{P}'$ are evaluated at $O(\log_\theta d/e)$ points and the while loop executes an $O(\log(L(d, r, 1, \theta)))$ number of times.

3.2 Cauchy Exclusion Test

We follow [6] and increase the chances for obtaining a correct result for the exclusion of a disc with unknown isolation by approximating the first three power sums of p_Δ in $D(0, 1)$ in Algorithm 3. One has:

Proposition 10. *Let* $\Delta = D(c, r)$. **CauchyET**$(\mathcal{P}, \mathcal{P}', \Delta)$ *requires evaluation of* $\mathcal{P}$ *and* $\mathcal{P}'$ *at* $O(C(d, r))$ *points and* $O(C(d, r))$ *arithmetic operations, all with precision less than* $L(d, r)$. *Let* m *be the output of the latter call.*

(a) If $D(c, 4r/3)$ *contains no roots of* p *then* $m = 0$. *Let* B *be a box so that* $2B$ *contains no root and suppose* $\Delta = D(B)$; *then* $m = 0$.
(b) If $m \neq 0$ *then* p *has a root in the disc* $(4/3)\Delta$.

Algorithm 3. CauchyET$(\mathcal{P}, \mathcal{P}', \Delta)$

Require: $\mathcal{P}$, $\mathcal{P}'$ evaluation oracles for p and p', s.t. p is monic of degree d. $\Delta = D(c, r)$.
Ensure: An integer $m \in \{-1, 0\}$.
1: $(success, [\square s_0, \square s_1, \square s_2]) \leftarrow$ **ApproxShs**$(\mathcal{P}, \mathcal{P}', \Delta, 4/3, 2, 1)$
2: **if** $success = false$ **or** $0 \notin \square s_0$ **or** $0 \notin \square s_1$ **or** $0 \notin \square s_2$ **then**
3: **return** -1
4: **return** 0

3.3 Cauchy Root Counter

We begin with a lemma illustrated in Fig. 1. See [7, Appendix A.4] for a proof.

Lemma 11. *Let* $c \in \mathbb{C}$ *and* $\rho_-, \rho_+ \in \mathbb{R}$. *Define* $\mu = \frac{\rho_+ + \rho_-}{2}$, $\rho = \frac{\rho_+ - \rho_-}{2}$, $w = \frac{\mu}{\rho}$, $v = \lceil 2\pi w \rceil$ *and* $c_j = c + \mu e^{j\frac{2\pi \mathbf{i}}{v}}$ *for* $j = 0, \ldots, v - 1$. *Then the re-union of the discs* $D(c_j, (5/4)\rho)$ *covers the annulus* $A(c, \rho_-, \rho_+)$.

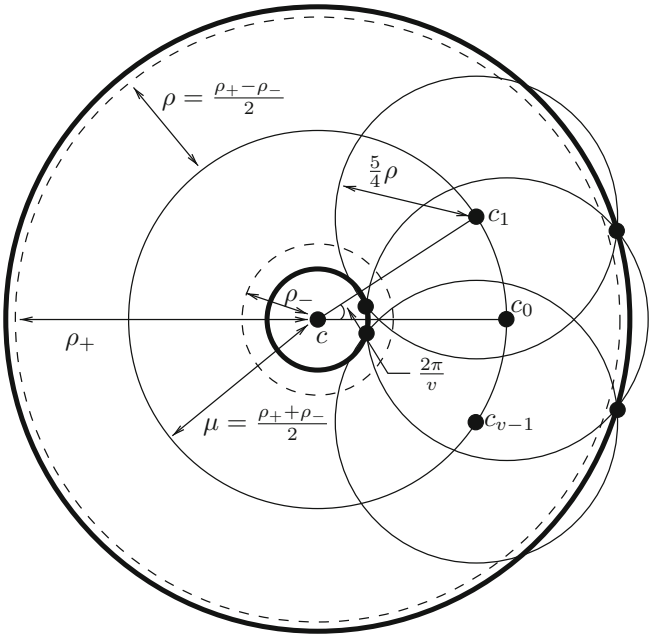

Fig. 1. Illustration for Lemma 11. In bold line, the inner and outer circles of the annulus covered by the v discs $D(c_j, (5/4)\rho)$.

For a disc $D(c, r)$ and a given $a > 1$, we follow Lemma 11 and cover the annulus $A(c, r/a, ra)$ with v discs of radius $r\frac{5(a-1/a)}{4*2}$ centered at v equally spaced points of the boundary circle of $D(c, r\frac{a+1/a}{2})$. Define

$$f_-(a, \theta) = \frac{1}{2}(a(1 - \frac{5}{4}\theta) + \frac{1}{a}(1 + \frac{5}{4}\theta)) \tag{13}$$

and

$$f_+(a, \theta) = \frac{1}{2}(a(1 + \frac{5}{4}\theta) + \frac{1}{a}(1 - \frac{5}{4}\theta)), \tag{14}$$

then the annulus $A(c, rf_-(a, \theta), rf_+(a, \theta))$ covers the θ-inflation of those v discs.

Algorithm 4 counts the number of roots of p in a disc and satisfies:

Proposition 12. *The call* **CauchyRC2**$(\mathcal{P}, \mathcal{P}', \Delta, a)$ *amounts to* $\lceil 2\pi\frac{a^2+1}{a^2-1}\rceil$ *calls to* **CauchyET** *and one call to* **CauchyRC1**.

Let $\Delta = D(c, r)$ *and* A *be the annulus* $A\left(c, rf_-(a, \frac{4}{3}), rf_+(a, \frac{4}{3})\right)$. *Let* m *be the output of the latter call.*

(a) If A *contains no root then* $m \geq 0$ *and* Δ *contains* m *roots.*
(b) If $m \neq 0$, *then* A *contains a root.*

We state the following corollary.

Algorithm 4. CauchyRC2$(\mathcal{P}, \mathcal{P}', \Delta, a)$

Require: $\mathcal{P}$, $\mathcal{P}'$ evaluation oracles for p and p', s.t. p is monic of degree d. $\Delta = D(c, r)$. $a \in \mathbb{R}, a > 1$.
Ensure: An integer $m \in \{-1, 0, \dots, d\}$.
// Verify that Δ is at least a-isolated with **CauchyET**
$\rho_- \leftarrow \frac{1}{a}r$, $\rho_+ = ar$.
$\rho \leftarrow \frac{\rho_+ - \rho_-}{2}$, $\mu \leftarrow \frac{\rho_+ + \rho_-}{2}$, $w \leftarrow \frac{\mu}{\rho}$, $v \leftarrow \lceil 2\pi w \rceil$, $\zeta \leftarrow \exp(\frac{2\pi \mathbf{i}}{v})$
for $i = 0, \dots, v-1$ **do**
  $c_i \leftarrow c + \mu\zeta^i$
  if **CauchyET**$(\mathcal{P}, \mathcal{P}', D(c_i, \frac{5}{4}\rho))$ returns -1 **then**
    return -1 *// $A\left(c, rf_-(a, \frac{4}{3}), rf_+(a, \frac{4}{3})\right)$ contains a root*
// Δ is at least a-isolated according to **CauchyET**
return **CauchyRC1**$(\mathcal{P}, \mathcal{P}', \Delta, a)$

Corollary 13. (of Proposition 12) *Let $\theta = 4/3$ and $a = 11/10$. Remark that*

$$f_-(a, \theta) = \frac{93}{110} > 2^{-1/4} \text{ and } f_+(a, \theta) = \frac{64}{55}.$$

The call **CauchyRC2**$(\mathcal{P}, \mathcal{P}', \Delta, a)$ *amounts to $\lceil 2\pi \frac{a^2+1}{a^2-1} \rceil = 67$ calls to* **CauchyET** *for discs of radius $\frac{21}{176}r \in O(r)$ and one call to* **CauchyRC1** *for Δ. This requires evaluation of $\mathcal{P}$ and $\mathcal{P}'$ at $O(C(d, r))$ points, and $O(C(d, r))$ arithmetic operations, all with precision less than $L(d, r)$.*

4 Root Radii Algorithms

4.1 Approximation of the Largest Root Radius

For a monic p of degree d and bit-size $\tau = \log \|p\|_1$, we describe a naive approach to the approximation of the largest modulus r_d of a root of p. Recall Cauchy's bound for such a polynomial: $r_d \leq 1 + 2^\tau$. The procedure below finds an r so that $r_d < r$ and either $r = 1$ or $r/2 < r_d$ when p is given by the evaluation oracles $\mathcal{P}, \mathcal{P}'$.

$r \leftarrow 1$, $m \leftarrow -1$
while $m \leq d$ **do**
  $m \leftarrow$ **CauchyRC2**$(\mathcal{P}, \mathcal{P}', D(0, r), 4/3)$
  if $m < d$ **then**
    $r \leftarrow 2r$

As a consequence of Proposition 12 each execution of the while loop terminates and the procedure terminates after no more than $O(\tau)$ execution of the **while** loop. It requires evaluation of $\mathcal{P}$ and $\mathcal{P}'$ at $O(\tau C(d, r))$ points and $O(\tau C(d, r))$ arithmetic operations all with precision less than $L(d, r)$. Its correctness is implied by correctness of the results of **CauchyRC2** which is in turn implied by correctness of the results of **CauchyET**.

4.2 Approximation of the $(d+1-m)$-th Root Radius

For a $c \in \mathbb{C}$ and an integer $m \geq 1$, we call $(d+1-m)$-th root radius from c and write it $r_m(c,p)$ the smallest radius of a disc centered in c and containing exactly m roots of p.

Algorithm 5 approximates $r_m(c,p)$ within the relative error ν. It is based on the RC **CauchyRC2** and reduces the width of an initial interval $[l,u]$ containing $r_m(c,p)$ with a double exponential sieve.

Algorithm 5. RootRadius$(\mathcal{P},\mathcal{P}',\Delta,m,\nu,\varepsilon)$

Require: $\mathcal{P}$, $\mathcal{P}'$ evaluation oracles for p and p', s.t. p is monic of degree d. A disc $\Delta = D(c,r)$, an integer $m \geq 1$, $\nu \in \mathbb{R}$, $\nu > 1$, and $\varepsilon \in \mathbb{R}$ such that $0 < \varepsilon \leq r/2$
Ensure: $r' > 0$
choose a s.t. $\nu^{-\frac{1}{4}} < f_-(a,\frac{4}{3}) < f_+(a,\frac{4}{3}) < 2$ // *when* $\nu = 2$ *take* $a = 11/10$
$l \leftarrow 0$, $u \leftarrow r$
// *Find a lower bound to* $r_{d+1-m}(c,p)$
$m' \leftarrow$ **CauchyRC2**$(\mathcal{P},\mathcal{P}',D(c,\varepsilon),a)$
if $m' = m$ **then**
  return ε
else
  $l \leftarrow f_-(a,\frac{4}{3})\varepsilon$
  // *Apply double exponential sieve to get* $l \leq r_{d+1-m} \leq u \leq \nu l$
while $l < u/\nu$ **do**
  $t \leftarrow (lu)^{\frac{1}{2}}$
  $m' \leftarrow$ **CauchyRC2**$(\mathcal{P},\mathcal{P}',D(c,t),a)$
  if $m' = m$ **then**
    $u \leftarrow t$
  else
    $l \leftarrow f_-(a,\frac{4}{3})t$
return u

The correctness of Algorithm 5 for given input parameters is implied by correctness of the results of **CauchyRC2** which is in turn implied by correctness of the results of **CauchyET**. Algorithm 5 satisfies the proposition below. See [7, Appendix A.5] for a proof.

Proposition 14. *The call* **RootRadius**$(\mathcal{P},\mathcal{P}',D(c,r),m,\nu,\varepsilon)$ *terminates after* $O(\log\log(r/\varepsilon))$ *iterations of the while loop. Let* $\Delta = D(c,r)$ *and* r' *be the output of the latter call.*

(a) If Δ *contains at least a root of* p *then so does* $D(c,2r')$.
(b) If Δ *contains* m *roots of* p *and* **CauchyRC2** *returns a correct result each time it is called in Algorithm 5, then either* $r' = \varepsilon$ *and* $r_m(c,p) \leq \varepsilon$, *or* $r_m(c,p) \leq r' \leq \nu r_m(c,p)$.

5 A Compression Algorithm

We begin with a geometric lemma illustrated in Fig. 2.

Lemma 15. *Let $c \in \mathbb{C}$ and $r, \varepsilon, \theta \in \mathbb{R}$ satisfying $0 < \varepsilon \leq r/2$ and $\theta \geq 2$. Let $c' \in D(c, \frac{r+\varepsilon}{\theta})$ and $u = \max\left(|c - c'| + \frac{r}{\theta}, r\right)$. Then*

$$D\left(c, \frac{r}{\theta}\right) \subseteq D\left(c', u\right) \subseteq D\left(c, \frac{7}{4}r\right) \subset D(c, r\theta).$$

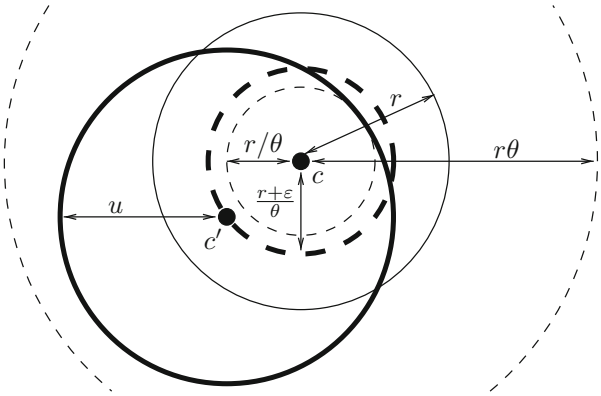

Fig. 2. Illustration for Lemma 15 with $\theta = 2$ and $\varepsilon = r/4$. c' is on the boundary circle of $D(c, (r+\varepsilon)/2)$, and $u := |c - c'| + r/\theta$.

The following lemma is a direct consequence of Lemma 15 because $s_1(p, c, r)/m$ is the center of gravity of the roots of p in $D(c, r)$.

Lemma 16. *Let $D(c, r)$ be at least $\theta \geq 2$-isolated and contain m roots. Let s_1^* approximate $s_1(p, c, r)$ such that $|s_1^* - s_1(p, c, r)| \leq \frac{m\varepsilon}{\theta}$ and $\varepsilon \leq \frac{r}{2}$. Then for $c' = \frac{s_1^*}{m}$ and $u = \max\left(|c - c'| + \frac{r}{\theta}, r\right)$, the disc $D(c', u)$ contains the same roots of p as $D(c, r)$.*

Algorithm 6 solves the ε-CRD problem for $\gamma = 1/8$. It satisfies the proposition below. See [7, Appendix A.6] for a proof.

Proposition 17. *The call* **Compression**$(\mathcal{P}, \mathcal{P}', \Delta, \varepsilon)$ *where $\Delta = D(c, r)$ requires evaluation of $\mathcal{P}$ and $\mathcal{P}'$ at $O\left(C\left(d, \varepsilon\right) \log\log\frac{r}{\varepsilon}\right)$ points and the same number of arithmetic operations, all with precision less than $L\left(d, \varepsilon/4\right)$. Let $m, D(c', r')$ be the output of the latter call.*

(a) If Δ is at least 2-isolated and $Z\left(\Delta, p\right) \neq \emptyset$, and if the call to **RootRadius** *returns a correct result, then $D(c', r')$ is equivalent to Δ, contains m roots of p and satisfies: either $r' \leq \varepsilon$, or $D(c', r')$ is at least $1/8$-rigid.*
(b) If $m' > 0$ then $D(c', 2r')$ contains at least a root of p.

Algorithm 6. Compression$(\mathcal{P}, \mathcal{P}', \Delta, \varepsilon)$

Require: $\mathcal{P}$, $\mathcal{P}'$ evaluation oracles for p and p', s.t. p is monic of degree d. A disc $\Delta = D(c, r)$, and a strictly positive $\varepsilon \in \mathbb{R}$.
Ensure: An integer m and a disc $D(c', r')$.

$\theta \leftarrow 2$, $\varepsilon' \leftarrow \varepsilon/2\theta$
$(success, [\square s_0, \square s_1]) \leftarrow$ **ApproxShs**$(\mathcal{P}, \mathcal{P}', \Delta, \theta, 1, \min(\varepsilon', 1))$
if not *success* **or** $\square s_0$ does not contain an integer > 0 **then**
    return $-1, \emptyset$
$m \leftarrow$ the unique integer in $\square s_0$
if $r/2 < \varepsilon$ **then**
    return m, $D(c, r/2)$
$c' \leftarrow \mathrm{c}\left(\square s_1\right)/m$ // $|c' - s_1(p, c, r)/m| < \varepsilon/4\theta$
if $m = 1$ **then**
    $m \leftarrow$ **CauchyRC1**$(\mathcal{P}, \mathcal{P}', D(c', 2\varepsilon'), 2)$
    return m, $D(c', 2\varepsilon')$
$u \leftarrow \max\left(|c - c'| + \frac{r}{\theta}, r\right)$
$r' \leftarrow$ **RootRadius**$(\mathcal{P}, \mathcal{P}', D(c', u), \frac{4}{3}, m, \theta, \varepsilon/2)$
return m, $D(c', r')$

6 Two Cauchy Root Finders

In order to demonstrate the efficiency of the algorithms presented in this paper, we describe here two experimental subdivision algorithms, named `CauchyQIR` and `CauchyComp`, solving the ε-CRC problem for oracle polynomials based on our Cauchy ET and RCs. Both algorithms can fail –in the case where **CauchyET** excludes a box of the subdivision tree containing a root – but account for such a failure. Both algorithm adapt the subdivision process described in [2]. `CauchyQIR` uses QIR Abbott iterations to ensure fast convergence towards clusters of roots. `CauchyComp` uses ε-compression presented in Sect. 5. In both solvers, the main subdivision loop is followed by a post-processing step to check that the output is a solution of the ε-CRC problem. The main subdivision loop does not involve coefficients of input polynomials but use evaluation oracles instead. However, we use coefficients obtained by evaluation-interpolation in the post-processing step in the case where some output discs contain more than one root. We observe no failure of our algorithms in all our experiments covered in Sect. 7.

6.1 Subdivision Loop

Let B_0 be a box containing all the roots of p. Such a box can be obtained by applying the process described in Subsect. 4.1.

Sub-Boxes, Component and Quadrisection. For a box $B(a + \mathbf{i}b, w)$, let $Children_1(B)$ be the set of the four boxes $\{B((a \pm w/4) + \mathbf{i}(b \pm w/4), w/2)\}$, and

$$Children_n(B) := \bigcup_{B' \in Children_{n-1}(B)} Children_1(B').$$

A box B is a *sub-box* of B_0 if $B = B_0$ or if there exist an $n \geq 1$ s.t. $B \in Children_n(B_0)$. A *component* C is a set of connected sub-boxes of B_0 of equal widths. The *component box* $B(C)$ of a component C is the smallest (square) box subject to $C \subseteq B(C) \subseteq B_0$ minimizing both $\mathcal{R}e(\mathrm{c}\,(B(C)))$ and $\mathcal{I}m(\mathrm{c}\,(B(C)))$. We write $D(C)$ for $D(B(C))$. If S is a set of components (resp. discs) and $\delta > 0$, write δS for the set $\{\delta D(A)$ (resp. $A) \mid A \in S\}$.

Definition 18. *Let Q be a set of components or discs. We say that a component C (resp. a disk Δ) is γ-separated (or γ-sep.) from Q when $\gamma D(C)$ (resp. $\gamma\Delta$) has empty intersection with all elements in Q.*

Remark 19. *Let Q be a set of components and $C \notin Q$ a component. If $Z\,(\mathbb{C}, p) = Z\,(\{C\} \cup Q, p)$ and C is 4-separated from Q then $2D(C)$ is at least 2-isolated.*

Subdivision Process. We describe in Algorithm 7 a subdivision algorithm solving the ε-CRC problem. The components in the working queue Q are sorted by decreasing radii of their containing discs. It is parameterized by the flag *compression* indicating whether compression or QIR Abbott iterations have to be used. In QIR Abbott iterations of Algorithm 7 in [3], we replace the Graeffe Pellet test for counting roots in a disc Δ by **CauchyRC2**$(\mathcal{P}, \mathcal{P}', \Delta, 4/3)$. If a QIR Abbott iteration in step 12 fails for input Δ, m, it returns Δ. Steps 20–21 prevent C to artificially inflate when a compression or a QIR Abbott iteration step does not decrease $D(C)$. For a component C, $Quadrisect(C)$ is the set of components obtained by grouping the set of boxes

$$\bigcup_{B \in C} \{B' \in Children_1(B) \mid \mathbf{CauchyET}(\mathcal{P}, \mathcal{P}', D(B')) = -1\}$$

into components.

The **while** loop in steps 4-22 terminates because all our algorithms terminate, and as a consequence of (a) in Proposition 9: any component will eventually be decreased until the radius of its containing disc reaches $\varepsilon/2$.

6.2 Output Verification

After the subdivision process described in steps 1–22 of Algorithm. 7, R is a set of pairs of the form $\{(\Delta^1, m^1), \ldots, (\Delta^\ell, m^\ell)\}$ satisfying, for any $1 \leq j \leq \ell$:

- Δ^j is a disc of radii $\leq \varepsilon$, m^j is an integer ≥ 1,
- Δ^j contains at least a root of p,
- for any $1 \leq j' \leq \ell$ s.t. $j' \neq j$, $3\Delta^j \cap \Delta^{j'} = \emptyset$.

The second property follows from (b) of Proposition 10 and (b) of Proposition 17 when compression is used. Otherwise, remark that a disk Δ in the output of QIR Abbott iteration in step 12 of Algorithm 7 verifies **CauchyRC2**$(\mathcal{P}, \mathcal{P}', \Delta, 4/3) > 0$ and apply (b) of Proposition 12. The third property follows from the **if** statement in step 15 of Algorithm 7. Decompose R as the disjoint union $R_1 \cup R_{>1}$ where R_1 is the subset of pairs (Δ^i, m^i) of R where $m^i = 1$ and $R_{>1}$ is the subset of pairs (Δ^i, m^i) of R where $m^i > 1$, and make the following remark:

Algorithm 7. CauchyRootFinder$(\mathcal{P}, \mathcal{P}', \varepsilon, compression)$

Require: $\mathcal{P}$ and $\mathcal{P}'$ evaluation oracles for p and p', s.t. p is monic of degree d. A (strictly) positive $\varepsilon \in \mathbb{R}$, a flag $compression \in \{true, false\}$.
Ensure: A flag *success* and a list $R = \{(\Delta^1, m^1), \ldots, (\Delta^\ell, m^\ell)\}$

$B_0 \leftarrow$ box s.t. $\#(B, p) = d$ as described in Subsect. 4.1
$Q \leftarrow \{B_0\}$ // *Q is a queue of components*
$R \leftarrow \{\}$ // *R is the empty list of results*
while Q is not empty **do**
  $C \leftarrow pop(Q)$
  if C is 4-separated from Q **then**
    if *compression* **then**
      $m, D(c, r) \leftarrow$ **Compression**$(\mathcal{P}, \mathcal{P}', 2D(C), \varepsilon/2)$
    else
      $m \leftarrow$ **CauchyRC1**$(\mathcal{P}, \mathcal{P}', 2D(C), 2)$
      if $m > 0$ **then**
        $D(c, r) \leftarrow$ QIR Abbott iteration for $D(C), m$
    if $m \leq 0$ **then**
      return $fail, \emptyset$
    if $r \leq \varepsilon/2$ **and** $D(c, 2r)$ is 3-sep. from $2Q$ **and** is 1-sep. from $6Q$ **then**
      $push(R, (D(c, 2r), m))$
      continue
    else
      $C' \leftarrow$ component containing $D(c, r)$
      if $C' \subset C$ **then**
        $C \leftarrow C'$
  $push(Q, Quadrisect(C))$
success $\leftarrow$ verify R as described in Subsect. 6.2
return *success*, R

Remark 20. *If* $m^1 + \ldots + m^\ell = d$ *and for any* $(\Delta^i, m^i) \in R_{>1}$, Δ^i *contains exactly* m^i *roots of* p, *then* R *is a correct output for the* ε*-CRC problem with input* p *of degree* d *and* ε.

According to Remark 20, checking that R is a correct output for the ε-CRC problem for fixed input p of degree d and ε amount to check that the m^i's add up to d and that for any $\Delta^i \in R_{>1}$, Δ^i contains exactly m^i roots of p. For this last task, we use evaluation-interpolation to approximate the coefficients of p and then apply the Graeffe-Pellet test of [2].

7 Experiments

We implemented Algorithm 7 in the `C` library `Ccluster`. Call `CauchyComp` (resp. `CauchyQIR`) the implementation of Algorithm 7 with *compression* = *true* (resp. *false*). In the experiments we conducted so far, `CauchyComp` and `CauchyQIR` never failed.

Test Suite. We experimented `CauchyComp`, `CauchyQIR` and `MPsolve` on Mandelbrot and Mignotte polynomials as defined in Sect. 1 as well as Runnel and random sparse polynomials. Let $r = 2$. The Runnel polynomial is defined inductively as

$$\mathtt{Run}_0(z) = 1, \quad \mathtt{Run}_1(z) = z, \quad \mathtt{Run}_{k+1}(z) = \mathtt{Run}_k(z)^r + z\,\mathtt{Run}_{k-1}(z)^{r^2}$$

It has real coefficients, a multiple root (zero), and can be evaluated fast. We generate random sparse polynomials of degree d, bitsize τ and $\ell \geq 2$ nonzero terms as follows, where p_i stands for the coefficient of the monomial of degree i in p: p_0 and p_d are randomly chosen in $[-2^{\tau-1}, 2^{\tau-1}]$, then $\ell - 2$ integers $i_1, \ldots, i_{\ell-1}$ are randomly chosen in $[1, d-1]$ and $p_{i_1}, \ldots, p_{i_{\ell-1}}$ are randomly chosen in $[-2^{\tau-1}, 2^{\tau-1}]$. The other coefficients are set to 0.

Results. We report in Table 1 results of those experiments for Mandelbrot and Mignotte polynomials with increasing degrees and increasing values of $\log_{10}(\varepsilon^{-1})$. We account for the running time t for the three above-mentionned solvers. For `CauchyQIR` (resp. `CauchyComp`), we also give the number n of exclusion tests in the subdivision process, and the time t_N (resp. t_C) spent in QIR Abbott iterations (resp. compression).

Our compression algorithm allows smaller running times for low values of $\log_{10}(\varepsilon^{-1})$ because it compresses a component C on the cluster it contains as of $2D(C)$ is 2-isolated, whereas QIR Abbott iterations require the radius Δ to be near the radius of convergence of the cluster for Schröder's iterations.

We report in Table 2 the results of runs of `CauchyComp` and `MPsolve` for polynomials of our test suite of increasing degree, for $\log_{10}(\varepsilon^{-1}) = 16$. For random sparse polynomials, we report averages over 10 examples. The column t_V accounts for the time spent in the verification of the output of `CauchyComp` (see Subsect. 6.2); it is 0 when all the pairs (Δ^j, m^j) in the output verify $m^j = 1$. It is > 0 when there is at least a pair with $m^j > 1$.

The maximum precision L required in all our tests was 106, which makes us believe that our analysis in Proposition 8 is very pessimistic. Our experimental solver `CauchyComp` is faster than `MPsolve` for polynomials that can be evaluated fast.

Table 2. Runs of `CauchyComp` and `MPsolve` on polynomials of our test suite for $\log_{10}(\varepsilon^{-1}) = 16$

	`CauchyComp`				`MPsolve`
d	t	n	t_C	t_V	t
Mandelbrot polynomials					
255	1.31	5007	0.21	0.00	**0.58**
511	**3.25**	10679	0.64	0.00	4.13
1023	**6.47**	18774	0.84	0.00	31.7
2047	**16.2**	39358	2.35	0.00	267.
Runnels polynomials					
341	2.55	4967	0.38	0.00	**0.45**
682	5.66	9392	0.87	0.02	**3.32**
1365	**12.6**	18030	2.00	0.05	26.2
2730	**29.7**	35612	4.26	0.12	236.
Mignotte polynomials, $a = 16$					
256	0.29	4131	0.15	0.00	**0.21**
512	**0.58**	8042	0.27	0.00	0.70
1024	**1.24**	16105	0.55	0.02	2.99
2048	**2.69**	32147	1.05	0.04	11.6
10 randomSparse polynomials with 3 terms and bitsize 256					
767	.902	10791.	.415	0.0	**.602**
1024	**1.35**	15526.	.560	0.0	1.36
1535	**2.04**	21244.	.861	0.0	2.35
2048	**2.98**	30642.	1.16	0.0	4.10
10 randomSparse polynomials with 5 terms and bitsize 256					
2048	4.77	29583.	1.60	0.0	**4.09**
3071	**6.92**	43003.	2.45	0.0	10.0
4096	**9.82**	56659.	3.38	0.0	24.0
6143	**17.7**	86857.	5.40	0.0	44.5
10 randomSparse polynomials with 10 terms and bitsize 256					
3071	11.9	44714.	4.09	0.0	**10.3**
4096	**17.5**	58138.	5.82	0.0	17.6
6143	**29.1**	85451.	8.93	0.0	51.9
8192	**40.6**	116289.	12.4	0.0	66.5

8 Conclusion

We presented, analyzed and verified practical efficiency of two basic subroutines for solving the complex root clustering problem for black box polynomials. One is a root counter, the other one is a compression algorithm. Both algorithms are well-known tools used in subdivision procedures for root finding.

We propose our compression algorithm not as a replacement of QIR Abbott iterations, but rather as a complementary tool: in future work, we plan to use

compression to obtain a disc where Schröder's/QIR Abbott iterations would converge fast. The subroutines presented in this paper laid down the path toward a Cauchy Root Finder, that is, an algorithm solving the ε-CRC problem for black box polynomials.

References

1. Abbott, J.: Quadratic interval refinement for real roots. ACM Commun. Comput. Algebra **48**(1/2), 3–12 (2014)
2. Becker, R., Sagraloff, M., Sharma, V., Xu, J., Yap, C.: Complexity analysis of root clustering for a complex polynomial. In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, pp. 71–78. ACM (2016)
3. 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. Symbol. Comput. **86**, 51–96 (2018)
4. Bini, D.A., Robol, L.: Solving secular and polynomial equations: a multiprecision algorithm. J. Comput. Appl. Math. **272**, 276–292 (2014)
5. Imbach, R., Pan, V.Y.: New practical advances in polynomial root clustering. In: Slamanig, D., Tsigaridas, E., Zafeirakopoulos, Z. (eds.) MACIS 2019. LNCS, vol. 11989, pp. 122–137. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-43120-4_11
6. Imbach, R., Pan, V.Y.: New progress in univariate polynomial root finding. In: Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation, pp. 249–256 (2020)
7. Imbach, R., Pan, V.Y.: Accelerated subdivision for clustering roots of polynomials given by evaluation oracles. arXiv preprint 2206.08622 (2022)
8. Imbach, R., Pan, V.Y., Yap, C.: Implementation of a near-optimal complex root clustering algorithm. In: Davenport, J.H., Kauers, M., Labahn, G., Urban, J. (eds.) ICMS 2018. LNCS, vol. 10931, pp. 235–244. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96418-8_28
9. Louis, A., Vempala, S.S.: Accelerated Newton iteration: roots of black box polynomials and matrix eigenvalues. In: IEEE 57th Annual Symposium on Foundations of Computer Science, pp. 732–740 (2016)
10. Luan, Q., Pan, V.Y., Kim, W., Zaderman, V.: Faster numerical univariate polynomial root-finding by means of subdivision iterations. In: Boulier, F., England, M., Sadykov, T.M., Vorozhtsov, E.V. (eds.) CASC 2020. LNCS, vol. 12291, pp. 431–446. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60026-6_25
11. Moroz, G.: Fast real and complex root-finding methods for well-conditioned polynomials. arXiv preprint 2102.04180 (2021)
12. Pan, V.Y.: Approximating complex polynomial zeros: modified Weyl's quadtree construction and improved Newton's iteration. J. Complex. **16**, 213–264 (2000)
13. Pan, V.Y.: Univariate polynomials: nearly optimal algorithms for numerical factorization and root-finding. J. Symbol. Comput. **33**, 701–733 (2002)
14. Pan, V.Y.: Old and new nearly optimal polynomial root-finders. In: England, M., Koepf, W., Sadykov, T.M., Seiler, W.M., Vorozhtsov, E.V. (eds.) CASC 2019. LNCS, vol. 11661, pp. 393–411. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26831-2_26

15. Pan, V.Y.: Acceleration of subdivision root-finding for sparse polynomials. In: Boulier, F., England, M., Sadykov, T.M., Vorozhtsov, E.V. (eds.) CASC 2020. LNCS, vol. 12291, pp. 461–477. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60026-6_27
16. Pan, V.Y.: New progress in polynomial root-finding. arXiv preprint 1805.12042 (2022)
17. Reinke, B.: Diverging orbits for the Ehrlich-Aberth and the Weierstrass root finders. arXiv preprint 2011.01660 (2020)
18. Renegar, J.: On the worst-case arithmetic complexity of approximating zeros of polynomials. J. Complex. **3**(2), 90–113 (1987)
19. Sagraloff, M., Mehlhorn, K.: Computing real roots of real polynomials. J. Symbol. Comput. **73**, 46–86 (2016)
20. Schönhage, A.: The fundamental theorem of algebra in terms of computational complexity. Manuscript. University of Tübingen, Germany (1982)