New Progress in Univariate Polynomial Root Finding

Rémi Imbach*
remi.imbach@nyu.edu
New York University

Victor Y. Pan†
victor.pan@lehman.cuny.edu
City University of New York

ABSTRACT

The recent advanced sub-division algorithm is nearly optimal for the approximation of the roots of a dense polynomial given in monomial basis; moreover, it works locally and slightly outperforms the user's choice MPSolve when the initial region of interest contains a small number of roots. Its basic and bottleneck block is counting the roots in a given disc on the complex plain based on Pellet's theorem, which requires the coefficients of the polynomial and expensive shift of the variable. We implement a novel method for both root-counting and exclusion test, which is faster, avoids the above requirements, and remains efficient for sparse input polynomials. It relies on approximation of the power sums of the roots lying in the disc rather than on Pellet's theorem. Such approximation was used by Schönhage in 1982 for the different task of deflation of a factor of a polynomial provided that the boundary circle of the disc is sufficiently well isolated from the roots. We implement a faster version of root-counting and exclusion test where we do not verify isolation and significantly improve performance of subdivision algorithms, particularly strongly in the case of sparse inputs. We present our implementation as heuristic and cite some relevant results on its formal support presented elsewhere.

KEYWORDS

Polynomial root finding, Subdivision, Root counting

ACM Reference Format:
Rémi Imbach and Victor Y. Pan. 2020. New Progress in Univariate Polynomial Root Finding. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '20), July 20–23, 2020, Kalamata, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3373207.3404063

1 INTRODUCTION

We seek complex roots of a degree d univariate polynomial p with real or complex coefficients. For a while the user choice for this problem has been the package `MPsolve` based on Erhlich-Aberth (simultaneous Newton-like) iterations. Their empirical global convergence (right from the start) is very fast, but its formal support is a long-known challenge, and the iterations approximate the roots in a fixed region of interest (ROI) about as slow as all complex roots.

In contrast, for the known algorithms subdividing a ROI, e.g., box, the cost of root-finding in a ROI decreases at least proportionally to the number of roots in it. Some recent subdivision algorithms have a proved nearly optimal complexity, are robust in the case of root clusters and multiple roots, and their implementation in [8] a little outperforms `MPsolve` for ROI containing only a small number of roots, which is an important benefit in many computational areas.

The Root Clustering Problem. $Z(\mathcal{S}, p)$ or $Z(\mathcal{S})$ is the root subset of p in a complex set $\mathcal{S}$; $\#(\mathcal{S}, p)$ or $\#(\mathcal{S})$ denotes the number of roots of p in $\mathcal{S}$. We always count roots with multiplicity.

We consider boxes (that is, squares with horizontal and vertical edges, parallel to coordinate axis) and discs $D(c, r) = \{z \text{ s.t. } |z - c| \leq r\}$ on the complex plane. For such a box (resp. disc) $\mathcal{S}$ and a positive δ we denote by $\delta\mathcal{S}$ the concentric δ-dilation. A disc Δ is an *isolator* if $\#(\Delta) > 0$; it is a *natural* isolator if in addition $\#(\Delta) = \#(3\Delta)$. A set $\mathcal{R}$ of roots of p is a *natural cluster* or just *cluster* for short if there exists a natural isolator Δ with $Z(\mathcal{R}) = Z(\Delta)$. Δ is an ε-isolator and the set $\mathcal{R}$ is an ε-cluster if ε exceeds the diameter of Δ.

The *Local Clustering Problem* (LCP) is the problem of computing natural ε-isolators for natural ε-clusters together with the sum of multiplicities of roots in the clusters in a fixed ROI:

Local Clustering Problem (LCP):
Given: a polynomial $p \in \mathbb{C}[z]$, a ROI $B_0 \subset \mathbb{C}$, $\varepsilon > 0$
Output: a set of pairs $\{(\Delta^1, m^1), \ldots, (\Delta^\ell, m^\ell)\}$ where:
- the Δ^j's are pairwise disjoint discs of radius $\leq \varepsilon$,
- $m^j = \#(\Delta^j, p) = \#(3\Delta^j, p)$ and $m^j > 0$ for $j = 1, \ldots, \ell$
- $Z(B_0, p) \subseteq \bigcup_{j=1}^{\ell} Z(\Delta^j, p) \subseteq Z(2B_0, p)$.

Root Clustering Problem (RCP) is a global version of LCP:

Root Clustering Problem:
Given: a polynomial $p \in \mathbb{C}[z]$ of degree d
Output: a set of pairs $\{(\Delta^1, m^1), \ldots, (\Delta^\ell, m^\ell)\}$ where:
- the Δ^j's are pairwise disjoint discs,
- $m^j = \#(\Delta^j, p) = \#(3\Delta^j, p)$ and $d > m^j > 0$ for $j = 1, \ldots, \ell$
- $\bigcup_{j=1}^{\ell} Z(\Delta^j, p) = Z(\mathbb{C}, p)$.

We can readily transform an algorithm for LCP into that RCP by using a bound on the norm of the roots of p, e.g., the Fujiwara bound (see [5]) for the ROI. Conversely, an algorithm RCP can initialize an algorithm for the LCP, followed by refining natural isolators to a fixed size, e.g., by means of solving the RCP itself. We can achieve quadratic convergence to the clusters by using Newton's iterations.

A nearly optimal subdivision algorithm of [1] solves the LCP by means of subdivision. It combines exclusion and counting tests based on Pellet's theorem and Newton iterations. [8] describes high-level improvements of [1] and a `C` implementation of its algorithm

*Rémi's work is supported by NSF Grants # CCF-1563942 and # CCF-1564132.
†Victor's work is supported by NSF Grants # CCF-1116736 and # CCF-1563942 and by PSC CUNY Award 698130048.

ISSAC '20, July 20–23, 2020, Kalamata, Greece

ACM ISBN 978-1-4503-7100-1/20/07…$15.00
https://doi.org/10.1145/3373207.3404063

called `Ccluster`[1]. Computational cost of application of Pellet's theorem to a disc $D(c, r)$ is dominated by the cost of shifting and scaling the variable $z \longrightarrow c + zr$ and of Dandelin-Gräffe's root-squaring iterations.

Our Contributions. The core tool for solving both LCP and RCP is a test for counting the number s_0 of roots in a disc $\Delta = D(c, r)$. If the boundary $\partial\Delta$ contains no roots, then by virtue of Cauchy's theorem

$$s_0 = \frac{1}{2\pi \mathbf{i}} \int_{\partial\Delta} \frac{p'(z)}{p(z)} dz, \text{ for } \mathbf{i} = \sqrt{-1}, \tag{1}$$

By following [7, 13, 20], we compute approximation s_0^* to s_0 by means of the evaluation of p'/p on q points of $\partial\Delta$. We give an *effective*[2] (*i.e.* implementable) description of our test, said to be P^*-test, for counting the number of roots in any disc Δ. This test involves no coefficients of p and can be applied to a *black box polynomial*, that is, a polynomial p given by a black box for its evaluation (and for implied evaluation of p' [10]). Unlike the counting tests based on Pellet's theorem, we do not require shifting and scaling the variable z and, moreover, replace Dandelin-Gräffe's costly root-squaring iterations by recursively doubling the number q of evaluation points.

By restricting our root-counting to decision whether the number of roots is 0 or not, we arrive at our exclusion test, said to be P^0-test; it decides if a disc contains no roots.

We show how to use our exclusion test in a subdivision algorithm for solving the RCP. Our algorithm can fail but always terminates. We provide some heuristic support for its correctness, and in Sec. 6 we point out to the most recent results on the formal support of the correctness of our approach, which seems to preserve the nearly optimal Boolean cost bound of the algorithms of [1] and [2]. Our goal, however, is not to compete with but to cooperate with algorithms of [1] and [2] and possibly to amend them.

We have implemented our algorithm in a procedure called `CclusterF`[3] and showed empirically that it allows significant practical improvements of root clustering compared to `Ccluster`. For sparse polynomials and polynomials defined by recursive process such as Mandelbrot's polynomials (see [3, Eq. (16)]), the resulting acceleration of the clustering algorithm of [1] is particularly strong. In experiments we carried out, `CclusterF` never failed.

Organization of the Paper. In Sec. 2 we approximate s_0 and estimate approximation error. Secs. 3 and 4 present our P^* and P^0-tests, respectively. In Sec. 5 we present our subdivision algorithm for solving the RCP using the P^0-test. In the rest of the present section, we recall the related work and the clustering algorithm of [1].

1.1 Previous Works

Univariate polynomial root-finding is a long-standing and still actual problem; it is intrinsically linked to numerical factorization of a polynomial into the product of its linear factors. The algorithms of [12] solved both problems of factorization and root-finding in record Boolean time, which is nearly optimal, that is, optimal up to a polylog factor in the input size and output precision. The algorithms are involved and have never been implemented. User's choice has been for a while the package of subroutines `MPsolve` (see [3] and [4]), based on simultaneous Newton-like (*i.e.* Ehrlich-Aberth) iterations. They converge to all roots simultaneously. As we said already, empirically they do this very fast right from the start, albeit with no formal support, and they approximate a small number of roots in a ROI not much faster than all roots. In contrast the nearly optimal cost of the algorithms of [12] and [1], already cited, is roughly proportional to the number of roots in a ROI. [1] extends the method of [2] to root clustering, *i.e.* it solves the LCP and is robust in the case of multiple roots; its implementation [8] is a little more efficient than `MPsolve` for ROIs containing a small number of roots; when all the roots are sought, `MPsolve` remains the user's choice. The algorithms of [1] and [2] follow subdivision algorithms of [17] and [11], presented there under the name of Quad-tree algorithms (inherited from [6]). [15, 16, 19] achieve a nearly optimal complexity in the real case; [9] implements the algorithm of [19]. Much more rudimentary variants of our algorithms and of their implementation appeared in [14] and [7], respectively. In Remark 8 we comment on a technical link to [20].

1.2 Solving the RCP

The root clustering algorithm in [1] combines two tests, called exclusion and counting test, with recursive subdivision of an initial box.

C^0 and C^* tests. The two tests C^0 and C^* exclude boxes with no roots of p and count the number of roots in a box, respectively. Both tests have a failure mode, *i.e.* return -1 when they cannot make decision. For a given complex disc Δ, $C^*(\Delta, p)$ (resp. $C^0(\Delta, p)$) returns an integer $k \geq 0$ (resp. 0) that indicate that there are precisely k (resp. no) roots in Δ. In the following, we frequently write $C^0(\Delta)$ for $C^0(\Delta, p)$ and $C^*(\Delta)$ for $C^*(\Delta, p)$.

In [1, 2, 8], both C^0 and C^* are based on the so called "soft Pellet test" denoted $T^*(\Delta, p)$ or $T^*(\Delta)$ which returns an integer $k \geq -1$ such that $k \geq 0$ only if p has k roots in Δ:

$$\begin{aligned} C^0(\Delta) :=& \begin{cases} 0 & \text{if } T^*(\Delta) = 0 \\ -1 & \text{otherwise} \end{cases} \\ C^*(\Delta) :=& \quad T^*(\Delta). \end{aligned} \tag{2}$$

Boxes, Quadri-section and Connected Components. The box B centered in $c = a + \mathbf{i}b$ with width w is defined as $[a - w/2, a + w/2] + \mathbf{i}[b - w/2, b + w/2]$. $w(B)$ denotes the width of B. The *containing disc* of B is the disc $\Delta(B) := D(c, \frac{3}{4}w(B))$ The four children of B are the four boxes centered in $(a \pm \frac{w}{4}) + \mathbf{i}(b \pm \frac{w}{4})$ and having width $\frac{w}{2}$.

Recursive subdivisions of a ROI B_0 amounts to the construction of a tree rooted in B_0. Below we refer to boxes that are nodes (and possibly leafs) of this tree as the boxes of the subdivision tree of B_0.

A *component* C is a set of connected boxes. The component box B_C of a component C is a smallest square box subject to $C \subseteq B_C \subseteq B_0$, where B_0 is the initial ROI. We write $\Delta(C)$ for $\Delta(B_C)$ and $w(C)$ for $w(B_C)$. Below we consider components made up of boxes of the same width; such a component is *compact* if $w(C)$ is at most 3 times the width of its boxes. Finally, a component C is *separated* from a set S of components if $\forall C' \in S, 4\Delta(C) \cap C' = \emptyset$ and $4\Delta(C) \subseteq 2B_0$.

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

[2] by effective, we refer to the pathway proposed in [21] to describe algorithms in three levels: abstract, interval, effective

[3] We have done this before deterministic support for correctness of our exclusion test appeared in [13] and verified correctness by using a test from [8].

Algorithm 1 Root Clustering Algorithm

Input: A polynomial $p \in \mathbb{C}[z]$ of degree d.
Output: Set R of components solving the LCP.

1: $w \leftarrow$ upper bound for the norm of the roots of p
2: $B_0 \leftarrow$ box centered in 0 with width w
3: $R \leftarrow \emptyset$, $Q \leftarrow \{B_0\}$ *// Initialization*
4: **while** Q is not empty **do** *// Main loop*
5: $C \leftarrow Q.pop()$ *//C has the widest component box in Q*
// Validation
6: **if** C is compact **and** C is separated from Q **then**
7: $k \leftarrow C^*(2\Delta(C), p)$
8: **if** $d > k > 0$ **then**
9: $R.push((C, k))$
10: **break**
// Bisection
11: $S \leftarrow$ empty set of boxes
12: **for** each box B of C **do**
13: **for** each child B' of B **do**
14: **if** $C^0(\Delta(B'), p)$ returns -1 **then**
15: $S.push(B')$
16: $Q.push($ connected components in S $)$
17: **return** R

A Root Clustering Algorithm. We give in Algo. 1 a simple root clustering algorithm based on subdivision. A ROI containing all the roots of p is constructed using the so-called Fujiwara bound for the norm of the roots of p (see [5]). In step 16, an implicit processing of S groups the boxes in components. The paper [1] proves that Algo. 1 terminates and outputs a correct solution provided that the C^0 and C^*-tests are as in Eq. (2).

In the **while** loop of Algo. 1, components with widest component box are processed first; together with the definition of a separated component, this implies the following remarks:

Remark 1. *Let C be a component in Algo. 1 that passes the test in step 6. Then C satisfies $\#(\Delta(C)) = \#(4\Delta(C))$.*

Remark 2. *If $\#(\frac{2\sqrt{2}}{3}\Delta) = \#(\frac{4}{3}\Delta)$, then $T^*(\Delta) \geq 0$ (see [1, Lem. 3]) and if C^* is defined as in (2), then k in step 7 of Algo. 1 is non-negative.*

2 THE POWER SUMS OF THE ROOTS IN THE UNIT DISC

Let roots $\{\alpha_1, \ldots, \alpha_{d_\Delta}\}$ of p lie in Δ, roots $\{\alpha_{d_\Delta+1}, \ldots, \alpha_d\}$ lie outside Δ, and no roots lie on the boundary $\partial\Delta$.

Definition 3 (The power sums of the roots in a disc). *The h-th power sum of the roots of p in Δ is the complex number*

$$s_h = \sum_{j=1}^{d_\Delta} \alpha_j^h \tag{3}$$

Hereafter q is an integer exceeding 1 and ζ denotes a primitive q-th root of unity. The h-th power sum of the roots in the unit disc $\Delta = D(0,1)$ can be approximated by

$$s_h^* = \frac{1}{q} \sum_{g=0}^{q-1} \zeta^{g(h+1)} \frac{p'(\zeta^g)}{p(\zeta^g)} \tag{4}$$

provided that Δ has no root on its boundary. The following theorem of [13] explicitly expresses the sums s_h^* through the roots.

Theorem 4. *Let $\alpha_1, \ldots, \alpha_d$ be all d roots of $p(z)$. Then*

$$s_h^* = \sum_{j=1}^{d} \frac{\alpha_j^h}{1-\alpha_j^q} \text{ unless } \alpha_j^q = 1 \text{ for some } j. \tag{5}$$

2.1 Proof of Theorem 4

We begin with recalling some auxiliary properties.

(i) Differentiate the equation $p(z) = \mathrm{lcf}(p) \prod_{j=1}^d (z-\alpha_j)$ and obtain

$$\frac{p'(z)}{p(z)} = \sum_{j=1}^{d} \frac{1}{z-\alpha_j}. \tag{6}$$

(ii) For a primitive q-th root of unity ζ, it holds that

$$\zeta^g \neq 1 \text{ for } 0 < g < q, \ \sum_{g=0}^{q-1} \zeta^g = 0, \text{ and } \zeta^q = 1. \tag{7}$$

(iii) Newman's expansion: if $|y| < 1$ then $\frac{1}{1-y} = \sum_{s=0}^{\infty} y^s$.

Lemma 5. *For a complex z with $|z| \neq 1$, integers $h \geq 0$ and $q > 1$ and a primitive q-th root of unity ζ it holds that*

$$\frac{1}{q} \sum_{g=0}^{q-1} \frac{\zeta^{(h+1)g}}{\zeta^g - z} = \frac{z^h}{1-z^q}. \tag{8}$$

Proof of Lem. 5: First let $|z| < 1$ and obtain

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

where the equation in the middle follows from Newman's expansion for $y = \frac{z}{\zeta^g}$. We can apply it because $|y| = |z|$ while $|z| < 1$.

Sum the fractions $\frac{z^s}{\zeta^{(s-h)g}}$
in g and deduce from (7) that

$$\frac{1}{q} \sum_{g=0}^{q-1} \frac{z^s}{\zeta^{(s-h)g}} = \begin{cases} z^s \text{ when } s = h + ql \text{ for an integer } l, \\ 0 \text{ otherwise.} \end{cases}$$

Therefore

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

Apply Newman's expansion for $y = z^q$ and deduce (8) provided that $|z| < 1$. Now let $|z| > 1$. Then

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

Sum these expressions in g, write $s := ql - h - 1$, and apply (7):

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

Apply Newman's expansion for $y = 1/z^q$ and obtain that

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

Hence (8) holds in the case where $|z| > 1$ as well. □

2.2 Error bounds for the approximation of the power sums

Definition 6 (Isolation ratio). *A complex disc Δ has an* isolation ratio $\rho \geq 1$ *for a polynomial p or equivalently is* ρ-isolated *if* $Z(\frac{1}{\rho}\Delta) = Z(\rho\Delta)$.

Corollary 7. *For $\rho > 1$, $\theta := 1/\rho$, and an integer h such that $0 \leq h < q$ let the disc $\Delta = D(0,1)$ be ρ-isolated and contain d_Δ roots of p. Then*

$$|s_h^* - s_h| \leq \frac{d_\Delta \theta^{q+h} + (d - d_\Delta)\theta^{q-h}}{1 - \theta^q}. \tag{9}$$

Remark 8. *The corollary does not imply Theorem 4. In [20] Schönhage proved the corollary and applied it to deflation of p, ignoring the case of $h = 0$ and root-counting problem and bypassing the theorem.*

Proof of Corollary 7: Let $\{\alpha_1, \ldots, \alpha_{d_\Delta}\}$ be the roots of p in Δ and $\{\alpha_{d_\Delta+1}, \ldots, \alpha_d\}$ be the roots of p outside Δ. First combine (5) with (3) to obtain

$$s_h^* - s_h = \sum_{j=1}^{d_\Delta} \frac{\alpha_j^{q+h}}{1 - \alpha_j^q} + \sum_{j=d_\Delta+1}^{d} \frac{\alpha_j^h}{1 - \alpha_j^q} \tag{10}$$

Recall that $\theta = 1/\rho < 1$. For $1 \leq j \leq d_\Delta$, one has

$$\left| \frac{\alpha_j^{q+h}}{1 - \alpha_j^q} \right| \leq \frac{\theta^{q+h}}{1 - \theta^q} \tag{11}$$

For $d_\Delta + 1 \leq j \leq d$, it holds that $1/\alpha_j \leq \theta$ and

$$\left| \frac{\alpha_j^h}{1 - \alpha_j^q} \right| = \left| \frac{\alpha_j^{h-q}}{1/\alpha_j^q - 1} \right| \leq \frac{\theta^{q-h}}{1 - \theta^q} \tag{12}$$

Combining (10), (11) and (12) implies inequality (9). □

3 COUNTING THE NUMBER OF ROOTS IN A WELL-ISOLATED DISC

Given $\rho > 1$, a black box polynomial p, and ρ-isolated disc $D(0,1)$, Corollary 7 suggests the following recipe for counting the roots of p in $D(0,1)$: first choose q such that $|s_0^* - s_0|$ is less than $1/4$ and then compute s_0^* of (4), at the overall cost of the evaluation of p and p' at $q = O(\log(d))$ points and $O(q)$ additional arithmetic operations. Clearly a unique integer in the disc $D(s_0^*, 1/4)$ is the number of roots in $D(0,1)$. In this section we extend this recipe to P^*-test for counting the roots of a black box polynomial p in any ρ-isolated disc $\Delta = D(c,r)$ for $\rho > 1$.

When Δ has isolation ratio 2 and p has degree 500, our test amounts to evaluating p and p' on $q = 11$ points.

If p and p' can be evaluated at a low computational cost, e.g. if p is sparse or defined by a recurrence as the Mandelbrot polynomial (see [3, Eq. (16)]), our P^*-test can be dramatically simplified.

3.1 Approximation of the 0-th Power Sum of the Roots in any Disc

Let $\Delta = D(c,r)$ and s_0 be the 0-th power sum of the roots of p in Δ, as defined in Def. 3. For a positive integer q, define

$$s_0^* = \frac{r}{q} \sum_{g=0}^{q-1} \zeta^g \frac{p'(c + r\zeta^g)}{p(c + r\zeta^g)} \tag{13}$$

where ζ is a primitive q-th root of unity.

Corollary 9. *Let Δ have isolation ratio ρ, and $\theta = 1/\rho$. Then*

$$|s_0^* - s_0| \leq \frac{d\theta^q}{1 - \theta^q} \tag{14}$$

$$\textit{Fix } e > 0. \textit{ If } q = \lceil \log_\theta(\frac{e}{d+e}) \rceil \textit{ then } |s_0^* - s_0| \leq e \tag{15}$$

Proof of Corollary 9: Let $p_\Delta(z)$ be the polynomial $p(c + rz)$. Then $p'_\Delta(z) = rp'(c + rz)$ and Eq. (13) rewrites $s_0^* = \frac{1}{q} \sum_{g=0}^{q-1} \zeta^g \frac{p'_\Delta(\zeta^g)}{p_\Delta(\zeta^g)}$. In addition, the unit disc $D(0,1)$ has isolation ratio ρ for p_Δ and contains s_0 roots of p_Δ. Then apply Thm. 7 to $p_\Delta(z)$ to obtain (14). (15) is a direct consequence of (14).

□

Remark that in (15), the required number q of evaluation points increases as the logarithm of ρ: if Δ has isolation ratio $\sqrt{\rho}$ (resp. ρ^2) instead of ρ, then $\frac{1}{2}q$ (resp. $2q$) evaluation points are required. Thus doubling the number of evaluation points has the same effect as root squaring operations.

Our test uses the following bound.

Lemma 10. *Suppose that $\Delta = D(c,r)$ has isolation ratio $\rho > 1$, $z \in \mathbb{C}$, $|z| = 1$, and g is an integer. Then*

$$|p(c + rz^g)| \geq \mathrm{lcf}(p) \frac{r^d(\rho - 1)^d}{\rho^d} \tag{16}$$

Proof of Lem. 10. Suppose that p has d_Δ non-necessarily distinct roots $\alpha_1, \ldots, \alpha_{d_\Delta}$ in Δ and $d - d_\Delta$ roots $\alpha_{d_\Delta+1}, \ldots, \alpha_d$ outside Δ. Since Δ has isolation ratio ρ, it follows that

$$|c + rz^g - \alpha_i| \geq r - \frac{r}{\rho} = \frac{r(\rho - 1)}{\rho} \text{ when } i \leq d_\Delta, \text{ and} \tag{17}$$

$$\geq \rho r - r = r(\rho - 1) \text{ when } i \geq d_\Delta + 1 \tag{18}$$

Write

$$p(c + rz^g) = \mathrm{lcf}(p) \prod_{i=1}^{d_\Delta} (c + rz^g - \alpha_i) \prod_{i=d_\Delta+1}^{d} (c + rz^g - \alpha_i)$$

and deduce from inequalities (17) and (18) that

$$|p(c + rz^g)| \geq \mathrm{lcf}(p) (\frac{r(\rho - 1)}{\rho})^{d_\Delta} (r(\rho - 1))^{d - d_\Delta} = \mathrm{lcf}(p) \frac{r^d(\rho - 1)^d}{\rho^{d_\Delta}}.$$

Bound (16) follows since $\rho > 1$. □

3.2 Black Box for Evaluating a Polynomial on an Oracle Number

Our P^*-test deals with *oracle numbers*, the black boxes for arbitrary precision approximation of complex numbers. Such oracle numbers can be implemented through arbitrary precision interval arithmetic or ball arithmetic. Let $\square\mathbb{C}$ be the set of complex intervals. If $\square a \in \square\mathbb{C}$, then $w(\square a)$ is the maximum width of real and imaginary parts of $\square a$.

For a number $a \in \mathbb{C}$, we call *oracle* for a a function $O_a : \mathbb{N} \to \square\mathbb{C}$ such that $a \in O_a(L)$ and $w(O_a(L)) \leq 2^{-L}$ for any 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 O_a is an oracle for a and $L \in \mathbb{N}$, then $p(a) \in \mathcal{I}_p(O_a, L)$ and $w(\mathcal{I}_p(O_a, L)) \leq 2^{-L}$.

Algorithm 2 $P^*(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, \rho)$

Input: $\mathcal{I}_p$, $\mathcal{I}_{p'}$ evaluation oracles for p and p', $\Delta = D(c, r)$, $\rho > 1$. p is monic and has degree d.
Output: an integer in $\{-1, 0, \ldots, d\}$

1: $L \leftarrow 53$, $w \leftarrow 1$, $e \leftarrow 1/4$
2: $\theta \leftarrow 1/\rho$, $q \leftarrow \lceil \log_\theta(\frac{e}{d+e}) \rceil$
3: $\ell \leftarrow r^d(\rho-1)^d/\rho^d$
4: **while** $w \geq 1/2$ **do**
5: **for** $g = 0, \ldots, q-1$ **do**
6: Compute intervals $\mathcal{I}_p(O_{c+r\zeta^g}, L)$ and $\mathcal{I}_{p'}(O_{c+r\zeta^g}, L)$
7: **if** $|\mathcal{I}_p(O_{c+r\zeta^g}, L)| < \ell$ **then**
8: **return** -1
9: Compute interval $\square s_0^*$ as $\frac{r}{q} \sum_{g=0}^{q-1} O_{\zeta^g}(L) \frac{\mathcal{I}_{p'}(O_{c+r\zeta^g}, L)}{\mathcal{I}_p(O_{c+r\zeta^g}, L)}$
10: $w \leftarrow w(\square s_0^*)$
11: $L \leftarrow 2L$
12: $\square s_0 \leftarrow \square s_0^* + [-e, e] + \mathbf{i}[-e, e]$
13: **if** $\square s_0$ contains a unique integer k **then**
14: **return** k
15: **return** -1

Consider evaluation oracles $\mathcal{I}_p$ and $\mathcal{I}_{p'}$ for p and its derivative p'. If p is given by $d+1$ oracles for its coefficients, one can easily construct $\mathcal{I}_p$ and $\mathcal{I}_{p'}$ by using, for instance, Horner's rule. However for some polynomials defined by a procedure, one can construct fast evaluation oracles $\mathcal{I}_p$ and $\mathcal{I}_{p'}$ from procedural definition.

3.3 The P^*-test

In Algo. 2 we describe our counter of the roots of a monic polynomial p in a disc $\Delta = D(c, r)$. Its input is made up of evaluation oracles for p and p', Δ, and a fixed isolation ratio $\rho > 1$ for Δ. It may fail and return -1 only if Δ is not ρ-isolated for $\rho > 1$. In the latter case its correctness cannot be guaranteed. The termination of Algo. 2 amounts to the termination of the **while** loop in step 4.

First suppose that $0 \leq g < q$ for an integer g, a disc Δ is ρ-isolated for $\rho > 1$, and $|p(c + r\zeta^g)| \geq \ell > 0$ (cf. (16)). Thus for $2^{-L} < \ell$, none of the $\mathcal{I}_p(O_{c+r\zeta^g}, L)$ can contain 0, and the width of the interval $\square s_0^*$ computed in step 9 strictly decreases with L (see, e.g., [18, Sec. 5] and in particular Eq. (5.10) and (5.11) that directly extend to $\mathbb{C}$).

Now suppose that the disc Δ is not ρ-isolated for a fixed $\rho > 1$. If one of the evaluation points $c + r\zeta^g$ is a root of p or if $p(c + r\zeta^g) < l$ then condition in step 7 is satisfied for $2^{-L} < \ell$, and the test returns -1. Otherwise $p(c + r\zeta^g) \geq \ell$ for all $g = 0, \ldots, q-1$, and then the interval $\square s_0^*$ computed in step 9 has width that strictly decreases with L.

This proves the termination of Algo. 2 when p is monic. One can easily write a terminating algorithm for non-monic polynomials assuming a lower bound on the leading coefficient of p.

The correctness of Algo. 2 is stated in the following proposition:

Proposition 11. *Let Δ be ρ-isolated. Then p has k roots in Δ if and only if $P^*(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, \rho)$ returns k.*

Proof of Prop. 11. Since Δ is ρ-isolated Lemma 10 implies that the condition in step 7 is never reached. By virtue of Corollary 9, the interval $\square s_0$ computed in step 12 of Algo. 2 has width less than 1 and contains a unique integer s_0, the number of roots of p in Δ counted with multiplicity. □

4 AN ALMOST SURE EXCLUSION TEST

The P^*-test of sec. 3 counts the roots in Δ using no Taylor's shift but just evaluates p at $O(\log(d))$ points on the contour of Δ, which is a major benefit versus [1]. However, we cannot ensure its success unless we know that Δ is ρ-isolated for ρ noticeably exceeding 1, and this disqualifies its use as an exclusion test within a subdivision framework of [1]. Our alternative version of the P^*-test in [7] works in the case where ρ is not known, and we used it as an exclusion test while confirming its output with the T^*-test.

Here we define an exclusion test based on the computation of approximations of the first k power sums (for a small $k > 0$). For a disk with a fixed isolation ratio one can compute an interval $\square s_0$ containing a unique integer s as in Algo. 2. However, if the isolation ratio is smaller, then s may differ from s_0. In the test described below, we verify further necessary conditions that $s = s_0$. Namely, we compute the k first power sums of the roots of p_Δ in $D(0, 1)$ for a small k: for h less that k, p contains no root in Δ only if the h-th powers of the roots of p in Δ sum to 0, which in turn happens only if the h-th powers of the roots of p_Δ in $D(0, 1)$ sum to 0.

Subsec. 4.1 extends Corollary 9 to the approximation of the h-th power sum of the roots of p_Δ in $D(0, 1)$. In Subsec. 4.2, we define our exclusion test and give a sufficient condition in terms of the distances of the roots to a box B for our test to exclude B. In Subsec. 4.3 we provide experimental results confirming our heuristic.

4.1 Approximation of h-th power sum

For a positive integer q and $0 \leq h < q$, define

$$s_h^* = \frac{r}{q} \sum_{g=0}^{q-1} \zeta^{g(h+1)} \frac{p'(c + r\zeta^g)}{p(c + r\zeta^g)}. \tag{19}$$

By replacing h by 0, (19) directly extends (13), and likewise bound (20) below directly extends (9).

Corollary 12 (of Theorem 7). *Let $\Delta = D(c, r)$ have isolation ratio ρ, and $\theta = 1/\rho$. Let p have degree d and d_Δ roots in Δ. Then*

$$|s_h^* - s_h| \leq \frac{d_\Delta \theta^{q+h} + (d - d_\Delta)\theta^{q-h}}{1 - \theta^q} \tag{20}$$

$$|s_h^* - s_h| \leq \frac{d\theta^{q-h}}{1 - \theta^q} \tag{21}$$

$$\text{Fix } e > 0. \text{ If } q = \lceil \log_\theta(\frac{e}{d+e}) \rceil + h, \text{ then } |s_h^* - s_h| \leq e \tag{22}$$

Proof of (21) and (22) in Corollary 12: We deduce (21) from (20) by noticing that $\theta < 1$ and $d_\Delta \theta^{q+h} \leq d_\Delta \theta^{q-h}$. (22) is a direct consequence of (21). □

4.2 The P^0-test

We describe our P^0-test in Algo. 3 in the case where p is monic. At the first stage, we rely on eq. (19) and for $0 \leq h \leq k$, compute the interval $\square s_h^*$, containing s_h^* and having width less than 1/2. At the second stage, for $0 \leq h \leq k$, we obtain the interval $\square s_h$ from $\square s_h^*$ by adding the errors bounded in (21). $\square s_h$ contains s_h for all h if Δ

Algorithm 3 $P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, \rho, k)$

Input: $\mathcal{I}_p, \mathcal{I}_{p'}$ evaluation oracles for p and p', $\Delta = D(c, r)$, $\rho > 1$. p is monic and has degree d. $k \geq 0$ is an integer.
Output: an integer in $\{-1, 0\}$

1: $L \leftarrow 53$, $w \leftarrow 1$, $\theta \leftarrow 1/\rho$
2: $e \leftarrow 1/4$, $q \leftarrow \lceil \log_\theta(\frac{e}{d+e}) \rceil + k$
3: $\ell \leftarrow r^d(\rho-1)^d/\rho^d$
4: **while** $w \geq 1/2$ **do**
5: **for** $g = 0, \ldots, q-1$ **do**
6: Compute intervals $\mathcal{I}_p(O_{c+r\zeta^g}, L)$ and $\mathcal{I}_{p'}(O_{c+r\zeta^g}, L)$
7: **if** $|\mathcal{I}_p(O_{c+r\zeta^g}, L)| < \ell$ **then**
8: **return** -1
9: **for** $h = 0, \ldots, k$ **do**
10: Compute interval $\square s_h^*$ as $\frac{r}{q} \sum_{g=0}^{q-1} O_{\zeta^{g(h+1)}}(L) \frac{\mathcal{I}_{p'}(O_{c+r\zeta^g}, L)}{\mathcal{I}_p(O_{c+r\zeta^g}, L)}$
11: $w \leftarrow \max_{h=0,\ldots,k} w(\square s_k^*)$
12: $L \leftarrow 2 * L$
13: **for** $h = 0, \ldots, k$ **do**
14: $e_h \leftarrow (d\theta^{q-h})/(1-\theta^q)$
15: $\square s_h \leftarrow \square s_h^* + [-e_h, e_h] + \mathbf{i}[-e_h, e_h]$
16: **if** $0 \notin \square s_h$ **then**
17: **return** -1
18: **return** 0

has isolation ratio $\rho > 1$. We have chosen q such that $\square s_h$ contains at most one integer for all h and we arrive at

Proposition 13. *If Δ has isolation ratio ρ then for $k \geq 0$, $P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, \rho, k)$ returns 0 if and only if Δ contains no root of p.*

One proves the termination of Algo. 3 with the same arguments as for Algo. 2, and the same remark about the assumption that p is monic holds. For a box B that contains no root of p, we give a sufficient condition for our test to return 0:

Proposition 14. *Let a disc $\Delta(B)$ contain a box B. If $2B$ contains no root of p then $P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta(B), 4/3, k)$ returns 0 for any $k \geq 0$.*

Proof of Prop. 14. Let B have center c and width w. Recall that $\Delta(B) = D(c, \frac{3}{4}w)$, thus $\frac{4}{3}\Delta(B) = D(c, w)$. Now, $D(c, w) \subseteq 2B$ and if $2B$ contains no roots of p, then so does $D(c, w)$ as well. Thus $\Delta(B)$ has isolation ratio $\geq \frac{4}{3}$. In this case, by virtue of Lemma 10, $p(c + r\zeta^g) \geq \ell$ for all $g = 0, \ldots, q-1$. As a consequence, after the **while** loop of Algo. 3, each interval $\square s_h^*$ has width strictly less than $1/2$, and contains s_h^*. Now, one has the following bounds:

$$|s_h^* - s_h| \leq \frac{d\theta^{q-h}}{1-\theta q} \leq \frac{d\theta^{q-k}}{1-\theta q} \leq 1/4 \qquad (23)$$

The first inequality comes from (21). The inequality in the middle holds since $\theta < 1$ and $h \leq k < q$. The right-hand side inequality is a consequence of (22) and the choice of e. Thus $\square s_h$ computed in step 15 of Algo. 3 contains s_h, which is 0, and contains a unique integer since it has width strictly less than 1. □

4.3 On the success of the P^0-test

Here we give experimental evidences that for a given disc Δ, if $P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, \rho, k)$ with $\rho = 4/3$ and $k = 2$ returns 0, then Δ is very likely to contain no roots of p.

		T^*-tests	P^0-tests, $k=0$		P^0-tests, $k=1$		P^0-tests, $k=2$		
d	n	t_0/t (%)	#TN	#FP	#TN	#FP	#TN	#FP	t_0^2/t_0
100 monic random dense polynomials per degree									
64	116302	87.2	3741	4	5611	0	7260	0	1.14
128	227842	90.5	6417	21	9935	0	12972	0	.599
191	340348	92.0	8850	26	13770	1	18004	0	.455
Bernoulli polynomials									
64	1566	87.5	32	0	42	0	60	0	.836
128	2954	88.4	49	0	65	0	87	0	.578
191	4026	88.7	100	0	163	0	212	0	.462
100 monic random sparse (10 monomials) polynomials per degree									
64	115850	86.2	3628	10	5430	0	6986	0	.981
128	226266	91.3	6471	11	9660	0	12556	0	.403
191	331966	92.1	8690	11	13425	2	17452	0	.280
Mignotte polynomials									
64	1196	85.7	30	0	48	0	63	0	1.00
128	2296	92.9	63	0	93	0	129	0	.298
191	3218	92.4	70	2	109	0	154	0	.264

Table 1: True negatives and false positives when using the P^*-test with $\rho = 4/3$ and $k = 0, 1, 2$.

We run Algo. 1 implemented in `Ccluster` for dense and sparse polynomials, random and taken from literature; each time `Ccluster` applies exclusion test based on T^*-test for a box B, we also apply $P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta(B), \frac{4}{3}, k)$ with values 0, 1, 2 for k.

The *false positives* are the cases where for a disc $\Delta(B)$, the T^*-test returns a positive number of roots or cannot decide whether $P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, 2, k)$ returns 0. For the polynomials we tested, when $k = 2$, there was no such false positives.

The *true negatives, i.e.* cases where a disc contains no root according to the T^*-test but $P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, 2, k)$ returns -1, shows how less efficacious than the T^*-test is our test.

4.3.1 Testing suite. For each degree $d \in \{64, 128, 191\}$, we generated 100 random monic dense polynomials whose coefficients are rational numbers $\frac{c}{256}$ where c is an integer chosen uniformly in $[-256, 256]$. We also generated for each degree above 100 random monic sparse polynomials as follows: choose 8 distinct random integers $d_1, \ldots, d_8$ in the range $[1, d-1]$, and let the coefficients of monomials of degrees $0, d_1, \ldots, d_8$ be rational numbers $\frac{c}{256}$ for a random integer chosen uniformly in $[-256, 256]$.

We also consider Bernoulli and Mignotte 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. It has about $d/2$ non-zero coefficients. The Mignotte polynomial of degree d and parameter $a = 8$ is $M_d(z) = z^d - 2(2^a z - 1)^2$.

4.3.2 Results. For a polynomial in our testing suite, let n be the number of exclusion tests performed by `Ccluster`, t be the running time of `Ccluster` and t_0 be the time spent in exclusion T^*-test. For each exclusion test, we also applied three times our P^0-test with isolation ratio $\rho = 4/3$ and $k = 0, 1, 2$. We denote by #TN the number of true negatives, #FP the number of false positives and t_0^2 the total time spent in the P^0-test with $k = 2$. We report in Table 1 the values n, t_0/t, #TN, #FP and t_0^2/t for each degree and each family of polynomials. For random dense and sparse polynomials, these values represented overall count over the 100 polynomials.

As expected, the number of false positives decreased when k increased, and we had no such false positives when we used $k = 2$. As a counterpart, the number of true negatives increased with k.

5 A FAST AND ALMOST SURE ROOT CLUSTERING ALGORITHM

In this section we present a fast root clustering algorithm based on Algo. 1 and on exclusion and counting tests defined as:

$$\begin{array}{ll} C^0(\Delta) := & P^0(\mathcal{I}_p, \mathcal{I}_{p'}, \Delta, 4/3, 2) \\ C^*(\Delta) := & T^*(\Delta) \end{array} \tag{24}$$

Notice that the exclusion test is performed while assuming an isolation ratio 4/3 for Δ, condition that cannot be ensured when Δ is the containing disc of the boxes of a subdivision tree constructed by Algo. 1. As a consequence, exclusion tests defined in (24) may return wrong results. Although very unlikely, as we show in Subsec. 4.3, this would compromise correctness of the process (termination is ensured by Prop. 14).

In Subsec. 5.1, we describe how we modified Algo. 1 to obtain a root clustering algorithm using C^0 and C^*-tests of (24) that always terminates and has a failure mode. When it succeeds, its result is correct. This procedure has been implemented in `C` within `Ccluster`, and we call it `CclusterF` below.

In Subsec. 5.2 we show experimental results on using `Ccluster` and `CclusterF` for clustering the roots of a bunch of polynomials. `CclusterF` never failed in the experiments we carried out. Moreover by comparing running times of both procedures, we show that using C^0-tests of (24) can lead to important improvement, which grow with degree and sparsity of considered polynomials.

5.1 Description of our algorithm

We give an informal description of how we modified Algo. 1 to deal with uncertainty of the result of the exclusion test P^0.

First, in addition to a list of clusters, our algorithm returns a flag in {`fail`, `success`} indicating whether its result is reliable.

Second, we replace steps 6 to 10 in Algo. 1 with steps 6 to 12 below:

6: **if** C is compact **and** C is separated from Q **then**
7: $k \leftarrow C^*(2\Delta(C), p)$
8: **if** $d > k > 0$ **then**
9: $R.push((C, k))$
10: **break**
11: **if** $k == -1$ **then**
12: **return** `fail`, R

Third, we replace the **return** statement in step 17 in Algo. 1 with the following simple routine:

17: sum the number of roots in the components in R
18: **if** it is equal to d **then**
19: **return** `success`, R
20: **else**
21: **return** `fail`, R

Notice that step 14 of Algo. 1 also involves the C^0-test which has to be understood here as defined in (24). Recall that for a box B, $C^0(\Delta(B))$ returns -1 when $2B$ contains a root (see Prop. 14); however when it returns 0, B may contain a root.

To see that our algorithm terminates, consider Prop. 14: it implies that after a finite number of subdivision steps, boxes in the subdivision tree form separated and compact connected components, at most one per root. Then for each of these connected components C our algorithm enters step 6 above and terminates.

When our algorithm returns the flag `success`, its output is correct, *i.e.* the components in R solve the root clustering problem. This is a direct consequence of the fact that $T^*(\Delta, p)$ returns $k \geq 0$ only if Δ contains k roots of p.

Our algorithm returns the flag `fail` only if an exclusion test returns a wrong result, *i.e.* excludes a box of the subdivision tree that contains a root. Assume the opposite: no exclusion test returns a wrong result. Then Rem. 1 holds; in particular $2\Delta(C)$ has isolation ratio 2 and from Rem. 2, $C^*(2\Delta(C), p)$ in step 7 above returns k positive. Moreover, each root lies in a box in a component in R before the step 17 above, and our algorithm returns `success`.

5.2 Experimental results

5.2.1 Test polynomials. In addition to the test polynomials of Subsec. 4.3, we consider the following ones.

(i) $T_d(z)$, the Chebyshev polynomial (of the first kind) of degree d: $T_0(z) = 1$, $T_1(z) = z$ and $T_{d+1}(z) = 2zT_d(z) - T_{d-1}(z)$, $d = 2, 3, \ldots$

(ii) $L_d(z)$, the Legendre polynomial of degree d: $L_0(z) = 1$, $L_1(z) = z$ and $L_{d+1}(z) = \frac{2d+1}{d+1} zL_d(z) - \frac{d}{d+1} L_{d-1}(z)$, $d = 2, 3, \ldots$

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

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

(iv) Letting $M_1(z) = z$ and $M_k(z) = zM_{k-1}(z)^2 + 1$, we define the Mandelbrot's polynomial $M_k(z)$ of degree $2^k - 1$.

Bernoulli, Chebyshev and Legendre polynomials of degree d have about $d/2$ nonzero coefficients. Polynomials with roots on a grid of degree d have about $d/4$ nonzero coefficients. Mignotte polynomials have 4 nonzero coefficients. Mandelbrot polynomials have no zero coefficients, but can be evaluated very fast by a straight line program.

5.2.2 Results. We computed clusters of roots of each polynomial of our testing set by using both `Ccluster` and `CclusterF`. In Table 2 we report for both solvers the size of the subdivision tree (columns TS) and the sequential running time in seconds on Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz machine with Linux (columns t and t'). In Table 2 we also report the number of failures of `CclusterF` (column #Fails) and the ratio t'/t in percents. Column t'_1/t' shows percents of time spent on evaluating oracle polynomials in the P^0-test. Column t'_2/t' shows percents of time spent on applying T^*-tests in `CclusterF`. As in Subsec. 4.3, the Table 2 displays the average data for random dense and sparse polynomials over the 100 polynomials of the family.

Remarks: (i) There was no occurrence of a failure of `CclusterF` for all the polynomials we tested.

(ii) The running time of `CclusterF` decreased as the degree and the sparsity of the polynomial increased. For random sparse polynomials and Mignotte polynomials, of degree 191, this was a 3-fold speed-up. The speed-up was more dramatic for polynomials evaluated very fast such as Mandelbrot polynomials. Except for the latter cases, `CclusterF` spent most of its computational time on evaluating oracle polynomials and checking correctness of the results.

	Ccluster		CclusterF					
d	TS	t	#Fails	TS	t'	t'/t (%)	t'_1/t' (%)	t'_2/t' (%)
100 monic random dense polynomials per degree								
64	127100	31.5	0	155992	41.2	130	76.8	1.72
128	250928	222	0	300696	149	67.3	83.2	4.52
191	361340	665	0	447628	340	51.1	85.1	5.96
Bernoulli polynomials								
64	1884	0.46	0	2148	0.49	106	73.4	2.04
128	3596	3.24	0	3932	1.86	57.4	85.4	2.15
191	4684	9.17	0	5476	4.84	52.7	84.2	5.99
Chebyshev polynomials								
64	2532	0.74	0	2980	0.79	106	82.2	1.26
128	4708	5.62	0	5188	3.33	59.2	84.9	.900
191	7268	17.0	0	8108	8.86	51.9	86.9	1.01
Legendre polynomials								
64	2676	0.75	0	2940	0.81	108	77.7	0.0
128	4836	5.76	0	5244	3.73	64.7	86.8	1.60
191	6996	16.4	0	7732	9.61	58.3	88.7	1.45
Polynomials with roots on a regular grid								
225	3412	8.74	0	3580	2.62	29.9	76.3	15.2
289	4548	17.0	0	5304	5.40	31.6	74.8	15.5
361	6276	30.9	0	7588	8.52	27.5	76.5	17.8
100 monic random sparse (10 monomials) polynomials per degree								
64	127220	27.9	0	159972	31.7	113	70.8	1.57
128	251196	216	0	303260	100	46.3	75.5	5.65
191	374872	638	0	457084	209	32.7	76.4	8.80
Mignotte polynomials								
64	1572	0.30	0	1856	0.31	103	74.1	0.0
128	2572	2.24	0	3564	0.95	42.4	74.7	4.21
191	3640	5.99	0	4228	1.79	29.8	72.6	10.0
Mandelbrot polynomials								
127	2852	3.46	0	3424	0.56	16.1	42.8	10.7
255	4968	18.4	0	5952	1.79	9.70	33.5	41.3
511	9632	118	0	11556	7.61	6.42	19.5	66.3

Table 2: Runs of `Ccluster` and `CclusterF` on polynomials of our testing suite.

In all the examples we tested, the depth of the subdivision tree constructed by `CclusterF` was at most one plus the depth of the tree constructed by `Ccluster`. Columns TS in Table 2 suggest that `CclusterF` tends to construct a slightly wider subdivision tree than `Ccluster`, which shows that the P^0-test is slightly less efficacious than the T^*-test for box exclusion.

6 CONCLUSION

We presented our exclusion test with doubling the number q of evaluation points as heuristic, but actually it has already probabilistic and even deterministic support; moreover even our root-counting has probabilistic support. Namely, by virtue of [13, Thm. 29 and Remark 30], based on our Theorem 4, if s_0^* is within a specified distance from an integer k, then $k = s_0$ with a high probability (whp) under the random root model and under no assumption about isolation of the disc. Furthermore by virtue of [13, Corollary 4.7] a disc contains no roots whp under a random coefficient model and again under no assumption about isolation of the disc as long as $2\tau^2 d^2 < 1$ for $\tau^2 = \sum_{h=0}^{q-1} |s_h^* - s_h|^2$ and $q \geq 2$. For $q > d$ under this bound the disc definitely contains no roots by virtue of [13, Corollary 4.6]. Notice that we can compute s_h^* for $h = 0, 1, \ldots, q-1$ at the cost of computing just s_0^* and in addition performing discrete Fourier transform at q points.

Our initial but extensive experiments showed significant acceleration of the known subdivision root finders, which is particularly strong for sparse inputs. Moreover they suggest that the latter result of [13] is overly pessimistic because exclusion test was always correct in these experiments already for q much smaller than d.

REFERENCES

[1] Ruben Becker, Michael Sagraloff, Vikram Sharma, Juan Xu, and Chee Yap. 2016. Complexity Analysis of Root Clustering for a Complex Polynomial. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation (ISSAC '16)*. ACM, New York, NY, USA, 71–78. https://doi.org/10.1145/2930889.2930939

[2] Ruben Becker, Michael Sagraloff, Vikram Sharma, and Chee Yap. 2018. A Near-Optimal Subdivision Algorithm for Complex Root Isolation based on Pellet Test and Newton Iteration. *Journal of Symbolic Computation* 86 (May-June 2018), 51–96.

[3] Dario A Bini and Giuseppe Fiorentino. 2000. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Num. Alg.* 23, 2 (2000), 127–173.

[4] Dario A Bini and Leonardo Robol. 2014. Solving secular and polynomial equations: A multiprecision algorithm. *J. Comput. Appl. Math.* 272 (2014), 276–292.

[5] Matsusaburô Fujiwara. 1916. Über die obere Schranke des absoluten Betrages der Wurzeln einer algebraischen Gleichung. *Tohoku Mathematical Journal, First Series* 10 (1916), 167–171.

[6] Peter Henrici and Irene Gargantini. 1969. Uniformly convergent algorithms for the simultaneous approximation of all zeros of a polynomial. In *Constructive Aspects of the Fundamental Theorem of Algebra*. Wiley-Interscience New York, 77–113.

[7] Rémi Imbach and Victor Y Pan. 2019. New practical advances in polynomial root clustering. *arXiv preprint arXiv:1911.06706* (2019).

[8] Rémi Imbach, Victor Y. Pan, and Chee Yap. 2018. Implementation of a Near-Optimal Complex Root Clustering Algorithm. In *Mathematical Software – ICMS 2018*. 235–244.

[9] Alexander Kobel, Fabrice Rouillier, and Michael Sagraloff. 2016. Computing Real Roots of Real Polynomials ... And Now For Real!. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation (ISSAC '16)*. ACM, New York, NY, USA, 303–310. https://doi.org/10.1145/2930889.2930937

[10] Seppo Linnainmaa. 1976. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics* 16, 2 (1976), 146–160.

[11] Victor Y Pan. 2000. Approximating complex polynomial zeros: modified Weyl's quadtree construction and improved Newton's iteration. *J. of Complexity* 16, 1 (2000), 213–264.

[12] Victor Y Pan. 2002. Univariate polynomials: nearly optimal algorithms for numerical factorization and root-finding. *J. of Symb. Comp.* 33, 5 (2002), 701–733.

[13] Victor Y. Pan. 2018. New Acceleration of Nearly Optimal Univariate Polynomial Root-finders. (2018). arXiv:cs.NA/1805.12042, last revised May 2020

[14] Victor Y. Pan. 2019. Old and New Nearly Optimal Polynomial Root-Finders. In *Computer Algebra in Scientific Computing*, Matthew England, Wolfram Koepf, Timur M. Sadykov, Werner M. Seiler, and Evgenii V. Vorozhtsov (Eds.). Springer International Publishing, Cham, 393–411.

[15] Victor Y. Pan and Elias P. Tsigaridas. 2013. On the Boolean Complexity of Real Root Refinement. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation (ISSAC '13)*. ACM, New York, NY, USA, 299–306. https://doi.org/10.1145/2465506.2465938

[16] Victor Y Pan and Elias P Tsigaridas. 2016. Nearly optimal refinement of real roots of a univariate polynomial. *J. of Symb. Comp.* 74 (2016), 181–204.

[17] James Renegar. 1987. On the worst-case arithmetic complexity of approximating zeros of polynomials. *J. of Complexity* 3, 2 (1987), 90–113.

[18] Siegfried M Rump. 2010. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica* 19 (2010), 287–449.

[19] Michael Sagraloff and Kurt Mehlhorn. 2016. Computing real roots of real polynomials. *J. of Symb. Comp.* 73 (2016), 46–86.

[20] Arnold Schönhage. 1982. The fundamental theorem of algebra in terms of computational complexity. *Manuscript. Univ. of Tübingen, Germany* (1982).

[21] Juan Xu and Chee Yap. 2019. Effective subdivision algorithm for isolating zeros of real systems of equations, with complexity analysis. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*. 355–362.