Root-Squaring for Root-Finding

Pedro Soto
The Graduate Center, CUNY
New York, New York, USA
psoto@gradcenter.cuny.edu

Soo Go
The Graduate Center, CUNY
New York, New York, USA
sgo@gradcenter.cuny.edu

ABSTRACT

We revisit the classical root-squaring formula of Dandelin-Lobachevsky-Graeffe for polynomials and find new interesting applications to root-finding.

CCS CONCEPTS

• **Computing methodologies** → **Hybrid symbolic-numeric methods**.

KEYWORDS

symbolic-numeric computing, root finding, polynomial algorithms, computer algebra

ACM Reference Format:
Pedro Soto and Soo Go. 2022. Root-Squaring for Root-Finding. In *Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC '22).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

We revisit the famous methods simaltanously discovered by Dandelin, Lobachevsky, and Graeffe (See [12] for a history of this problem) and and make furhter progress by applying this indentity to the problem of approximating the root radius of a polynomial. This is a key step in many sub-division based root finding algorithms. One of the novelties of our algorithm is the Rational Root Tree Algorithm, *i.e.,* Alg. 1, which allows one to compute the angles of the iterated 2^{th} roots in Eq. 3 with exact percision over the rationals; thus, greatly reducing the numerical stability Eq. 3 since, as we will see in Sec. 5 (as well as Thm. 4.1), the most numerically unstable steps in Eq. 3 is the computation of the roots $x^{\frac{1}{2^m}}$. Another advantage of our algorithm is that it assumes the blackbox polynomial model; *i.e.,*, our algorithm works under the stronger assumption that one only compute $p(x)$ via some oracle. We give both theoritical gaurentess in Sec. 5 and empirical evidence in Sec. 6 that our algorithm performs well.

ISSAC '22, July 04–07, 2022, Lille, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 RELATED WORKS

The main method for rootfinding by root-squaring was simaltanously discovered by Dandelin, Lobachevsky, and Graeffe in the 19^{th} century (See [12]). Two of the first works to consider algorithms based on the formulae for modern computers were [9] and [10]; the later of which gave explicit pseudocode for the a rootfinding algorithm that uses Eq. 1. The work [15] also considers rootfinding using Eq. 1 and further state that the DLG method becomes very unstable for $\ell > 2$; by contrast, as we will see in Sec. 6, our algorithm performs reasonably well for ℓ as large as 12. The same authors went on design a variation of the DLG rootfinding algorithm that makes use of different renomalizations and other preprocessing tranformations for DLG root finding in [16], the work also proves convergence results for their DLG iterative algorithm. The authors of [3] consider DLG based algorithms for the solutions of fractional-order polynomials, *i.e.,* generalized polynomials that have rational exponents. In [20] the authors use apply the DLG formulae to root counting. The authors of [11] apply the DLG formulae to solving polynomials over finite fields. The authors of [2] apply DLG iterations to the benchmark problem (*i.e.,*, the isolsation of the roots of a polynomials, See Sec.3) to improve upon the record bound in [19] which at the time was unborken for 14 years. This short survey illustrates that the the application of the DLG to root finding (while more than a centuary old) is still an interesting research topic. One of the novelties and improvements of our algorithms is to consider the identies given by Eq. 2, instead of the usual Eq.1, to approximating the root radius. Furthermore, we show that, contrary to intuition, our algorithm is numerical stable and well behaved when taking the limit of Eq. 2 to 0.

3 BACKGROUND AND MOTIVATION

3.1 Dandelin, Lobachevsky, and Graeffe's Formaulae

The common procedure for root radii approximation, based on the classical technique of recursive root-squaring, is to first make an input polynomial $p(x)$ monic by scaling it and/or the variable x and then apply k DLG (that is, Dandelin's aka Lobachevsky's or Gräffe's) root-squaring iterations (cf. [12]),

$$p_0(x) = \frac{1}{p_d}p(x),\ p_{i+1}(x) = (-1)^d p_i(\sqrt{x})p_i(-\sqrt{x}),\ i = 0, 1, \ldots.\ell \tag{1}$$

for a fixed positive integer ℓ (see Remark 1 below). The ith iteration squares the roots of $p_i(x)$ and consequently the root radii from the origin, as well as the isolation of the unit disc $D(0, 1)$. Then one approximates the ratio ρ_+/ρ_-, the new scaled ratio of the root radii, for the polynomial $p_\ell(x)$ within a factor of γ and readily recovers the approximation of this ratio for $p_0(x)$ and $p(x)$ within a factor of $\gamma^{1/2^\ell}$.

Given the coefficients of $p_i(x)$ we can reduce the ith root-squaring iteration, that is, the computation of the coefficients of $p_{i+1}(x)$, to polynomial multiplication and perform it in $O(d\log(d))$ arithmetic operations. Unless the positive integer ℓ is small, the absolute values of the coefficients of $p_\ell(x)$ vary dramatically, and realistically one should either stop because of severe problems of numerical stability or apply the stable algorithm by Gregorio Malajovich and Jorge P. Zubelli [15], which performs a single root-squaring at arithmetic cost of order d^2.

For black box polynomials p, however, we apply DLG iterations without computing the coefficients, and the algorithm turns out to be quite efficient: for ℓ iterations evaluate $p(x)$ at 2^ℓ equally spaced points on a circle and obtain the values of the polynomial $p_\ell(x) = \prod \left(x - x_j^{2^\ell}\right)$ at these 2^ℓ points.

Furthermore, we evaluate the ratio $p'(x)/p(x) = p_0'(x)/p_0(x)$ at these points by applying the recurrence

$$\frac{p_{i+1}'(x)}{p_{i+1}(x)} = \frac{1}{2\sqrt{x}}\Big(\frac{p_i'(\sqrt{x}\,)}{p_i(\sqrt{x}\,)} - \frac{p_i'(-\sqrt{x}\,)}{p_i(-\sqrt{x}\,)}\Big),\ i = 0, 1, \dots \tag{2}$$

Recurrences (1) and (2) reduce evaluation of $p_\ell(c)$ to the evaluation of $p(c)$ at $q = 2^\ell$ points $c^{1/q}$ and for $c \neq 0$ reduce evaluation of the ratio $p_\ell'(x)/p_\ell(x)$ at $x = c$ to the evaluation of the ratio $p'(x)/p(x)$ at the latter $q = 2^\ell$ points $x = c^{1/q}$. We will see that we can apply recurrence (2) to support fast convergence to the convex hull of the roots.

For $x = 0$ recurrence (2) can be specialized as follows:

$$\frac{p_\ell'(0)}{p_\ell(0)} = \Big(\frac{p_{\ell-1}'(x)}{p_{\ell-1}(x)}\Big)'_{x=0} = \Big(\frac{p'(x)}{p(x)}\Big)^{(\ell)}_{x=0},\ \ell = 1, 2, \dots \tag{3}$$

Notice an immediate extension:

$$\frac{p_\ell^{(h)}(0)}{p_\ell(0)} = \prod_{g=1}^{h}\Big(\frac{p^{(g)}(x)}{p^{(g-1)}(x)}\Big)^{(\ell)}_{x=0},\ h = 1, 2, \dots. \tag{4}$$

Equations (3) and more generally (4) enable us to strengthen upper estimates in Eq. 5 & Eq. 6 and more generally Eq. 15 for root-radii $r_j(0,p)$ at the origin because $r_j(0,p_\ell) = r_j(0,p)^{2^\ell}$ for $j = 1, \dots, d$ (see Eq. 5 & Eq. 6); we can approximate the higher order derivatives $\Big(\frac{p^{(g)}(x)}{p^{(g-1)}(x)}\Big)^{(\ell)}$ at $x = 0$ by following Remark. 2. Besides the listed applications of root-squaring, one can apply DLG to randomized exclusion tests for sparse polynomial. One can apply root-squaring $p(x) \mapsto p_\ell(x)$ to improve the error bound for the approximation of the power sums of the roots of $p(x)$ in the unit disc $D(0,1)$ by Cauchy sums, but the improvement is about as much and at the same additional cost as by increasing the number q of points of evaluation of the ratio $\frac{p'}{p}$.

Remark 1. *One can approximate the leading coefficient p_d of a black box polynomial $p(x)$. This coefficient is not involved in recurrence (2), and one can apply recurrence (1) by using a crude approximation to p_d and if needed can scale polynomials $p_i(x)$ for some i.*

3.2 Extremal root radii

We cover some known estimates for extremal root radii in Sec. 3.3. The most used estimates (Eq. 15 for $i = 1$ and $c = 0$) readily follow from straightforward algbraic manipluations of $p_{\rm rev} := p_d + p_{d-1}x^1 + \dots + p_0x^d$:

$$r_d(c,p) \le d\left|\frac{p(c)}{p'(c)}\right| \tag{5}$$

and

$$r_1(c,p) \ge \left|\frac{p'_{\rm rev}(c)}{dp_{\rm rev}(c)}\right|. \tag{6}$$

We strengthen these estimates in the case of $c = 0$ by recalling DLG rootsquaring iterations Eq. 2 of Sec. 3.1 and Equation 3. This immediately implies the following extension of Eq. 5 & Eq. 6 for $c = 0$ and all non-negative integers ℓ :

$$r_d(0,p)^{2^\ell} \le d\left|\Big(\frac{p'(x)}{p(x)}\Big)^{(\ell)}_{x=0}\right|^{-1} \tag{7}$$

and

$$r_1(0,p)^{2^\ell} \ge d^{-1}\left|\Big(\frac{p'_{\rm rev}\,(x)}{p_{\rm rev}\,(x)}\Big)^{(\ell)}_{x=0}\right| \tag{8}$$

Remark 2. *Given a complex c and a positive integer ℓ one can approximate the values at $x = c$ of $(p(x)/p'(x))^{(\ell)}$ for a black box polynomial $p(x)$ by using divided differences, by extending expressions $p^{(i+1)}(c) = \lim_{x\to c}\frac{p^{(i)}(x)-p^{(i)}(c)}{x-c}$ for $i = 0, 1, \dots, \ell-1$ and based on the mean value theorem [7].*

For $\ell = 1$ one can instead apply the algorithm supporting the following elegant theorem and implicit in its constructive proof; the complexity of straightforward recursive extension to $\ell = 2, 3, \dots$ increases exponentially in ℓ.

Theorem 3.1. *Given an algorithm that evaluates a black box polynomial $p(x)$ at a point x over a field $\mathcal{K}$ of constants by using A additions and subtractions, S scalar multiplications (that is, multiplications by elements of the field $\mathcal{K}$), and M other multiplications and divisions, one can extend this algorithm to the evaluation at x of both $p(x)$ and $p'(x)$ by using $2A + M$ additions and subtractions, $2S$ scalar multiplications, and $3M$ other multiplications and divisions.*

Proof. [14], [1] proves the theorem for any function $f(x_1, \dots, x_s)$ that has partial derivatives in all its s variables $x_1, \dots, x_s$. □

Next we prove that estimates 7 and 8 are extremely poor for worst case inputs.

Theorem 3.2. *The ratios $\left|\frac{p(0)}{p'(0)}\right|$ and $\left|\frac{p_{\rm rev}(0)}{p'_{\rm rev}(0)}\right|$ are infinite for $p(x) = x^d - h^d$, while $r_d(c,p) = r_1(c,p) = r_d\,(c, p_{rev}\,) = r_1\,(c, p_{rev}\,) = |h|$.*
Proof. Observe that the roots $x_j = h\exp\left(\frac{(j-1)\mathbf{i}}{2\pi d}\right)$ of $p(x) = x^d - h^d$ for $j = 1, 2, \dots, d$ are the d th roots of unity up to scaling by h.

Clearly, the problem persists at the points x where $p'(x)$ and $p'_{\rm rev}\,(x)$ vanish; rotation of the variable $\mathcal{R}_a : p(x) \mapsto t(x) = p(ax)$ for $|a| = 1$ does not fix it but shifts $\mathcal{T}_c : p(x) \mapsto t(x) = p(x-c)$ for $c \neq 0$ can fix it, thus enhancing the power of estimates 7 and 8. Furthermore,

$$\frac{1}{r_d(c,p)} \le \frac{1}{d}\left|\frac{p'(c)}{p(c)}\right| = \frac{1}{d}\left|\sum_{j=1}^{d}\frac{1}{c-x_j}\right|$$

by virtue of 5 & 6, and so the approximation to the root radius $r_d(c,p)$ is poor if and only if severe cancellation occurs in the summation of the d roots, and similarly for the approximation of $r_1(c,p)$. Such cancellations only occur for a narrow class of polynomials $p(x)$, with a low probability under random root models, although it occurs for the polynomials $p(x) = x^d - h^d$ of Thm. 5 and with high probability for random polynomials of [8].

3.3 Classical estimates for extremal root radii

Next we recall some non-costly estimates known for the extremal root radii $r_1 = r_1(0,p)$ and $r_d = r_d(0,p)$ in terms of the coefficients of p (cf. [13], [17], and [21]) and the two parameters

$$\tilde{r}_- := \min_{i\geq 1}\left|\frac{p_0}{p_i}\right|^{\frac{1}{i}}, \tilde{r}_+ := \max_{i\geq 1}\left|\frac{p_{d-i}}{p_d}\right|^{\frac{1}{i}} \tag{9}$$

These bounds on r_1 and r_d hold in dual pairs since $r_1(0,p)r_d(0,p_{\text{rev}}) = 1$. Furthermore, we have that

$$\frac{1}{d}\tilde{r}_+ \leq r_1 < 2\tilde{r}_+, \frac{1}{2}\tilde{r}_- \leq r_d \leq d\tilde{r}_-, \tag{10}$$

$$\tilde{r}_+\sqrt{\frac{2}{d}} \leq r_1 \leq \frac{1+\sqrt{5}}{2}\tilde{r}_+ < 1.62\tilde{r}_+ \text{if } p_{d-1} = 0, \tag{11}$$

$$0.618\tilde{r}_- < \frac{2}{1+\sqrt{5}}\tilde{r}_- \leq r_d \leq \sqrt{\frac{d}{2}}\tilde{r}_- \text{if } p_1 = 0, \tag{12}$$

$$r_1 \leq 1 + \sum_{i=0}^{d-1}\left|\frac{p_i}{p_d}\right|, \frac{1}{r_d} \leq 1 + \sum_{i=1}^{d}\left|\frac{p_i}{p_0}\right|. \tag{13}$$

$M(p) := |p_d|\max_{j=1}^{d}\{1,|x_j|\}$ is said to be the Mahler measure of p, and so $M(p_{\text{rev}}) := |p_0|\max_{j=1}^{d}\left\{1,\frac{1}{|x_j|}\right\}$. It holds that

$$r_1^2 \leq \frac{M(p)^2}{|p_d|} \leq \max_{i=0}^{d-1}\left|\frac{p_i}{p_d}\right|^2, \frac{1}{r_d^2} \leq \frac{M(p_{\text{rev}})^2}{|p_0|^2} \leq \max_{i=1}^{d}\left|\frac{p_i}{p_0}\right|^2 \tag{14}$$

Theorem 167 supports very fast approximation of all root radii of p at the origin at a very low cost, which complements estimates 9, 10 11, 12, 13, and 14.

One can extend all these bounds to the estimates for the root radii $r_j(c,p)$ for any fixed complex c and all j by observing that $r_j(c,p) = r_j(0,t)$ for the polynomial $t(x) = p(x-c)$ and applying Taylor's shift; *i.e.,* applying the mapping $\mathcal{S}_{c,\rho} : p(x) \mapsto p\left(\frac{x-c}{\rho}\right)$.

The algorithms in Sec. 4 closely approximate root radii $r_j(c,p)$ for a black box polynomial p and a complex point c at reasonably low cost, but the next well-known upper bounds on r_d and lower bounds on r_1 (cf. [13],[6],[18],[4], and [5]) are computed at even a lower cost, defined by a single fraction $\frac{p_0}{p_i}$ or $\frac{p_{d-i}}{p_d}$ for any i, albeit these bounds are excessively large for the worst case input.

Finally, we have that $r_d \leq \rho_{i,-} := \left(\binom{d}{i}\left|\frac{p_0}{p_i}\right|\right)^{\frac{1}{i}}, \frac{1}{r_1} \leq \frac{1}{\rho_{i,+}} := \left(\binom{d}{i}\left|\frac{p_d}{p_{d-i}}\right|\right)^{\frac{1}{i}}$ and therefore, since $p^{(i)}(0) = i!p_i$ for all $i > 0$, that

$$r_d \leq \rho_{i,-} = \left(i!\binom{d}{i}\left|\frac{p(0)}{p^{(i)}(0)}\right|\right)^{\frac{1}{i}}, \frac{1}{r_1} \leq \frac{1}{\rho_{i,+}} = \left(i!\binom{d}{i}\left|\frac{p_{\text{rev}}(0)}{p_{\text{rev}}^{(i)}(0)}\right|\right)^{\frac{1}{i}} \tag{15}$$

for all i; from which we obtain relations 5 and 6 for $i = 1$.

4 ALGORITHM DESIGN

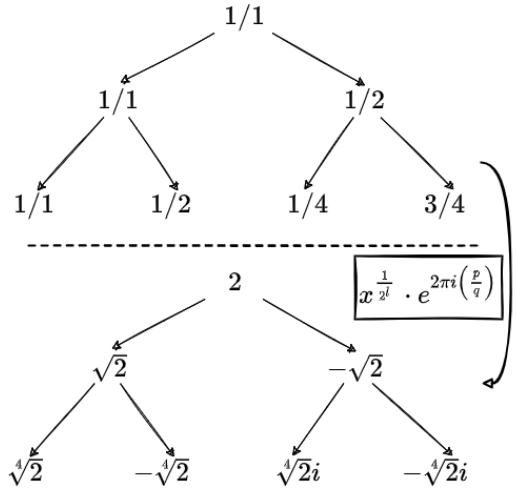

Figure 1: The upper tree depicts the steps of CIRCLE_ROOTS_RATIONAL_FORM(p,q,l) in Alg.1 for $l = 2$, $p = 1$, and $q = 1$. The lower tree depicts the steps of ROOTS(r,t,u,l) in Alg.2 for $r = 2$, $l = 2$, $p = 1$, and $q = 1$

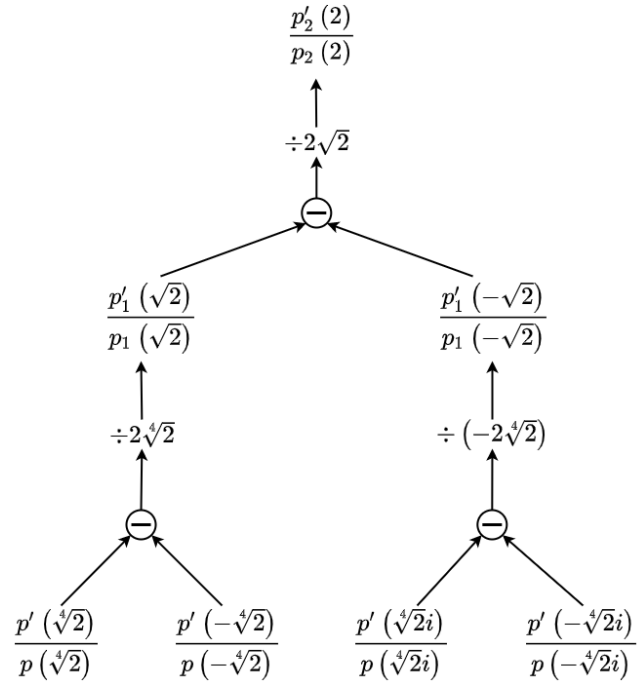

Figure 2: The steps of DLG_RATIONAL_RORM(p,p',r,t,u,l) in Alg.3 for $r = 2$, $l = 2$, $t = 1$, and $u = 1$.

In this section give the design of the general algorithm for approximating the root radius given by Eq. 3. There are two main steps:

1) going down the rational root tree, *i.e.*, performing Alg. 1, and 2) going up the rational root tree, *i.e.,* performing Alg. 3. The going-down is depicted in Fig. 1 and the going-up is depicted in Fig. 2. The other procedures are simple bookkeeping/preprocessing steps in between the two main going-down/going-up steps. In particular we 1) first convert a complex number x into polar coordinates r, θ where $\theta \approx \frac{p}{q} = \frac{p}{2^\epsilon}$, *i.e.,* perform Alg. 4, 2) we then perform the first going-down pass which gives the rational angles for the roots of x in fraction form, *i.e.,* perform Alg. 1, 3) we then perform the second pass of the going down algorithm where we compute the values $|x|^{\frac{1}{2^m}} \exp(2\pi i \frac{p}{q})$ at the m^{th} level, and 4) finally, we then compute the values given by Eq. 2 going back up the rational root tree.

Algorithm 1 Circle_Roots_Rational_Form(p, q, l)

```
if p%q == 0 then
  r, s := (1,1)
else
  r, s := (p,2q)
end if
if r%s == 0 then
  t, u := (1,2)
else
  t, u := (2r + s, 2s)
end if
if l == 1 then
  return [(r, s),(t, u)]
else if l != 0 then
  left := Circle_Roots_Rational_Form(r, s, l − 1)
  right := Circle_Roots_Rational_Form(t, u, l − 1)
  return left ∪ right
else
  return [(p, q)]
end if
```

The intuition behind Algorithm. 1 is that the square root operation satisfies

$$p\%q \neq 0 \implies \sqrt{\exp\left(2\pi i \frac{p}{q}\right)} = \exp\left(2\pi i \frac{p}{2q}\right) \tag{16}$$

and

$$p\%q = 0 \implies \sqrt{\exp\left(2\pi i \frac{1}{1}\right)} = \exp\left(2\pi i \frac{1}{1}\right) = 1, \tag{17}$$

and the negation operation satisfies

$$p\%q \neq 0 \implies -\exp\left(2\pi i \frac{r}{s}\right) = \exp\left(2\pi i \frac{2r+s}{2s}\right) \tag{18}$$

and

$$p\%q = 0 \implies \sqrt{\exp\left(2\pi i \frac{1}{1}\right)} = \exp\left(2\pi i \frac{1}{2}\right) = -1; \tag{19}$$

thus, the first four lines of Algorithm. 1 computes the (angle of the) positive square root, $\sqrt{x}$, of complex number on the unit circle and the next four lines Alg. 1 computes the (angle of the) negative square root, $-\sqrt{x}$. Therefore, we have:

Theorem 4.1. *For a complex number, x, with a rational angle,* i.e., $x = |x| \exp\left(2\pi i \frac{p}{q}\right)$, *Alg. 1 correctly computes the roots in Eq. 2; in particular; for a rational angle in fractional form, it does so with exact precision.*

Proof. Equation 16, 17, 18, and 19 give the base case and the theorem follows by a straightforward induction. □

Algorithm 2 and Algorithm 4 are both striaghtforward; thus, we do not need to explain them in detail and instead explain the intuition behind Alg. 3.

Algorithm 2 Roots(r, t, u, l)

```
root_tree = Circle_Roots_Rational_Form(p, q, l)
circ_root = [exp (2 · π · i · r/s) for r, s in root_tree]
roots =[ ∛r·root for root in circ_root]
return roots
```

Algorithm 3 DLG_Rational_Form(p, p', r, t, u, l)

```
root := Roots(r, t, u, l)
for r_i ∈ root do
  base_step[i] := p'(r_i)/p(r_i)
end for
diff[0] := base_step
for i ≤ l do
  for j ≤ 2^{l−i−1} do
    diff[i + 1][j]:=1/2 (diff[i][2j]−diff[i][2j+1])/root[2j]
    root = roots(r, t, u, l − 1 − i)
  end for
end for
return diff[l][0]
```

Algorithm 3 is a straightforward dynamic programming on Equation 2; thus, by Thm. 4.1, we have that Alg. 3 correctly computes Eq. 2. To be precise Alg. 3 does the following: 1) it computes the last layer of the recursion Eq. 2 (*i.e.,* it computes $\frac{p'(x^{\frac{1}{2^l}})}{p(x^{\frac{1}{2^l}})}$) and then 2) it recursively applies Eq. 2 via dynamic programming until it finally computes $\frac{p'_t(x)}{p_t(x)}$ which is the desired quantity. The former is given by the line "base_step$[i]$:= $\frac{p'(r_i)}{p(r_i)}$" and the latter is given by the line "diff$[i+1][j]$:=$\frac{1}{2}\frac{\text{diff}[i][2j]-\text{diff}[i][2j+1]}{\text{root}[2j]}$" in Alg. 3; the desired output (*i.e.,* $\frac{p'(x^{\frac{1}{2^l}})}{p(x^{\frac{1}{2^l}})}$)is given by the line "diff$[l][0]$".

Algorithm 4 DLG(p, p', l, x, ϵ)

```
angle := 1/(2πi) log(x)
u := 2^ϵ
t := (angle · u)%1
r := |x|
return DLG_Rational_Form(p, p', r, t, u, l)
```

The final step in the algorithm is to use Algorithm 4 to compute root radius approximations r_d and r_1. The procedure is given by Algorithm. 5. The rationale for generating a random x is that there

Algorithm 5 DLG_Root_Radius$(p, p', p_{\text{rev}}, p'_{\text{rev}}, l, \epsilon, \delta)$

Uniformly Randomly Generate x in the unit circle
$d := \deg(p)$
$r_{\min} := d/\text{DLG}(p, p', l, x \cdot 2^{-\delta}, \epsilon)$
$r_{\max} := \text{DLG}(p_{\text{rev}}, p'_{\text{rev}}, l, x \cdot 2^{-\delta}, \epsilon)/d$

may be roots close to 0 and thus by taking the limit in certain directions we avoid these possible poles; in particular, we have that

$$\lim_{\delta,\epsilon\to\infty} \text{DLG}(p, p', l, x\cdot 2^{-\delta}, \epsilon) = \frac{p'_\ell(0)}{p_\ell(0)} = \left(\frac{p'_{\ell-1}(x)}{p_{\ell-1}(x)}\right)'_{x=0} = \left(\frac{p'(x)}{p(x)}\right)^{(\ell)}_{x=0}, \tag{20}$$

for any x, if $p(0) \neq 0$ (see Thm 5.3).

5 THEORETICAL ANALYSIS

We give some theoritical guarentess; Theorem 5.1 and Theorem 5.2 give compputational complexity bounds and Theorem 5.3 proves the correctness of Alg. 5.

Theorem 5.1. *Algorithm 3 performs q floating point subtractions, divisions, and multiplications and q applications of* sin *and* cos*, where $q = 2^l$; furthermore, Algorithm 3 performs at most Cq integer additions, "multiplications-by-2", and $\%2^\epsilon$ (*i.e., *mod* 2^ϵ*) operations, where $C = 1, 3, 2$ respectively.*

Proof. Looking at Fig. 1 and Fig. 2 it is straighforward to see that the computational tree for the Alg. 3 is a binary tree with $2^l = q$ nodes; the proof for the constants $C = 1, 3, 2$ is equally straighforward inspection of the operations performed in Alg. 1. □

Theorem 5.2. *The Cq integer additions, "multiplications-by-2", and $\%2^\epsilon$ (*i.e., *mod* 2^ϵ*) operations in Algorithm. 3 have negligble overhead; more precisely, integer addtions are always additions of $2\epsilon \log \ell$-bit integers and "multiplications-by-2" and $\%2^\epsilon$ (*i.e., *mod* 2^ϵ*) operations have constant time overhead.*

Proof. Since Alg. 4 always passes in a denominator which is a power of two all of the integer % and · operations are in fact "multiplications-by-2", and $\%2^\epsilon$ (*i.e.,* mod 2^ϵ) operations by a straightforward proof similar to the one in Algorithm. 3. Thus these operation are essentially constant overhead bit shift operations on a computing machine with binary words. Since $p\%r$ always reduces $p \mapsto 1$, we have that whenever an overflow of more than $\log r$-bits happens in Alg. 1 it gets converted to an $\log r$-bit integer; thus, it suffices to prove that $\log r$ is bounded by $\epsilon \log \ell$. However, this is once again an easy induction on the binary computation tree; since this tree has depth ℓ we see that any denominator is bouonded by $\epsilon \log \ell$ by a simple induction. □

Theorem 5.3. *With probability 1, Algorithm 5 computes the bounds given by Equation 5 and Equation 6.*

Proof. Prove □

6 EXPERIMENTAL RESULTS

In this section we give the results of our experiments where we use DLG iterations to estimate the extremal root radii of the polynomial test suite included as part of the MPSolve package. To measure accuracy, we compute the relative error of the estimates in comparison to the extremal roots obtained by MPSolve using the formula

$$\text{relative error}_{r_i} = \frac{|\tilde{r_i} - r_i|}{|r_i|},$$

where r_i is the root radius found using MPSolve and $\tilde{r_i}$ our estimate.

All experiments were run using Python 3.7.7 on MacOS 11.6.1 with 2.8 GHz Dual-Core Intel Core i5 with 8 GB memory. The entries '-' in the tables indicate the test was terminated before completion.

7 CONCLUSION

REFERENCES

[1] Walter Baur and Volker Strassen. 1983. The complexity of partial derivatives. *Theoretical computer science* 22, 3 (1983), 317–330.
[2] Ruben Becker, Michael Sagraloff, Vikram Sharma, and Chee-Keng Yap. 2018. A near-optimal subdivision algorithm for complex root isolation based on the Pellet test and Newton iteration. *J. Symb. Comput.* 86 (2018), 51–96.
[3] Stanislaw Bialas and Henryk Górecki. 2010. Generalization of Vieta's formulae to the fractional polynomials, and generalizations the method of Graeffe-Lobachevsky. *Bulletin of The Polish Academy of Sciences-technical Sciences* 58 (2010), 624–629.
[4] Dario Andrea Bini and Giuseppe Fiorentino. 2000. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms* 23, 2 (2000), 127–173.
[5] Dario A Bini and Leonardo Robol. 2014. Solving secular and polynomial equations: A multiprecision algorithm. *J. Comput. Appl. Math.* 272 (2014), 276–292.
[6] Carsten Carstensen. 1991. Inclusion of the roots of a polynomial based on Gerschgorin's theorem. *Numer. Math.* 59, 1 (1991), 349–360.
[7] Carl de Boor. 1995. Divided Differences.
[8] Paul Erdos and Pál Turán. 1950. On the distribution of roots of polynomials. *Annals of mathematics* (1950), 105–119.
[9] A. A. Grau. 1963. On the Reduction of Number Range in the Use of the Graeffe Process. *J. ACM* 10, 4 (oct 1963), 538–544. https://doi.org/10.1145/321186.321198
[10] A. A. Grau. 1965. Algorithm 256: Modified Graeffe Method [C2]. *Commun. ACM* 8, 6 (jun 1965), 379–380. https://doi.org/10.1145/364955.364974
[11] Bruno Grenet, Joris van der Hoeven, and Grégoire Lecerf. 2015. Deterministic root finding over finite fields using Graeffe transforms. *Applicable Algebra in Engineering, Communication and Computing* 27 (2015), 237–257.
[12] Alston S. Householder. 1959. Dandelin, Lobacevskii, or Graeffe. *The American Mathematical Monthly* 66, 6 (1959), 464–466. http://www.jstor.org/stable/2310626
[13] Movlud Kerimovich Kerimov. 1977. Applied and computational complex analysis. Vol. 1. Power series, integration, conformal mapping, location of zeros: Henrici P. xv+ 682 pp., John Wiley and Sons, Inc., New York–London, 1974. Book review. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 17, 5 (1977), 1330–1331.
[14] Seppo Linnainmaa. 1976. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics 1976 16:2* 16 (1976), 146–160. Issue 2. https://doi.org/10.1007/BF01931367
[15] Gregorio Malajovich and Jorge P. Zubelli. 2001. On the Geometry of Graeffe Iteration. *J. Complex.* 17 (2001), 541–573.
[16] Gregorio Malajovich and Jorge P. Zubelli. 2001. Tangent Graeffe iteration. *Numer. Math.* 89 (2001), 749–782.
[17] Maurice Mignotte and Doru Stefanescu. 1999. *Polynomials: An algorithmic approach.* Springer.
[18] Victor Y Pan. 2000. Approximating complex polynomial zeros: modified Weyl's quadtree construction and improved Newton's iteration. *journal of complexity* 16, 1 (2000), 213–264.
[19] Victor Y Pan. 2002. Univariate polynomials: nearly optimal algorithms for numerical factorization and root-finding. *Journal of Symbolic Computation* 33, 5 (2002), 701–733.
[20] Joris van der Hoeven. 2011. Efficient root counting for analytic functions on a disk.
[21] Chee-Keng Yap et al. 2000. *Fundamental problems of algorithmic algebra.* Vol. 49. Oxford University Press Oxford.

Table 1: Experimental Data for chebyshev

DEGREE	ℓ	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.155	0.0939	0.27	[0.0785, 0.997]
40	5	616	332	0.0981	0.0589	1.13	[0.0393, 0.999]
80	6	617	334	0.0593	0.0353	5.26	[0.0196, 1.0]
160	7	617	334	2.71	0.0205	20.88	[0.00982, 1.0]
320	8	617	334	37.6	0.465	88.4	[0.00491, 1.0]

Table 2: Experimental Data for chrma

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
21	4	616	332	0.215	0.139	0.22	[1.0, 3.17]
85	6	617	334	0.0369	0.0452	5.63	[1.0, 3.25]
341	8	617	334	0.717	0.547	105.48	[0.884, 3.41]

Table 3: Experimental Data for chrma_d

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.16	0.175	0.26	[1.3, 3.01]
84	6	617	334	0.0548	0.00507	5.53	[1.1, 3.06]
340	8	617	334	0.658	0.699	93.82	[0.741, 3.11]

Table 4: Experimental Data for chrmc

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
22	4	616	332	0.211	0.138	0.3	[1.0, 3.03]
342	8	617	334	0.718	0.279	94.64	[0.897, 4.13]

Table 5: Experimental Data for chrmc_d

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
11	3	616	332	0.251	0.242	0.05	[1.27, 2.8]
43	5	616	332	0.101	0.0996	1.23	[1.02, 2.97]
171	7	617	334	0.268	1.13	22.05	[0.715, 3.07]
683	9	619	338	0.164	0.257	374.69	[0.519, 3.1]

Table 6: Experimental Data for curz

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.156	0.854	0.21	[0.452, 1.15]
40	5	616	332	0.199	0.776	1.13	[0.379, 1.26]
80	6	617	334	0.086	0.227	5.49	[0.318, 1.34]
160	7	617	334	0.0385	0.509	20.79	[0.271, 1.38]

Table 7: Experimental Data for easy

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
100	6	617	334	0.12	0.0809	6.35	[0.949, 0.98]
200	7	617	334	0.068	0.047	25.57	[0.971, 0.99]
400	8	617	334	0.0381	0.0265	104.47	[0.983, 0.995]
1600	10	619	338	0.01	0.00	2228.22	[0.995, 0.999]
3200	11	619	338	0.00	-	-	[0.997, 0.999]

Table 8: Experimental Data for exp

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
50	5	616	332	3.76	0.126	1.48	[14.9, 39.4]
100	6	617	334	0.367	0.0598	7.37	[28.9, 83.9]
200	7	617	334	0.714	0.839	28.22	[56.8, 176.0]
400	8	617	334	0.965	0.985	107.96	[113.0, 365.0]

Table 9: Experimental Data for geom1

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
10	3	616	332	0.334	0.25	0.05	[1.0, 1.0E+18]
15	3	616	332	0.403	1.0	0.08	[1.0, 1.0E+28]
20	4	616	332	0.206	1.0	0.28	[1.0, 1.0E+38]
40	5	616	332	0.122	1.0	1.18	[1.0, 1.0E+78]

Table 10: Experimental Data for geom2

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
10	3	616	332	0.334	0.25	0.06	[1.0E-18, 1.0]
15	3	616	332	8.09E+4	0.287	0.06	[1.0E-28, 1.0]
20	4	616	332	2.63E+26	0.171	0.23	[1.0E-38, 1.0]
40	5	616	332	1.61E+72	0.109	1.14	[1.0E-78, 1.0]

Table 11: Experimental Data for geom3

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
10	3	616	332	0.334	0.25	0.05	[9.54E-7, 0.25]
20	4	616	332	2.41	0.171	0.24	[9.09E-13, 0.25]
40	5	616	332	1.95E+18	0.109	1.15	[8.27E-25, 0.25]
80	6	617	334	1.83E+45	0.0662	5.73	[6.84E-49, 0.25]

Table 12: Experimental Data for `geom4`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
10	3	616	332	0.334	0.25	0.05	[4.0, 1.05E+6]
20	4	616	332	0.206	0.707	0.27	[4.0, 1.1E+12]
40	5	616	332	0.122	1.0	1.15	[4.0, 1.21E+24]
80	6	617	334	0.0709	1.0	5.28	[4.0, 1.46E+48]

Table 13: Experimental Data for `hermite`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.155	0.129	0.28	[0.245, 5.39]
40	5	616	332	0.0981	0.0876	1.31	[0.175, 8.1]
80	6	617	334	0.0593	0.0555	5.97	[0.124, 11.9]
160	7	617	334	0.0348	0.0335	23.56	[0.0877, 17.2]
320	8	617	334	2.08	0.785	88.88	[0.062, 24.7]

Table 14: Experimental Data for `kam1`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
7	2	615	331	0.368	1.0	0.01	[3.0E-12, 15.8]
7	2	615	331	0.368	1.0	0.01	[3.0E-40, 1.0E+4]
7	2	615	331	2.37E+93	1.0	0.01	[3.0E-140, 1.0E+14]

Table 15: Experimental Data for `kam2`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
9	3	616	332	0.107	1.0	0.03	[1.73E-6, 251.0]
9	3	616	332	0.107	1.0	0.03	[1.73E-20, 1.0E+8]
9	3	616	332	4.23E+46	1.0	0.03	[1.73E-70, 1.0E+28]

Table 16: Experimental Data for `kam3`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
9	3	616	332	0.107	1.0	0.03	[1.73E-6, 251.0]
9	3	616	332	0.107	1.0	0.03	[1.73E-20, 1.0E+8]
9	3	616	332	4.23E+46	1.0	0.03	[1.73E-70, 1.0E+28]

Table 17: Experimental Data for `kir1`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
44	5	616	332	4.41E-5	0.000443	1.28	[0.5, 0.5]
84	6	617	334	2.29E-5	0.000464	2.9	[0.5, 0.5]
164	7	617	334	1.16E-5	0.000476	9.79	[0.5, 0.5]
8	3	616	332	0.000244	0.000244	0.02	[0.5, 0.5]

Table 18: Experimental Data for `kir1_mod`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
44	5	616	332	0.000983	0.00095	1.26	[0.5, 0.5]
84	6	617	334	0.00364	0.00364	2.99	[0.498, 0.502]
164	7	617	334	0.00734	0.00749	10.35	[0.496, 0.504]

Table 19: Experimental Data for `lagurerre`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.206	0.167	0.22	[0.0705, 66.5]
40	5	616	332	0.122	0.108	1.28	[0.0357, 142.0]
80	6	617	334	0.0709	0.0659	5.63	[0.018, 297.0]
160	7	617	334	3.09	0.954	22.21	[0.00901, 610.0]
320	8	617	334	41.4	0.996	89.64	[0.00451, 1.24E+3]

Table 20: Experimental Data for `lar1`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	7.06E+9	1.0	0.1	[3.73E-22, 1.0E+50]
200	7	617	334	9.69E+19	0.311	0.83	[3.73E-22, 41.0]

Table 21: Experimental Data for `legendre`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.155	0.0969	0.25	[0.0765, 0.993]
40	5	616	332	0.0981	0.0604	1.15	[0.0388, 0.998]
80	6	617	334	0.0593	0.0359	6.08	[0.0195, 1.0]
160	7	617	334	2.72	0.0637	19.97	[0.00979, 1.0]
320	8	617	334	37.7	0.456	81.12	[0.0049, 1.0]

Table 22: Experimental Data for `lsr`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
24	4	616	332	2.88E+8	1.0	0.29	[1.0E-20, 1.0E+20]
52	5	616	332	1.81E+14	1.0	0.25	[1.0E-20, 1.0E+10]
52	5	616	332	1.81E+34	1.0	0.22	[1.0E-40, 1.0E+20]
52	5	616	332	1.81E+74	1.0	0.2	[1.0E-80, 1.0E+40]
224	7	617	334	3.62E+18	1.0	9.8	[1.0E-20, 1.0E+20]
500	8	617	334	1.92E+3	1.0	6.79	[0.0001, 2.0E+4]
500	8	617	334	5.77E+3	0.995	3.27	[3.33E-5, 1.0E+3]
500	8	617	334	1.05	1.0	3.05	[0.0916, 1.0E+200]

Table 23: Experimental Data for `mand`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
1023	9	619	338	0.357	0.142	569.08	[0.321, 2.0]
127	6	617	334	0.0898	0.0488	8.58	[0.373, 2.0]

Table 24: Experimental Data for `mig1`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.126	1.0	0.08	[0.01, 2.26]
50	5	616	332	0.0157	0.987	1.8	[0.00999, 1.83E+3]
100	6	617	334	0.0563	1.0	0.5	[0.01, 1.15]
100	6	617	334	0.0183	1.0	4.05	[0.01, 7.92]
200	7	617	334	2.66	1.0	0.8	[0.01, 1.07]
200	7	617	334	2.59	0.999	9.37	[0.01, 2.33]
500	8	617	334	18.2	0.99	1.59	[0.01, 1.03]

Table 25: Experimental Data for `mult`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
15	3	616	332	0.29	0.189	0.12	[0.869, 1.07]
20	4	616	332	0.0782	0.475	0.24	[0.01, 2.68]
22	4	616	332	0.213	0.105	0.24	[1.0, 20.0]
68	6	617	334	0.0566	0.0556	4.59	[0.25, 2.24]

Table 26: Experimental Data for `nroots`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
50	5	616	332	5.18E+13	1.0	0.12	[1.0, 1.0]
100	6	617	334	7.06E+6	1.0	0.19	[1.0, 1.0]
200	7	617	334	2.69E+3	1.0	0.39	[1.0, 1.0]
400	8	617	334	50.5	0.981	0.79	[1.0, 1.0]
800	9	619	338	6.34	0.864	1.73	[1.0, 1.0]
1600	10	619	338	1.71	0.631	3.19	[1.0, 1.0]
3200	11	619	338	0.645	0.392	6.79	[1.0, 1.0]
6400	12	623	346	0.289	0.224	14.55	[1.0, 1.0]

Table 27: Experimental Data for `nrooti`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
50	5	616	332	4.94E+13	1.0	0.1	[1.0, 1.0]
100	6	617	334	7.07E+6	1.0	0.32	[1.0, 1.0]
200	7	617	334	2.69E+3	1.0	0.46	[1.0, 1.0]
400	8	617	334	50.5	0.981	0.85	[1.0, 1.0]
800	9	619	338	6.34	0.864	1.85	[1.0, 1.0]
1600	10	619	338	1.71	0.631	3.23	[1.0, 1.0]
3200	11	619	338	0.645	0.392	7.02	[1.0, 1.0]
6400	12	623	346	0.289	0.224	13.31	[1.0, 1.0]

Table 28: Experimental Data for `sendra`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
160	7	617	334	0.658	2.14	24.34	[0.987, 2.01]
20	4	616	332	0.283	0.158	0.25	[0.9, 2.05]
320	8	617	334	0.809	1.64	88.08	[0.994, 2.0]
40	5	616	332	0.159	0.101	1.25	[0.95, 2.02]
80	6	617	334	0.287	0.573	5.2	[0.975, 2.01]

Table 29: Experimental Data for `spiral`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
10	3	616	332	5.1E-7	1.21E-6	0.05	[1.0, 1.0]
15	3	616	332	3.49E-7	1.15E-6	0.07	[1.0, 1.0]
20	4	616	332	4.55E-7	1.3E-6	0.23	[1.0, 1.0]
25	4	616	332	3.67E-7	1.25E-6	0.29	[1.0, 1.0]
30	4	616	332	3.08E-7	1.21E-6	0.41	[1.0, 1.0]

Table 30: Experimental Data for `toep`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
128	7	617	334	0.0386	0.562	18.71	[1.31, 64.4]
256	8	617	334	0.0219	0.918	73.35	[1.34, 64.4]
128	7	617	334	0.0386	0.0272	17.68	[0.4, 13.2]
256	8	617	334	0.0225	0.599	72.66	[0.383, 13.2]

Table 31: Experimental Data for `wilk`

DEGREE	l	$e = -\log(\lvert x \rvert)$	MPMATH PRECISION	RELATIVE ERROR r_d	RELATIVE ERROR r_1	RUNTIME	MPSOLVE ROOT RADIUS
20	4	616	332	0.206	0.141	0.22	[1.0, 20.0]
30	4	616	332	0.237	0.0859	0.33	[1.0, 319.0]
40	5	616	332	0.122	0.0927	1.27	[1.0, 40.0]
80	6	617	334	0.0709	0.121	5.59	[1.0, 80.0]
160	7	617	334	0.0404	0.824	21.82	[1.0, 160.0]
320	8	617	334	0.0228	0.983	89.77	[1.0, 320.0]