Root Radii and Subdivision for Polynomial Root-Finding

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

[1] Courant Institute of Mathematical Sciences, New York University, New York, USA
remi.imbach@nyu.edu

[2] Lehman College and Graduate Center of City University of New York,
New York, USA
victor.pan@lehman.cuny.edu

Abstract. We depart from our approximation of 2000 of all root radii of a polynomial, which has readily extended Schönhage's efficient algorithm of 1982 for a single root radius. We revisit this extension, advance it, based on our simple but novel idea, and yield significant practical acceleration of the known near optimal subdivision algorithms for complex and real root-finding of user's choice. We achieve this by means of significant saving of exclusion tests and Taylor's shifts, which are the bottleneck of subdivision root-finders. This saving relies on our novel recipes for the initialization of root-finding iterations of independent interest. We demonstrate our practical progress with numerical tests, provide extensive analysis of the resulting algorithms, and show that, like the preceding subdivision root-finders, they support near optimal Boolean complexity bounds.

Keywords: Real root isolation · Complex root clustering · Root radii algorithm · Subdivision iterations

1 Introduction

Overview. The recent subdivision iterations for univariate polynomial Complex Root Clustering (CRC) and Real Root Isolation (RRI) approximate all roots in a fixed Region of Interest (RoI) and, like the algorithm of Pan (1995, 2002), achieve near optimal bit complexity for the so called benchmark problem. Furthermore they allow robust implementations, one of which is currently the user's choice for solving the RRI problem, including the task of the approximation of all real roots. Another implementation, for the CRC problem, is slower by several orders of magnitude than the package MPSolve (the user's choice) for the task of finding all complex roots. However it outperforms MPSolve for solving the CRC problem where the RoI contains only a small number of roots. We significantly accelerate these highly efficient root-finding iterations by applying our novel

Victor's work is supported by NSF Grants CCF 1563942 and CCF 1733834 and PSC CUNY Award 63677 00 51.

F. Boulier et al. (Eds.): CASC 2021, LNCS 12865, pp. 136–156, 2021.
https://doi.org/10.1007/978-3-030-85165-1_9

techniques for their initialization. Next we specify the background and outline our contributions.

Polynomial Roots and Root Radii. For a polynomial P of degree d in $\mathbb{Z}[z]$ and a Gaussian integer $c \in \mathbb{G} := \{a + \mathbf{i}b \mid a \in \mathbb{Z}, b \in \mathbb{Z}\}$, let $\alpha_1(P,c), \ldots, \alpha_d(P,c)$ be the d non-necessarily distinct roots of P such that

$$|\alpha_1(P,c) - c| \geq |\alpha_2(P,c) - c| \geq \ldots \geq |\alpha_{d-1}(P,c) - c| \geq |\alpha_d(P,c) - c|. \quad (1)$$

For all $1 \leq i \leq d$, write $r_i(P,c) := |\alpha_i(P,c) - c|$, $\alpha_i(P) := \alpha_i(P,0)$ and $r_i(P) := r_i(P,0)$, so that

$$r_1(P,c) \geq r_2(P,c) \geq \ldots \geq r_{d-1}(P,c) \geq r_d(P,c). \quad (2)$$

Then

Root Radii Covering (RRC) Problem
Given: a polynomial $P \in \mathbb{Z}[z]$ of degree d, a Gaussian integer $c \in \mathbb{G}$, a real number $\delta > 0$
Output: d positive numbers $\rho_{c,1}, \ldots, \rho_{c,d}$ satisfying

$$\forall s = 1, \ldots, d, \quad \frac{\rho_{c,s}}{1+\delta} \leq r_s(P,c) \leq (1+\delta)\rho_{c,s}. \quad (3)$$

$\rho_{c,1}, \ldots, \rho_{c,d}$ of Eq. (3) for fixed $c \in \mathbb{G}$ and $\delta > 0$ define d possibly overlaping concentric annuli. The connected components of their union form a set $\mathcal{A}_c$ of $d_c \leq d$ disjoint concentric annuli centered at c. They cover all roots of P and are said to be an *annuli cover* of the roots of P. We are going to use them in subdivision root-finding iterations.

Two Root-Finding Problems. We count roots with multiplicity and consider discs $D(c,r) := \{z \mid |z - c| \leq r\}$ on the complex plane. For a positive δ let $\delta\Delta$ and δB denote the concentric δ-dilation of a disc Δ and a real line segment (*i.e.* interval) B. Then

Complex Root Clustering (CRC) Problem
Given: a polynomial $P \in \mathbb{Z}[z]$ 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 $\leq \varepsilon$,
- Δ^j and $3\Delta^j$ contain $m^j > 0$ roots,
- each complex root of P is in a Δ^j.

Real Root Isolation (RRI) Problem
Given: a polynomial $P \in \mathbb{Z}[z]$ of degree d
Output: $\ell \leq d$ couples $(B^1, m^1), \ldots, (B^\ell, m^\ell)$ satisfying:

- the B^i's are disjoint real line segments,
- each B^i contains a unique real root of multiplicity m^j,
- each real root of P is in a B^i.

Table 1. Runs of `Risolate`, `RisolateR`, `Ccluster` and `CclusterR` on two polynomials. `Ccluster` and `CclusterR` are called with input $\varepsilon = 2^{-53}$.

			`Risolate`		`RisolateR`			`Ccluster`		`CclusterR`		
d	τ	$d_{\mathbb{R}}$	t	n	t	n	t'	t	n	t	n	t'
Bernoulli polynomial												
512	2590	124	6.15	672	0.38	17	0.25	136	13940	50.7	4922	7.59
Mignotte polynomial												
512	256	4	1.57	49	1.67	14	0.27	88.8	10112	28.3	2680	3.05

It is quite common to pre-process $P \in \mathbb{Z}[z]$ in order to make it square-free, with $m^j = 1$ for all j, but we do not use this option. We can state both CRC and RRI problems for P with rational coefficients and readily reduce them to the above versions with integer coefficients by scaling.

Write $\|P\| := \|P\|_\infty$ and call the value $\log_2 \|P\|$ the *bit-size* of P.

The Benchmark Problem. For the bit-complexity of the so called *benchmark* root-finding problem of the isolation of all roots of a square-free $P \in \mathbb{Z}[z]$ of degree d and bit-size τ the record bound of 1995 [13] is $\widetilde{O}(d^2(d+\tau))$, near optimal for $\tau > d$ and based on a divide and conquer approach. It was reached again in 2016 [1,2], based on subdivision iterations and implemented in [8].

Our Contributions. We first present and analyze an algorithm `SolveRRC` that solves the RRC problem for polynomials with integer coefficients and any fixed center $c \in \mathbb{G}$. Our algorithm is adapted from work [12] (which has extended Schönhage's highly efficient approximation of a single root radius in [15]) to simultaneous approximation of all d root radii. Our specialization of this root radii algorithm to the case of integer polynomials and our analysis of its bit-complexity are novel. We use `SolveRRC` for $\delta \in d^{-O(1)}$ and $|c| \in O(1)$; under such assumptions, it solves the RRC problem with a bit-complexity in $\widetilde{O}(d^2(d+\tau))$.

We then improve solvers for the RRI and the CRC problems based on subdivision with annuli covers that we compute by applying `SolveRRC`. The complexity of subdivision root-finders is dominated at its bottleneck stages of root-counting and particularly exclusion tests, at which costly *Taylor's shifts*, aka the shifts of the variable, are applied. We significantly accelerate the root-finders for both RRI and CRC problems by means of using fewer exclusion tests and calls for root-counting and hence fewer Taylor's shifts. We achieve this by limiting complex root-finding to the intersection of three annuli covers of the roots centered in $0, 1$ and $\mathbf{i}$ and by limiting real root-finding to the intersection of a single annuli cover centered in 0 with the real line.

Our improvements are implemented within the `C` library `Ccluster`[1] which provides an eponymous solver for the CRC problem and a solver for the RRI problem called `Risolate`. Our novel solvers are called below `CclusterR` and `RisolateR`, and in Table 1 we overview how those two solvers perform against `Ccluster` and `Risolate` on a Bernoulli and a Mignotte polynomial. For each test polynomial we show its degree d, its bit-size τ, and the number $d_{\mathbb{R}}$ of real roots. For each solver, t denotes the sequential running time in seconds on an `Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz` machine with Linux and n denotes the total number of Taylor's shift required in the subdivision process. For `CclusterR` and `RisolateR`, t' is the time spent on solving the RRC problem with `SolveRRC`.

We compute the annuli covers in a pre-processing step by applying algorithm `SolveRRC` for input relative width $\delta = 1/d^2$. This choice of δ is empiric, and in this sense our improvement of subdivision is heuristic. From a theoretical point of view, this allows our algorithms for solving the RRI and the CRC problems to support a near optimal bit-complexity. From a practical point of view, this allows us to significantly reduce the running time of solvers based on subdivision by using fewer Taylor's shifts in exclusion and root-counting tests, as we highlighted in Table 1 (see the columns t and n).

The distance between roots of a polynomial of degree d and bit-size τ can be way less than $1/d^2$ (see for instance [11]); thus by computing with `SolveRRC` intervals that contain the root radii of relative width $\delta = 1/d^2$, we do not intend to separate the roots of input polynomials, and our improvement has no effect in the cases where distances between some roots are less than δ. We illustrate this in Table 1 for a Mignotte polynomial that has four real roots among which two roots have a pairwise distance that is close to the theoretical separation bound. Most of the computational effort in a subdivision solver for real roots isolation is spent on the separation of the close roots, and this remains true where we use annuli covers with relative width larger than the roots separation.

We compare our implementation with `ANewDsc` (see [10], implementing [14]) and `MPSolve` (see [3], implementing Ehrlich's iterations), which are the current user's choices for solving the RRI and the CRC problems, respectively.

Related Work. We departed from the subdivision polynomial root-finding for the CRC and RRI problems in [1] and [14], resp., and from the algorithms for the RRC problem in [15] (see [15][Cor. 14.3], and [5][Algorithm 2]) and [12]. We achieved practical progress by complementing these advanced works with our novel techniques for efficient computation of $O(1)$ annuli covers of the roots. We rely on the customary framework for the analysis of root-finding algorithms and cite throughout the relevant sources of our auxiliary techniques.

Organization of the Paper. In Sect. 2 we describe an algorithm for solving the RRC problem. In Sects. 3 and 4 we present our algorithms for solving the

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

RRI and CRC problem, respectively. In Subsect. 1.1 we introduce definitions. In Subsect. 1.2 we briefly describe the subdivision algorithm for the CRC problem of [1] and its adaptation to the solution of the RRI problem.

1.1 Definitions

Write $P := P(z) := P_d \prod_{i=1}^{d}(z - \alpha_i(P)) = \sum_{j=0}^{d} P_j z^j$.

Root Squaring Iterations. For a positive integer ℓ write

$$P^{[\ell]} := (P_d)^{2^\ell} \prod_{i=1}^{d}(z - \alpha_i(P)^{2^\ell}) \tag{4}$$

and so $P^{[0]} = P$, $P^{[\ell]} = (P^{[\ell-1]})^{[1]}$ for $\ell \geq 1$, and

$$|\alpha_1(P^{[\ell]})| \geq |\alpha_2(P^{[\ell]})| \geq \ldots \geq |\alpha_{d-1}(P^{[\ell]})| \geq |\alpha_d(P^{[\ell]})|. \tag{5}$$

$P^{[\ell]}$ is called the ℓ-th root squaring iteration of P, aka the ℓ-th Dandelin-Lobachevsky-Gräffe (DLG) iteration of P.

Write $P^{[\ell-1]} = \sum_{j=0}^{d}(P^{[\ell-1]})_j z^j$, $P_e^{[\ell-1]} = \sum_{j=0}^{\lfloor \frac{d}{2} \rfloor}(P^{[\ell-1]})_{2j} z^j$ and $P_o^{[\ell-1]} = \sum_{j=0}^{\lfloor \frac{d-1}{2} \rfloor}(P^{[\ell-1]})_{2j+1} z^j$. $P^{[\ell]}$ can be computed iteratively based on the formula:

$$P^{[\ell]} = (-1)^d \left[(P_e^{[\ell-1]})^2 - z(P_o^{[\ell-1]})^2\right]. \tag{6}$$

The j-th coefficient $(P^{[\ell]})_j$ of $P^{[\ell]}$ is related to the coefficients of $P^{[\ell-1]}$ by:

$$(P^{[\ell]})_j = (-1)^{d-j}(P^{[\ell-1]})_j^2 + 2 \sum_{k=\max(0,2j-d)}^{j-1} (-1)^{d-j}(P^{[\ell-1]})_k (P^{[\ell-1]})_{2j-k} \tag{7}$$

***L*-bit Approximations.** For any number $c \in \mathbb{C}$, we say that $\widetilde{c} \in \mathbb{C}$ is an L-bit approximation of c if $\|\widetilde{c} - c\| \leq 2^{-L}$. For a polynomial $P \in \mathbb{C}[z]$, we say that $\widetilde{P} \in \mathbb{C}$ is an L-bit approximation of P if $\|\widetilde{P} - P\| \leq 2^{-L}$, or equivalently if $\|\widetilde{P_j} - P_j\| \leq 2^{-L}$ for all j.

Boxes, Quadri-Section, Line Segments, Bi-section. $[a - w/2, a + w/2] + \mathbf{i}[b - w/2, b + w/2]$ is the box B of width w centered at $c = a + \mathbf{i}b$. The disc $\Delta(B) := D(c, \frac{3}{4}w)$ is a *cover* of B.

Partition B into four congruent boxes (children of B), of width $w/2$ and centered at $(a \pm \frac{w}{4}) + \mathbf{i}(b \pm \frac{w}{4})$.

$\Delta(B) := D(c, w/2)$ is the minimal disc that covers a real line segment $B := [c - w/2, c + w/2]$ of width w centered at $c \in \mathbb{R}$.

Partition the segment B into two segments (children of B) of width $w/2$ centered at $(c \pm \frac{w}{4}))$.

Let $\mathcal{C}$ be a connected component of boxes (resp. real line segments); $B_{\mathcal{C}}$ is the minimal box (resp. real line segment) covering $\mathcal{C}$.

1.2 Subdivision Approach to Root-Finding

The work [1] describes an algorithm for solving a local version of the CRC problem: for an initial RoI B_0 (a box) it finds clusters of roots with pairwise distance less than ε in a small inflation of B_0. Since our CRC problem is for input polynomials in $\mathbb{Z}[z]$, one can define a RoI containing all the roots by using, for instance, the Fujiwara bound (see [4]).

Subdivision Iterations. The algorithm in [1] uses subdivision iterations or Quad-tree algorithms (inherited from [7], see also [12]), which constructs a tree rooted in the RoI B_0 whose nodes are sub-boxes of B_0. A node B is *included* only if $2B$ contains a root, *excluded* only if it contains no root. A node is *active* if it is neither included nor excluded. At the beginning of a subdivision iteration, each active node B is tested for exclusion. Then active boxes are grouped into connected components, and for each connected component $\mathcal{C}$ such that $4\Delta(B_{\mathcal{C}})$ intersect no other connected component, a root-counter is applied to $2\Delta(B_{\mathcal{C}})$. If $2\Delta(B_{\mathcal{C}})$ contains $m > 0$ roots and $\Delta(B_{\mathcal{C}})$ has radius less than ε, then $(\Delta(B_{\mathcal{C}}), m)$ is returned as a solution and the boxes of $\mathcal{C}$ are marked as included. Each remaining active node is quadrisected into its four active children, to which a new subdivision iteration is applied. Incorporation of Newton's iterations enables quadratic convergence toward clusters of radii ε.

Solving the RRI Problem. Using a root separation lower bound (e.g., of [11]), one can derive from [1] a solution of the RRI problem based on the symmetry of roots of $P \in \mathbb{Z}[z]$ along the real axis. Let disc $D(c, r)$ with $c \in \mathbb{R}$ contain m roots of P. For $m = 1$ the root in $D(c, r)$ is real. If $m \geq 1$ and $r \leq \text{sep}(P)$, then $D(c, r)$ contains a real root of multiplicity m, where $\text{sep}(P)$ is a root separation lower bound for P. For the RRI problem, the RoI B_0 is a line segment, and the subdivision tree of B_0 is built by means of segment bisection.

The $\boldsymbol{T^0}$ and $\boldsymbol{T^*}$ Tests. In the algorithm of [1], the exclusion test and root counter are based on Pellet's theorem (see [2]). For a disc $\Delta = D(c, r)$, the counting test $T^*(\Delta, P)$ returns an integer $k \in \{-1, 0, \ldots, d\}$ such that $k \geq 0$ only if P has k roots in Δ. A result $k = -1$ accounts for a failure and holds when some roots of P are close to the boundary of Δ. For a given disc Δ, the exclusion test $T^0(\Delta, P)$ returns 0 if $T^*(\Delta, P)$ returns 0 and returns -1 if $T^*(\Delta, P)$ returns a non-zero integer. The T^* of [2] takes as an input an L-bit approximation of P and with working absolute precision L performs about $\log \log d$ DLG iterations of the Taylor's shift $P(c+rz)$ of P. Write $L(\Delta, P)$ for the precision L required to carry out the T^*-test. Based on Pellet's theorem we obtain the following results.

Proposition 1 (see [2], Lemmas 4 and 5). *Let B be the box (or real line segment) centered in c with width w. The total cost in bit operations for carrying out $T^*(\Delta(B), P)$ or $T^0(\Delta(B), P)$ is bounded by*

$$\widetilde{O}(d(\log \|P\| + d \log \max(1, |c|, w) + L(\Delta, P))). \tag{8}$$

$T^*(\Delta(B), P)$ returns an integer $k \in \{-1, 0, \dots, d\}$; if $k \geq 0$ then $\Delta(B)$ contains k roots; if $k = -1$ then P has a root in $2B \setminus (1/2)B$. $T^0(\Delta(B), P)$ returns an integer $k \in \{-1, 0\}$; if $k = 0$ then P has no root in $\Delta(B)$; if $k = -1$ then P has a root in $2B$.

Bit Complexity. Proposition 1 enables one to bound the Boolean cost of exclusion tests and root-counting as well as the size of the subdivision tree and hence the cost of solving the benchmark problem in [1].

By applying subdivision iterations with an exclusion test and a root counter satisfying Proposition 1 one yields an algorithm with the same bit-complexity as the algorithm of [1], namely, $\widetilde{O}(d^2(d + \tau))$ for the benchmark problem.

Implementations. A modified version of [1] for the CRC problem has been implemented and made public within the library `Ccluster`. An implementation of the modified algorithm of [1] solving the RRI problem, called `Risolate`, is also available within `Ccluster`.

2 Root Radii Computation

We describe and analyse an algorithm for solving the RRC problem for a $P \in \mathbb{G}[z]$. Let $c \in \mathbb{G}$ and $Pc(z) := P(c+z)$, so that $r_s(P, c) = r_s(Pc)$ for all $1 \leq s \leq d$. Hence the RRC problem for a $c \neq 0$ reduces to the RRC problem for $c = 0$ at the cost of shifting the variable.

The next remark reduces the RRC problem for $c = 0$ and any $\delta > 0$ to the RRC problem for $1 + \delta = 4d$ by means of DLG iterations:

Remark 2. *Let $g = \lceil \log \dfrac{\log(4d)}{\log(1+\delta)} \rceil$, let $\rho' > 0$ such that there exist an s with:*

$$\frac{\rho'}{4d} < r_s(Pc^{[g]}) < (4d)\rho'. \tag{9}$$

Define $\rho = (\rho')^{\frac{1}{2^g}}$ and recall that $r_s(Pc^{[g]}) = r_s(P, c)^{2^g}$. Then

$$\frac{\rho}{1+\delta} < r_s(P, c) < (1+\delta)\rho. \tag{10}$$

g is in $O(\log d)$ if δ is in $d^{-O(1)}$ (for instance, $\delta \geq d^{-1}$ or $\delta \geq d^{-2}$).

Now define the RRC* problem as the RRC problem for $1 + \delta = 4d$ and $c = 0$:

RRC* problem
Given: a polynomial $P \in \mathbb{G}[z]$ of degree d, satisfying $P(0) \neq 0$
Output: d positive real numbers $\rho'_1, \dots, \rho'_d$ satisfying

$$\forall s = 1, \dots, d, \quad \frac{\rho'_s}{4d} < r_s(P) < (4d)\rho'_s. \tag{11}$$

In this setting, we assume that 0 is not a root of P and thus $P_0 \neq 0$. When 0 is a root of multiplicity m, then $r_d(P) = \ldots = r_{d-m+1}(P) = 0$ and $P_0 = \ldots = P_{m-1} = 0$, which is easily detected (since $P \in \mathbb{G}[z]$) and treated accordingly.

In Subsect. 2.1 we recall an algorithm SolveRRC* satisfying:

Proposition 3. *Algorithm* SolveRRC* *in Algorithm 1 solves the RRC* problem by involving* $O(d \log \|P\|)$ *bit operations.*

In Subsect. 2.2 we prove this proposition. In Subsect. 2.3 we present an algorithm SolveRRC satisfying:

Theorem 4. *The algorithm* SolveRRC *of Subsect. 2.3 solves the RRC problem for* $\delta = d^{-2}$ *at a Boolean cost in*

$$\widetilde{O}(d^2(d \log(|c| + 1) + \log \|P\|)). \tag{12}$$

This bound turns into $\widetilde{O}(d^2(d + \log \|P\|))$ *for* $|c| \in O(1)$ *and into* $\widetilde{O}(d^2 \log \|P\|)$ *for* $|c| = 0$.

Below we will use root radii computation as a pre-processing step for Complex Root Clustering and Real Root Isolation. For Real Root Isolation, we use SolveRRC to compute an annuli cover centered at 0. For Complex Root Clustering, we use SolveRRC to compute three annuli covers with the three centers $0, 1, \mathbf{i}$. According to our analysis of the RRC problem, the cost of its solution for $O(1)$ centers c such that $|c| \in O(1)$ is dominated by a near optimal bit-complexity of root-finding.

For $c = 0$, our algorithm has a larger bit complexity than the algorithm of [15] (see [15][Cor. 14.3] and [5][Algorithm 2]), which is in $\widetilde{O}(d^2 \log^2 d)$ when $\log \|P\| \in O(d)$. Our algorithm, however, computes d root radii at once where Schönhage's algorithm computes only a single root radius. It is not clear whether the latter algorithm can be extended to an algorithm that would solve the RRC problem within the same bit-complexity bound.

2.1 Solving the RRC* Problem

Recall that $P = \sum_{i=0}^{d} P_i z^i$ and define, for $i = 0, \ldots, d$,

$$p_i = \begin{cases} \log |P_i| & \text{if } P_i \neq 0, \\ -\infty & \text{otherwise.} \end{cases} \tag{13}$$

According to the following result, adapted from Proposition 4.2 and its proof in [12], one can solve the RRC* problem by computing the upper part of the convex hull of the set of points $\{(i, p_i) | i = 0, \ldots, d\}$ and assuming that the points $(i, -\infty)$ lie below any line in the plane.

Proposition 5. *Given an integer* s, *let* t' *and* h' *be integers s.t.*

(i′) $t' < d + 1 - s \leq t' + h' \leq d$, *and*
(ii′) $\forall 0 \leq i \leq d$, *the point* (i, p_i) *lies below the line* $((t', p_{t'}), (t' + h', p_{t'+h'}))$.

Then $\rho'_s = \left|\dfrac{P_{t'}}{P_{t'+h'}}\right|^{\frac{1}{h'}}$ *satisfies:* $\dfrac{\rho'_s}{2d} < r_s(P) < (2d)\rho'_s$.

Call CH the upper part of the convex hull of the points $\{(i, p_i) | i = 0, \dots, d\}$, and remark that for a given integer s, the integers t' and $t'+h'$ satisfying (i') and (ii') in Proposition 5 are the abscissæ of the endpoints of the segment of CH above (s, p_s). CH can be computed exactly (the P_i's are Gaussian integers). However for solving the RRC* problem, it is sufficient to compute the upper part of the convex hull of M-bit approximations of the p_i's with $M \geq 1$. For $i = 0, \dots, d$, define

$$\widetilde{p_i} := \begin{cases} M\text{-bit approximation of } p_i & \text{if } |P_i| > 1, \\ 0 & \text{if } |P_i| = 1, \\ -\infty & \text{otherwise.} \end{cases} \tag{14}$$

Let $\widetilde{CH}$ be the upper part of the convex hull of $\{(i, \widetilde{p_i}) | i = 0, \dots, d\}$ and let points $(i, -\infty)$ lie below any line in the plane. Given an index s, the following proposition bounds the slope of the edge of CH above $d+1-s$ in terms of the slope of the edge of $\widetilde{CH}$ above $d+1-s$ and M.

Proposition 6. *Given an integer s, let t, h, t', and h' be integers such that*

(i) $t < d+1-s \leq t+h \leq d$,
(ii) $\forall 0 \leq i \leq d$, *the point* $(i, \widetilde{p_i})$ *lies below the line* $((t, \widetilde{p_t}), (t+h, \widetilde{p_{t+h}}))$,
(i') $t' < d+1-s \leq t'+h' \leq d$, *and*
(ii') $\forall 0 \leq i \leq d$, *the points* (i, p_i) *lie below the line* $((t', p_{t'}), (t'+h', p_{t'+h'}))$.

Then

$$\frac{\widetilde{p_{t+h}} - \widetilde{p_t}}{h} - 2^{-M+1} \leq \frac{p_{t'+h'} - p_{t'}}{h'} \leq \frac{\widetilde{p_{t+h}} - \widetilde{p_t}}{h} + 2^{-M+1}. \tag{15}$$

For a given integer s, the existence of integers t, h, t', h' satisfying (i), (ii), (i'), (ii') follows from the existence of the convex hulls CH and $\widetilde{CH}$. We postpone the proof of Proposition 6. Remark that $2^{2^{-L}} \leq 1 + 2^{-L}$ for $L \geq 0$, apply Proposition 5, and obtain:

Corollary 7 (of Proposition 6). *Let s, t, h be as in Proposition 6. Define $\widetilde{\rho_s}'$ as* $\left|\dfrac{P_t}{P_{t+h}}\right|^{\frac{1}{h}}$. *Then*

$$\frac{\widetilde{\rho_s}'}{(2d)(1+2^{-M+1})} < r_s(P) < (2d)(1+2^{-M+1})\widetilde{\rho_s}'. \tag{16}$$

We are ready to describe our Algorithm 1, which solves the RRC* problem. In steps 1–2, 1-bit approximations $\widetilde{p_i}$ of $p_i = \log|P_i|$ are computed from P_i, for $i = 0, \dots, d$. This requires $O(d \log \|P\|)$ bit operations. In step 3 we compute the convex hull $\widetilde{CH}$ of a polygon with $d+1$ vertices $(0, \widetilde{p_0}), \dots, (d, \widetilde{p_d})$ ordered with respect to their ordinates. Using Graham's algorithm of [6], we only need

Algorithm 1. `SolveRRC`$^*(P)$

```
Input: $P \in \mathbb{G}[z]$ of degree $d$ s.t. $P(0) \neq 0$.
Output: $d$ positive real numbers $\widetilde{\rho_1}', \dots, \widetilde{\rho_d}'$.
1: for $i = 0, \dots, d$ do
2:     Compute $\widetilde{p_i}$, a 1-bit approximation of $p_i$, as defined in Eq. (14).
3: $\widetilde{CH} \leftarrow \{(i_k, \widetilde{p_{i_k}}) | k = 0, \dots, \ell\}$, the upper part of the convex hull of $\{(i, \widetilde{p_i}) | i = 0, \dots, d\}$
4: for $k = 1, \dots, \ell$ do
5:     for $s = d + 1 - i_k, \dots, d + 1 - i_{k-1}$ do
6:         $\widetilde{\rho_s}' \leftarrow |\frac{P_{i_{k-1}}}{P_{i_k}}|^{\frac{1}{i_k - i_{k-1}}}$ //double precision floating point
7: return $\widetilde{\rho_1}', \dots, \widetilde{\rho_d}'$
```

$O(d)$ arithmetic operations (additions) with numbers of magnitude $O(\log \|P\|)$. In steps 4,5,6, the $\widetilde{\rho_s}'$'s for $s = 0, \dots, d$ are computed as in Corollary 7. This task is performed with rounding to double precision arithmetic and requires $O(d)$ bit operations. Finally, $(1 + 2^{-M+1}) \leq 2$ if $M \geq 1$; thus the $\widetilde{\rho_s}'$'s in the output satisfy $\forall s = 1, \dots, d$, $\frac{\widetilde{\rho_s}'}{4d} < r_s(P) < (4d)\widetilde{\rho_s}'$, and Proposition 3 follows.

2.2 Proof of Proposition 6

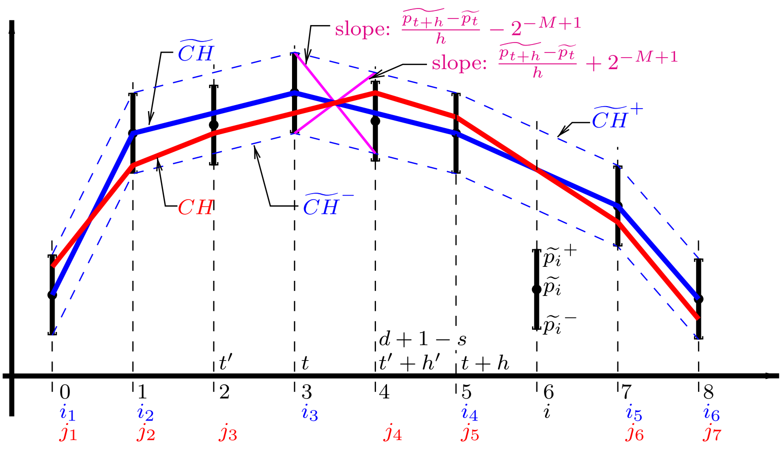

Fig. 1. The convex hulls CH and $\widetilde{CH}$ (Color figure online)

For $i = 0, \dots, d$, define $\widetilde{p_i}^+$ and $\widetilde{p_i}^-$ as

$$\widetilde{p_i}^+ = \begin{cases} \widetilde{p_i} + 2^{-M} & \text{if } |\widetilde{p_i}| > -\infty, \\ -\infty & \text{otherwise} \end{cases}, \text{ and } \widetilde{p_i}^- = \begin{cases} \widetilde{p_i} - 2^{-M} & \text{if } |\widetilde{p_i}| > -\infty, \\ -\infty & \text{otherwise} \end{cases}. \quad (17)$$

$\widetilde{CH}$ is the upper part of the convex hull of $\{(i, \widetilde{p_i})|i = 0, \ldots, d\}$. Suppose that it is the poly-line passing through $\{(i_k, \widetilde{p_{i_k}})|k = 0, \ldots, \ell\}$. It defines two poly-lines:

- $\widetilde{CH}^+$, the poly-line with vertices $\{(i_k, \widetilde{p_{i_k}}^+|k = 0, \ldots, \ell\}$, and
- $\widetilde{CH}^-$, the poly-line with vertices $\{(i_k, \widetilde{p_{i_k}}^-)|k = 0, \ldots, \ell\}$.

CH is the upper part of the convex hull of $\{(i, p_i)|i = 0, \ldots, d\}$, and suppose it is the poly-line with vertices $\{(j_k, p_{j_k})|k = 0, \ldots, \ell'\}$. For demonstration see Fig. 1 where $d = 8$, the $\widetilde{p_i}$'s are drawn with black circles, the intervals $[\widetilde{p_i}^-, \widetilde{p_i}^+]$ with bold vertical bars, $\widetilde{CH}$ with a bold blue poly-line, $\widetilde{CH}^+$ and $\widetilde{CH}^-$ with dashed blue poly-lines, and CH with a bold red line. One has:

Proposition 8. *The poly-line CH lies below the poly-line $\widetilde{CH}^+$ and above the poly-line $\widetilde{CH}^-$.*

Proof of Proposition 8: In order to prove that CH lies below $\widetilde{CH}^+$, we show that if $j_t, i_k, i_{k'}$ is a triple of integers such that (j_t, p_{j_t}) is a vertex of CH and $[(i_k, \widetilde{p_{i_k}}^+), (i_{k'}, \widetilde{p_{i_{k'}}}^+)]$ is an edge of $\widetilde{CH}^+$, then (j_t, p_{j_t}) lies on or below the line $((i_k, \widetilde{p_{i_k}}^+), (i_{k'}, \widetilde{p_{i_{k'}}}^+))$. Suppose this is not the case, *i.e.* the point (j_t, p_{j_t}) lies strictly above the line $((i_k, \widetilde{p_{i_k}}^+), (i_{k'}, \widetilde{p_{i_{k'}}}^+))$. Since $p_{j_t} \leq \widetilde{p_{j_t}}^+$, $\widetilde{p_{j_t}}^+$ lies strictly above $((i_k, \widetilde{p_{i_k}}^+), (i_{k'}, \widetilde{p_{i_{k'}}}^+))$, thus $\widetilde{p_{j_t}}$ lies strictly above $((i_k, \widetilde{p_{i_k}}), (i_{k'}, \widetilde{p_{i_{k'}}}))$ and $\widetilde{CH}$ is not the convex hull of $\{(i, \widetilde{p_i})|i = 0, \ldots, d\}$, which is a contradiction.

In order to show that $\widetilde{CH}^-$ lies below CH, we show that for a given triple of integers $i_t, j_k, j_{k'}$ such that $(i_t, \widetilde{p_{i_t}}^-)$ is a vertex of $\widetilde{CH}^-$ and $[(j_k, p_{j_k}), (j_{k'}, p_{j_{k'}})]$ is an edge of CH, the point $(i_t, \widetilde{p_{i_t}}^-)$ lies on or below the line $((j_k, p_{j_k}), (j_{k'}, p_{j_{k'}}))$. Suppose it is not the case. Since $p_{i_t} \geq \widetilde{p_{i_t}}^-$, the point p_{i_t} lies strictly above the line passing through $((j_k, p_{j_k}), (j_{k'}, p_{j_{k'}}))$ and CH is not the convex hull of $\{(i, p_i)|i = 0, \ldots, d\}$, which is a contradiction. □

Proof of Proposition 6: Given the integer s, let t, h, t', h' be integers such that conditions $(i), (ii), (i')$ and (ii') hold.

By virtue of (i') and (ii'), $](t', p_{t'}), (t' + h', p_{t'+h'})]$ is the edge of CH whose orthogonal projection onto the abscissa axis contains $d + 1 - s$, and $\dfrac{p_{t'+h'} - p_{t'}}{h'}$ is the slope of that edge.

By virtue of (i) and (ii), $](t, \widetilde{p_t}), (t + h, \widetilde{p_{t+h}})]$ is the edge of $\widetilde{CH}$ whose orthogonal projection onto the abscissa axis contains $d + 1 - s$. Consider the two segments $](t, \widetilde{p_t}^-), (t + h, \widetilde{p_{t+h}}^-)]$ and $](t, \widetilde{p_t}^+), (t + h, \widetilde{p_{t+h}}^+)]$ that are the edges of $\widetilde{CH}^-$ and $\widetilde{CH}^+$, respectively, whose orthogonal projections onto the abscissa axis also contain $d + 1 - s$.

From Proposition 8 CH is a poly-line enclosed by $\widetilde{CH}^-$ and $\widetilde{CH}^+$ and since the first coordinates of its vertices are integers, its slope $\dfrac{p_{t'+h'} - p_{t'}}{h'}$ above $d + 1 - s$ is bounded below by $\dfrac{\widetilde{p_{t+h}} - \widetilde{p_t}}{h} - 2^{-M+1}$ and above by $\dfrac{\widetilde{p_{t+h}} - \widetilde{p_t}}{h} + 2^{-M+1}$, which proves Proposition 6. See Fig. 1 for an illustration. □

Algorithm 2. `SolveRRC`(P, c, δ)

Input: $P \in \mathbb{Z}[z]$ of degree d, a center $c \in \mathbb{G}$ and a relative precision $\delta > 0$.
Output: d positive real numbers $\rho_{c,s}, \dots, \rho_{c,d}$ solving task S.

```
1: $g \leftarrow \lceil \log \frac{\log(4d)}{\log(1+\delta)} \rceil$
2: compute $Pc^{[g]}$
3: $\rho'_1, \dots, \rho'_d \leftarrow$ SolveRRC$^*(Pc^{[g]})$
4: for  $s = 0, \dots, d$ do
5:     $\rho_{c,s} \leftarrow (\rho'_s)^{1/2^g}$
6: return $\rho_{c,1}, \dots, \rho_{c,d}$
```

2.3 Solving the RRC Problem

Using Remark 2, we define in Algorithm 2 the algorithm `SolveRRC`. To estimate the cost at steps 2 and 3, let $\mathcal{M} : \mathbb{N} \rightarrow \mathbb{N}$ be such that two polynomials of degree at most d and bit-size at most τ can be multiplied by using $O(\mathcal{M}(d\tau))$ bit operations. Recall the following:

1. computing Pc requires $O(\mathcal{M}(d^2 \log d + d^2 \log(|c|+1) + d \log \|P\|))$ bit operations,
2. $\|Pc\| \leq \|P\|(|c|+1)^d$,
3. computing $\|Pc^{[i]}\|$ from $\|Pc^{[i-1]}\|$ requires $O(\mathcal{M}(d \log \|Pc^{[i-1]}\|))$ bit operations,
4. $\|Pc^{[i]}\| \leq (d+1)(\|Pc^{[i-1]}\|)^2 \leq \dots \leq (d+1)^{2^i}(\|Pc\|)^{2^i}$.

For 1 and 2, see for instance [16][Theorem 2.4] and [16][Lemma 2.1]). 3 and 4 are derived from Eqs. (6) and (7), respectively. From 2, 4 and $g \in O(\log d)$ one obtains

$$\log \|Pc^{[g]}\| \in O(d \log(d+1) + d \log \|P\| + d^2 \log(|c|+1)), \tag{18}$$

thus performing g DLG iterations for Pc involves

$$O(g\mathcal{M}(d \log \|Pc^{[g]}\|)) = O(\log d \mathcal{M}(d(d \log(d+1) + d \log \|P\| + d^2 \log(|c|+1)))) \tag{19}$$

bit operations; this dominates the cost of step 2. Due to Schönhage-Strassen or Harvey-van der Hoeven multiplication, $\mathcal{M}(n) \in \widetilde{O}(n)$, and so step 2 involves

$$\widetilde{O}(d^2 \log \|P\| + d^3 \log(|c|+1)) \tag{20}$$

bit operations. Step 3 involves $O(d \log \|Pc^{[g]}\|)$ bit operations, the cost of the **for** loop in steps 4–5 is dominated by the cost of step 2, and we complete the proof of Theorem 4.

2.4 Implementation Details

The exact computation of $Pc^{[g]}$ can involve numbers of very large bit-size (see Eq. (18)), and the key point for the practical efficiency of our implementation of Algorithm 2 is to avoid this. Instead, we use *ball arithmetic*, *i.e.* arbitrary precision floating point arithmetic with absolute error bounds, implemented for instance in the `C` library `arb` (see [9]).

3 Real Root Isolation

In this section, we approximate the root distances to 0 in order to improve in practice subdivision approaches to real root isolation, and in particular the one described in Subsect. 1.2. Let us define the notion of annuli cover of the roots of $P \in \mathbb{Z}[z]$.

Definition 1. *A set $\mathcal{A}_c$ of disjoint concentric annuli centered in c is an* annuli cover *of the roots of P of degree d if*

1. *$\forall A \in \mathcal{A}_c$, there are integers $t(A)$ and $h(A)$ such that*

$$\alpha_{t(A)}(P,c), \alpha_{t(A)+1}(P,c), \ldots, \alpha_{t(A)+h(A)}(P,c) \in A, \tag{21}$$

2. *$\forall i \in \{1, \ldots, d\}$, there is an $A \in \mathcal{A}_c$ such that $\alpha_i(P,c) \in A$.*

For an annulus $A \in \mathcal{A}_c$, $\underline{r}(A)$ and $\overline{r}(A)$ are the interior and exterior radii of A, respectively. Write $s_+(A) := P(c + \underline{r}(A))P(c + \overline{r}(A))$ and $s_-(A) := P(c - \underline{r}(A))P(c - \overline{r}(A))$.

Given an annuli cover $\mathcal{A}_0$ centered at 0, we can skip many calls for exclusion and root-counting based on the following:

Remark 9. *Let $A \in \mathcal{A}_0$ such that $\underline{r}(A) > 0$.*

1. *If $h(A) = 0$ and $s_+(A) > 0$ (resp. $s_-(A) > 0$), $A \cap \mathbb{R}_+$ (resp. $A \cap \mathbb{R}_-$) contains no real root of P.*
2. *If $h(A) = 0$ and $s_+(A) < 0$ (resp. $s_-(A) < 0$), $A \cap \mathbb{R}_+$ (resp. $A \cap \mathbb{R}_-$) contains one real root of P.*
3. *If $h(A) > 0$ and $s_+(A) < 0$ (resp. $s_-(A) < 0$), $A \cap \mathbb{R}_+$ (resp. $A \cap \mathbb{R}_-$) contains at least one real root of P.*

In Subsects. 3.1 and 3.2 we describe our exclusion test and root counter based on Remark 9. In Subsect. 3.3 we describe our algorithm solving the RRI problem. In Subsect. 3.4 we present the results of our numerical tests.

3.1 Annuli Cover and Exclusion Test

Let B be a real line segment that does not contain 0, let $\mathcal{A}$ be an annuli cover centered in 0, and let $A \in \mathcal{A}$. Define that:

- s_B, the sign of B, is -1 (resp. 1) if $B < 0$ (resp. $B > 0$),
- $\mathbb{R}s_B(A)$ is $A \cap \mathbb{R}_-$ if $s_B < 0$, and is $A \cap \mathbb{R}_+$ otherwise,
- $ss_B(A)$ is $s_-(A)$ if $s_B < 0$, and is $s_+(A)$ otherwise,
- $n(\mathcal{A}, B)$ is the number of annuli $A \in \mathcal{A}$ s.t. $A \cap B \neq \emptyset$,

Algorithm 3. $C^0_{\mathbb{R}}(B, P, \mathcal{A})$

```
Input: A polynomial $P \in \mathbb{Z}[z]$ of degree $d$, a segment $B$ of $\mathbb{R}$, an annuli cover $\mathcal{A}$ of
    the roots of $P$ centered in 0. Assume $0 \notin B$.
Output: an integer in $\{-1, 0\}$; if 0 then $P$ has no real root in $B$; if $-1$ then there is
    a root in $2B$.
Compute $n(\mathcal{A}, B)$, $n_0(\mathcal{A}, B)$ and $n_{\geq 1}(\mathcal{A}, B)$
if $n(\mathcal{A}, B) = n_0(\mathcal{A}, B)$ then
    return 0
if $n_{\geq 1}(\mathcal{A}, B) >= 1$ then
    return $-1$
return $T^0(\Delta(B), P)$
```

- $n_0(\mathcal{A}, B)$ is the number of annuli $A \in \mathcal{A}$ s.t.

$$(A \cap B \neq \emptyset) \wedge (h(A) = 0) \wedge ss_B(A) > 0, \tag{22}$$

- $n_{\geq 1}(\mathcal{A}, B)$: the number of annuli $A \in \mathcal{A}$ s.t.

$$(A \cap B \neq \emptyset) \wedge (h(A) \geq 0) \wedge ss_B(A) < 0 \wedge \mathbb{R}s_B(A) \subseteq 2B. \tag{23}$$

By virtue of Remark 9, if $n(\mathcal{A}, B) = n_0(\mathcal{A}, B)$, then all the annuli intersecting B contain no root, thus B contains no root. If $n_{\geq 1}(\mathcal{A}, B) \geq 1$ then $2B$ contains at least one real root.

Our exclusion test $C^0_{\mathbb{R}}$ is described in Algorithm 3. For computing $n(\mathcal{A}, B)$, $n_0(\mathcal{A}, B)$ and $n_{\geq 1}(\mathcal{A}, B)$ in Step 1, we use double precision interval arithmetic, hence Step 1 involves $O(d)$ bit operations. This implies

Proposition 10. *Let B be a real line segment with $0 \notin B$, and let $\mathcal{A}$ be an annuli cover of the roots of P centered in 0. The cost of carrying out $C^0_{\mathbb{R}}(B, P, \mathcal{A})$ is bounded by the cost of carrying out $T^0(\Delta(B), P)$.*

$C^0_{\mathbb{R}}(B, P, \mathcal{A})$ returns an integer k in $\{-1, 0\}$. If $k = 0$, then P has no real roots in B. If $k = -1$, then P has a root in $2B$.

3.2 Annuli Cover and Root Counter

In order to describe our root counter, we define:

- $n_1(\mathcal{A}, B)$: the number of annuli $A \in \mathcal{A}$ s.t. :

$$(A \cap B \neq \emptyset) \wedge (h(A) = 0) \wedge (ss_B(A) < 0) \wedge (\mathbb{R}s_B(A) \subseteq B), \tag{24}$$

- $n'_{\geq 1}(\mathcal{A}, B)$: the number of annuli $A \in \mathcal{A}$ s.t.:

$$(\ A \cap B \neq \emptyset) \wedge (h(A) \geq 0) \wedge ss_B(A) < 0 \wedge (\mathbb{R}s_B(A) \subseteq 2B \setminus (1/2)B\ ). \tag{25}$$

By virtue of Remark 9, if $n(\mathcal{A}, B) = n_0(\mathcal{A}, B) + n_1(\mathcal{A}, B)$, B contains exactly $n_1(\mathcal{A}, B)$ real roots. If $n'_{\geq 1}(\mathcal{A}, B) \geq 1$ then P has at least one real root in $2B \setminus (1/2)B$.

Our root counter is described in Algorithm 4. We use double precision interval arithmetic for computing $n(\mathcal{A}, B)$, $n_0(\mathcal{A}, B)$, $n_1(\mathcal{A}, B)$ and $n'_{\geq 1}(\mathcal{A}, B)$ in Step 1, thus Step 1 involves $O(d)$ bit operations.

Algorithm 4. $C^*_{\mathbb{R}}(B, P, \mathcal{A})$

Input: A polynomial $P \in \mathbb{Z}[z]$ of degree d, a segment B of $\mathbb{R}$, an annuli cover $\mathcal{A}$ of the roots of P centered in 0. Assume $0 \notin 2B$.

Output: an integer k in $\{-1, 0, 1, \ldots, d\}$; if $k \geq 0$ then P has k roots in B; if -1 then there is a root in $2B \setminus (1/2)B$.

```
Compute $n(\mathcal{A}, B)$, $n_0(\mathcal{A}, B)$, $n_1(\mathcal{A}, B)$ and $n'_{\geq 1}(\mathcal{A}, B)$
if $n(\mathcal{A}, B) = n_0(\mathcal{A}, B) + n_1(\mathcal{A}, B)$ then
    return $n_1(\mathcal{A}, B)$
if $n'_{\geq 1}(\mathcal{A}, B) \geq 1$ then
    return $-1$
return $T^*(\Delta(B), P)$
```

Proposition 11. *Let B be a real line segment with $0 \notin B$ and let $\mathcal{A}$ be an annuli cover of the roots of P centered in 0. The cost of carrying out $C^*_{\mathbb{R}}(B, P, \mathcal{A})$ is bounded by the cost of carrying out $T^*(\Delta(B), P)$.*

$C^*_{\mathbb{R}}(B, P, \mathcal{A})$ returns an integer k in $\{-1, 0, \ldots, d\}$. If $k \geq 0$, then P has k roots in $\Delta(B)$. If $k = -1$, then P has a root in $2B \setminus (1/2)B$.

3.3 Annuli Cover and the RRI Problem

Consider the following procedure.

Stage 1: Compute $\mathcal{A}_0$ by calling `SolveRRC`$(P, 0, d^{-2})$.
Stage 2: Apply the subdivision procedure of Subsect. 1.2 while using $C^0_{\mathbb{R}}(B, P, \mathcal{A}_0)$ (resp. $C^*_{\mathbb{R}}(B, P, \mathcal{A}_0)$) as an exclusion test (resp. root counter) for real line segment B of the subdivision tree. In the verification step of Newton iterations, use the T^*-test of [2].

At Stage 1, we obtain $\mathcal{A}_0$ by computing the connected components made up of the concentric annuli defined by the output of `SolveRRC`$(P, 0, d^{-2})$.

By virtue of Theorem 4 and Propositions 10 and 11, this procedure solves the RRI problem, and its bit-complexity is bounded by the bit-complexity of the algorithm described in [1], thus it is near optimal for the benchmark problem.

3.4 Experimental Results

The procedure given in Subsect. 3.3 has been implemented within the library `Ccluster`; we call this implementation `RisolateR`. Comparison of `RisolateR` with `Risolate` reveals practical improvement due to using our root radii algorithms in subdivision process. We also compare `RisolateR` with the subdivision algorithm of [14] whose implementation `ANewDsc` is described in [10] and is currently the user's choice for real root isolation.

Test Polynomials. We consider the following polynomials.

The Bernoulli polynomial of degree d is $B_d(z) = \sum_{k=0}^{d} \binom{d}{k} b_{d-k} z^k$ where the b_i's are the Bernoulli numbers.

The Wilkinson polynomial of degree d is $W_d(z) = \prod_{i=1}^{d}(z - i)$.

For integers $n > 0$, we define polynomials with $(2n + 1) \times (2n + 1)$ roots on the nodes of a regular grid centered at 0 as

$$P_{(2n+1)\times(2n+1)}(z) = \prod_{-n \leq a,b \leq n} (z - a + \mathbf{i}b). \tag{26}$$

The Mignotte polynomial of degree d and bitsize τ is $M_{d,\tau}(z) = z^d - 2(2^{\frac{\tau}{2}-1}z - 1)^2$.

We also consider dense polynomials of degree d with coefficients randomly chosen within $[-2^{\tau-1}, 2^{\tau-1}]$ (under the uniform distribution).

Results. In our test polynomials with several degrees/bit-sizes, we used `Risolate`, `RisolateR` and `ANewDsc` to solve the RRI problem. Our non-random examples have only simple roots and for those examples `ANewDsc` is called with option `-S 1` to avoid testing input polynomial for being square-free.

Times are sequential times in seconds on a `Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz` machine with Linux. We report in Table 2:

- d, τ and $d_{\mathbb{R}}$, that is, the degree, the bit-size and the number of real roots, respectively,
- t_1 (resp. t_2), the running time of `Risolate` (resp. `RisolateR`),
- n_1 (resp. n_2), the number of T^0-tests in `Risolate` (resp. `RisolateR`),
- n'_1 (resp. n'_2), the number of T^*-tests in `Risolate` (resp. `RisolateR`),
- t_3, the time required to compute the annuli cover in `RisolateR`,
- t_4, the running time in second of `ANewDsc`.

For random polynomials, we display averages over 10 examples of those values. We also display σ_1, σ_2, and σ_4, the standard deviation of running time of `Risolate`, `RisolateR` and `ANewDsc`, respectively.

Compare columns n_1, n'_1 and n_2, n'_2 in Table 2: using the annuli cover both in exclusion tests and root counter reduces dramatically the number of Pellet's tests performed in the subdivision process, and significantly decreases the running time (see column t_2/t_1). In the cases where the ratio τ/d is low `RisolateR` spent most of the time on solving the RRC problem (see column t_3/t_2). Finally, `ANewDsc` remains faster than `RisolateR` for polynomials having a few real roots or a low bit-size, whereas this trend seems to reverse when the ratios of the number of real roots and/or bit-size over the degree increase (see columns t_2 and t_4). Mignotte polynomials of even degree have four real roots among which two are separated by a distance way less than the relative size of d^{-2}, the relative size of annuli in the computed annuli cover. In such cases, the knowledge of root radii enables no significant improvement because subdivision solvers spend most of their running time on performing Newton's iterations that converge to the cluster of two close roots, and then on separating the two roots.

Table 2. Runs of `Risolate`, `RisolateR` and `ANewDsc` on our test polynomials

			`Risolate`		`RisolateR`				`ANewDsc`
d	τ	$d_{\mathbb{R}}$	t_1 (σ_1)	n_1, n'_1	t_2 (σ_2)	n_2, n'_2	t_3/t_2 (%)	t_2/t_1 (%)	t_4 (σ_4)
10 monic random dense polynomials per degree/bit-size									
256	8192	6.00	3.02 (1.13)	128.,80.2	.374 (.080)	4.40,25.3	16.3	12.3	.784 (1.73)
256	16384	7.80	5.09 (1.99)	183.,122.	.499 (.132)	2.60,22.9	22.2	9.80	2.76 (5.19)
256	32768	7.40	7.59 (3.20)	172.,125.	.442 (.174)	4.40,27.4	16.3	5.82	1.18 (.600)
256	65536	7.00	10.7 (6.33)	170.,140.	.480 (.160)	4.30,25.4	10.4	4.46	1.91 (1.18)
391	8192	7.20	8.87 (2.99)	157.,107.	1.12 (.310)	4.60,26.0	15.0	12.6	3.29 (5.42)
391	16384	8.40	10.1 (4.12)	186.,116.	1.39 (.575)	6.20,28.9	15.3	13.7	10.2 (19.8)
391	32768	8.60	18.6 (6.98)	202.,155.	1.38 (.528)	4.00,29.0	14.5	7.41	1.67 (.750)
391	65536	7.60	23.9 (13.9)	178.,137.	1.88 (1.17)	3.90,33.6	18.5	7.86	13.9 (18.9)
512	8192	6.60	31.1 (18.5)	158.,104.	3.68 (4.72)	6.00,25.9	12.4	11.8	1.26 (1.03)
512	16384	5.20	41.1 (20.1)	152.,106.	5.00 (4.63)	6.50,25.8	5.37	12.1	1.70 (2.17)
512	32768	6.00	56.7 (28.1)	167.,122.	2.00 (.596)	4.40,28.4	18.1	3.53	5.95 (7.61)
512	65536	6.60	86.5 (34.2)	180.,137.	4.84 (3.67)	5.90,32.7	5.19	5.60	60.1 (118.)
Bernoulli polynomials									
256	1056	64	1.13	292, 82	0.08	12, 3	54.2	7.77	0.20
391	1809	95	2.66	460, 145	0.30	12, 2	76.1	11.2	1.09
512	2590	124	6.15	528, 144	0.38	14, 3	65.9	6.30	1.58
791	4434	187	16.3	892, 264	2.39	20, 1	85.0	14.6	9.92
1024	6138	244	56.3	1048, 283	2.42	12, 3	76.5	4.30	14.9
Wilkinson polynomials									
256	1690	256	3.63	1030, 283	0.17	0, 10	41.1	4.90	1.57
391	2815	391	17.6	1802, 541	0.68	0, 10	51.7	3.88	5.69
512	3882	512	25.9	2058, 533	1.04	0, 11	46.9	4.01	27.1
791	6488	791	165.	3698, 1110	7.04	0, 11	57.1	4.26	158.
1024	8777	1024	265.	4114, 1049	8.38	0, 12	51.2	3.15	309.
Polynomials with roots on a regular grid									
289	741	17	0.40	86, 30	0.13	0, 16	81.7	34.4	0.09
441	1264	21	0.91	106, 36	0.21	0, 20	77.3	23.4	0.39
625	1948	25	1.59	118, 39	0.92	0, 24	89.1	58.0	0.80
841	2800	29	3.30	154, 51	1.67	0, 28	87.4	50.7	2.56
1089	3828	33	8.06	166, 55	2.20	0, 32	76.4	27.3	4.49
Mignotte polynomials									
512	256	4	1.57	34, 15	1.67	2, 12	16.5	106.	0.76
512	512	4	3.07	34, 15	4.81	2, 14	5.70	156.	1.90
512	1024	4	5.91	34, 15	5.96	2, 10	4.13	100.	5.28
512	2048	4	13.8	34, 15	13.2	2, 9	2.42	95.3	14.1
512	4096	4	29.7	50, 17	30.8	2, 6	.753	103.	36.0

4 Complex Root Clustering

In this section, by approximating the root distances from three centers, namely 0, 1 and **i** we improve practical performance of subdivision algorithms for complex root clustering.

Using three annuli covers $\mathcal{A}_0, \mathcal{A}_1$ and $\mathcal{A}_{\mathbf{i}}$ of the roots of P, one can compute a set $\mathcal{D}$ of $O(d^2)$ complex discs containing all the roots of P, and then skip expensive Pellet-based exclusion tests of the boxes that do not intersect the union of these discs.

In Subsect. 4.1 we describe an exclusion test using the set $\mathcal{D}$ of discs containing the roots of P, and in Subsect. 4.2 we present a procedure solving the CRC with near optimal bit complexity. In Subsect. 4.3 we show experimental results.

4.1 Annuli Cover and Exclusion Test

Let $\mathcal{D}$ be a set of $O(d^2)$ complex discs covering all the roots of P, *i.e.* any root of P is in at least one disc in $\mathcal{D}$. A box B such that $B \cap \mathcal{D} = \emptyset$ cannot contain a root of P.

We define an exclusion test based on the above consideration, called $C^0_{\mathbb{C}}$-test and described in Algorithm 5. For a box B having a nonempty intersection with the real line, the number $n_{\geq 1}(\mathcal{A}_0, B)$ of annuli intersecting B and containing at least one real root in $B \cap \mathbb{R}$ is used to save some T^0-tests.

Algorithm 5. $C^0_{\mathbb{C}}(B, P, \mathcal{D}, \mathcal{A}_0)$

Input: A polynomial $P \in \mathbb{Z}[z]$ of degree d, a box B of $\mathbb{C}$, a set $\mathcal{D}$ of $O(d^2)$ complex discs covering all the roots of P, an annuli cover $\mathcal{A}_0$ centered in 0

Output: an integer in $\{-1, 0\}$; if 0 then P has no real root in B; if -1 then there is a root in $2B$.

```
Compute the number $n$ of discs in $\mathcal{D}$ having nonempty intersection with $B$.
if $n = 0$ then
    return 0
if $B \cap \mathbb{R} \neq \emptyset$ then
    Compute $n_{\geq 1}(\mathcal{A}_0, B)$
    if $n_{\geq 1}(\mathcal{A}_0, B) >= 1$ then
        return $-1$
return $T^0(\Delta(B), P)$
```

Proposition 12. *Let $\mathcal{D}$ contain $O(d^2)$ discs covering the roots of P and let $\mathcal{A}_0$ be an annuli cover of the roots of P centered in 0. The cost of performing $C^0_{\mathbb{C}}(B, P, \mathcal{D}, \mathcal{A}_0)$ is bounded by the cost of performing $T^0(\Delta(B), P)$.*

$C^0_{\mathbb{C}}(B, P, \mathcal{D}, \mathcal{A}_0)$ returns an integer k in $\{-1, 0\}$. If $k = 0$, then P has no root in B. If $k = -1$, then P has a root in $2B$.

4.2 Annuli Cover and the CRC Problem

Consider the following procedure.

Stage 1: For $c = 0, 1, \mathbf{i}$, compute $\mathcal{A}_c$ by calling `SolveRRC`(P, c, d^{-2}).
Stage 2: Use $\mathcal{A}_0$, $\mathcal{A}_1$ and $\mathcal{A}_{\mathbf{i}}$ to compute a set $\mathcal{D}$ of at most $2d^2$ discs covering all roots of P.
Stage 3: Apply the Complex Root Clustering Algorithm of Subsect. 1.2 but let it apply $C^0_{\mathbb{C}}(B, P, \mathcal{D}, \mathcal{A}_0)$ instead of $T^0(\Delta(B), P)$ as the exclusion test for

boxes B of the subdivision tree. In the verification step of Newton iterations, use the T^*-test of [2].

In Stage 1, for $c = 0, 1, \mathbf{i}$, $\mathcal{A}_c$ is obtained by computing the connected components of the concentric annnuli defined by the output of SolveRRC(P, c, d^{-2}). According to Theorem 4, Stage 1 involves $\widetilde{O}(d^2(d + \log \|P\|_\infty))$ bit operations.

In Stage 2, $\mathcal{D}$ is computed as follows: using double precision floating point arithmetic with correct rounding, first compute complex discs containing all possible intersections of an annulus in $\mathcal{A}_0$ with an annulus in $\mathcal{A}_1$, and obtain a set $\mathcal{D}$ of at most $2d^2$ complex discs containing all roots of P. Then, for each disc Δ in $\mathcal{D}$ check if Δ and its complex conjugate $\overline{\Delta}$ have a nonempty intersection with at least one annulus of $\mathcal{A}_\mathbf{i}$, and remove Δ from $\mathcal{D}$ if it does not. This step has cost in $O(d^3)$.

By virtue of Proposition 12, the cost of performing Stage 3 is bounded by the cost of performing the algorithm described in Subsect. 1.2. This procedure solves the CRC problem and supports near optimal complexity for the benchmark problem.

4.3 Experimental Results

The procedure of Subsect. 4.2 is implemented within `Ccluster`; below we call this implementation `CclusterR` and present experimental results that highlight practical improvement due to using our root radii algorithm in subdivision.

We used `Ccluster` and `CclusterR` with input value $\varepsilon = 2^{-53}$ to find clusters of size at most ε. We also used `MPSolve-3.2.1`, with options `-as -Ga -o16 -j1` to find approximations with 16 correct digits of the roots.

For our test polynomials (see 3.4) we report in Table 3:

- d and τ denoting the degree and the bit-size, respectively,
- t_1 (resp. t_2), the running time of `Ccluster` (resp. `CclusterR`),
- n_1 (resp. n_2), the number of T^0-tests in `Ccluster` (resp. `CclusterR`),
- t_3, the time for computing the three annuli covers in `CclusterR`, - t_4, the running time of `MPSolve` in seconds.

For random polynomials, we show averages over 10 examples of those values. We also show σ_1, σ_2, and σ_4, the standard deviations of the running times of `Ccluster`, `CclusterR` and `MPSolve`. For the real root isolator presented in Sect. 3, using root radii enables significant saving of Pellet-based exclusion tests in the subdivision process (compare columns n_1 and n_2) and yields a speed-up factor about 3 for our examples (see column t_2/t_1). This speed-up increases as the number of real roots increases (see, e.g., Wilkinson polynomials) because some exclusion tests for boxes B containing the real line are avoided when $2B$ contains at least one root which we can see from the number $n_{\geq 1}(\mathcal{A}_0, B)$ computed in the $C^0_{\mathbb{C}}$ test. The time spent for computing the three annuli covers remains low compared to the running time of `CclusterR` (see column t_3/t_2). `MPSolve` remains the user's choice for approximating all complex roots.

Table 3. Runs of `Ccluster`, `CclusterR` and `MPSolve` on our test polynomials

		`Ccluster`		`CclusterR`				`MPSolve`
d	τ	t_1 (σ_1)	n_1	t_2 (σ_2)	n_2	t_3/t_2 (%)	t_2/t_1 (%)	t_4 (σ_4)
10 monic random dense polynomials per degree								
128	128	4.43 (.760)	2598.	1.46 (.235)	463.	7.81	33.1	.031 (.003)
191	191	13.5 (1.82)	3846.	4.40 (.528)	694.	4.20	32.6	.063 (.007)
256	256	23.7 (2.52)	4888.	7.87 (.672)	909.	7.04	33.2	.106 (.013)
391	391	70.9 (9.23)	7494.	22.5 (1.95)	1460.	3.67	31.7	.209 (.037)
512	512	154. (17.9)	9996.	46.1 (6.00)	1840.	7.08	29.9	.392 (.102)
Bernoulli polynomials								
128	410	3.86	2954	1.25	548	7.48	32.3	0.07
191	689	12.2	4026	4.51	942	8.07	36.8	0.16
256	1056	24.7	5950	10.1	1253	6.57	41.1	0.39
391	1809	75.1	8322	27.4	1907	16.2	36.5	0.97
512	2590	133.	11738	49.9	2645	12.7	37.5	2.32
Wilkinson polynomials								
128	721	8.43	3786	1.09	14	14.4	12.9	0.17
191	1183	25.4	5916	2.99	18	27.9	11.7	0.51
256	1690	50.7	7500	6.34	18	21.7	12.4	1.17
391	2815	201.	12780	23.1	22	36.2	11.4	4.30
512	3882	379.	14994	51.3	22	35.6	13.5	9.33
Polynomials with roots on a regular grid								
169	369	7.37	3072	1.99	592	4.03	27.1	0.05
289	741	27.1	5864	10.2	1573	3.18	37.9	0.13
441	1264	81.4	9976	24.4	1713	4.28	29.9	0.56
625	1948	228.	15560	70.2	2508	15.0	30.7	1.16
841	2800	493.	19664	169.	4294	5.75	34.2	3.84
Mignotte polynomials								
512	256	88.8	9304	28.3	1611	11.0	31.8	0.76
512	512	88.3	9304	29.3	1570	9.20	33.1	0.79
512	1024	101.	9304	32.1	1647	8.62	31.7	0.91
512	2048	106.	9304	33.4	1990	7.50	31.2	1.12
512	4096	102.	9304	50.1	3593	4.88	49.0	1.10

References

1. 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, ISSAC 2016, pp. 71–78. ACM, New York (2016)
2. Becker, R., Sagraloff, M., Sharma, V., Yap, C.: A near-optimal subdivision algorithm for complex root isolation based on Pellet test and Newton iteration. J. Symb. Comput. **86**, 51–96 (2018)
3. Bini, D.A., Robol, L.: Solving secular and polynomial equations: a multiprecision algorithm. J. Comput. Appl. Math. **272**, 276–292 (2014)
4. Fujiwara, M.: Über die obere schranke des absoluten betrages der wurzeln einer algebraischen gleichung. Tohoku Math. J. First Series **10**, 167–171 (1916)
5. Gourdon, X.: Algorithmique du theoreme fondamental de l'algebre. Research Report RR-1852, INRIA (1993). https://hal.inria.fr/inria-00074820
6. Graham, R.L., Yao, F.F.: Finding the convex hull of a simple polygon. J. Algorithms **4**(4), 324–331 (1983)
7. Henrici, P., Gargantini, I.: Uniformly convergent algorithms for the simultaneous approximation of all zeros of a polynomial. In: Constructive Aspects of the Fundamental Theorem of Algebra, pp. 77–113. Wiley-Interscience New York (1969)
8. Imbach, R., Pan, V.Y., Yap, C.: Implementation of a near-optimal complex root clustering algorithm. Math. Softw. - ICMS **2018**, 235–244 (2018)

9. Johansson, F.: Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. IEEE Trans. Comput. **66**, 1281–1292 (2017)
10. Kobel, A., Rouillier, F., Sagraloff, M.: Computing real roots of real polynomials ... and now for real! In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, pp. 303–310. ACM, New York (2016)
11. Mignotte, M.: On the distance between the roots of a polynomial. Appl. Algebra Eng. Commun. Comput. **6**(6), 327–332 (1995)
12. Pan, V.Y.: Approximating complex polynomial zeros: modified Weyl's quadtree construction and improved Newton's iteration. J. Complex. **16**(1), 213–264 (2000)
13. Pan, V.Y.: Univariate polynomials: nearly optimal algorithms for numerical factorization and root-finding. J. Symb. Comput. **33**(5), 701–733 (2002)
14. Sagraloff, M., Mehlhorn, K.: Computing real roots of real polynomials. J. Symb. Comput. **73**, 46–86 (2016)
15. Schönhage, A.: The fundamental theorem of algebra in terms of computational complexity. Preliminary report, University of Tübingen, Germany (1982)
16. Von Zur Gathen, J., Gerhard, J.: Fast algorithms for Taylor shifts and certain difference equations. In: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, pp. 40–47. ACM (1997)