# Fast computation of higher order derivatives of a black box function

Pedro Soto[1][0000−0002−7120−7362], Soo Go[2], and Victor Pan[3]

[1] The Graduate Center, CUNY New York, NY, USA psoto@gradcenter.cuny.edu
https://pedrojuansoto.github.io/
[2] The Graduate Center, CUNY New York, NY, USA sgo@gradcenter.cuny.edu
[3] Lehman College, CUNY New York, NY, USA victor.pan@lehman.cuny.edu

**Abstract.** **Keywords:** symbolic-numeric computing · root finding · polynomial algorithms · computer algebra.

## 1 Background and Motivation

Let $p(x)$ be a polynomial with *roots* $x_1, ..., x_d \in \mathbb{C}$ so that

$$p(x) = \sum_{i=0}^{d} p_i x^i = p_d \prod_{i=1}^{d} (x - x_i),\ p_d \neq 0 \tag{1}$$

The reverse polynomial $p_{\text{rev}}$ of $p(x)$ is defined as

$$p_{\text{rev}}(x) := x^d p\Big(\frac{1}{x}\Big) = \sum_{i=0}^{d} p_i x^{d-i},\ p_{\text{rev}}(x) = p_0 \prod_{j=1}^{d} \Big(x - \frac{1}{x_j}\Big) \text{ if } p_0 \neq 0. \tag{2}$$

### 1.1 Root-squaring: DLG and FG root-finders

DLG root-squaring iterations (cf. [5]) are defined by

$$p_0(x) = \frac{1}{p_d} p(x),\ p_{k+1}(x^2) = (-1)^d p_k(x) p_k(-x),\ k = 0, 1, \dots .\ell \tag{3}$$

for a fixed positive integer $\ell$.

TODO: description of the DLG root-finder, including the following key expressions:

Let the roots of equation $p(x) = 0$ satisfy

$$|x_1| \geq |x_2| \geq \cdots \geq |x_{d-i+1}| > \max_{j > d-i+1} |x_j|. \tag{4}$$

We use the fact

$$\frac{p_k'(0)}{p_k(0)} = \Big(\frac{p_{k-1}'(x)}{p_{k-1}(x)}\Big)'_{x=0} = \Big(\frac{p'(x)}{p(x)}\Big)^{(k)}_{x=0},\ k = 1, 2, \dots \tag{5}$$

Notice an immediate extension:

$$\frac{p_\ell^{(k)}(0)}{p_\ell(0)} = \prod_{g=1}^{k} \left( \frac{p^{(g)}(x)}{p^{(g-1)}(x)} \right)_{x=0}^{(\ell)}, \ k = 1, 2, \ldots. \tag{6}$$

FG iterations fixes a polynomial $q(x)$ and computes

$$q_0(x) = q(x), \ q_{i+1}(x^2) = ... \tag{7}$$

TODO: description of the FG root-finder, including the following root radius assumption

$$|x_1| > |x_2| > \cdots > |x_d|. \tag{8}$$

The companion paper [] presents the root-finders in more detail.

### 1.2 DLG iterations for black box polynomials

Given the coefficients of $p_i(x)$ we can reduce the $i$th 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 $\ell$ 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 [9], which performs a single root-squaring at arithmetic cost of order $d^2$.

For a black box polynomial $p(x)$, however, we apply DLG iterations without computing the coefficients, and the algorithm turns out to be quite efficient: for $\ell$ iterations evaluate $p(x)$ at $2^\ell$ 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 $2^\ell$ points.

### 1.3 Estimation of extremal root radii

If we do not know whether assumptions (4) and (8) hold, we can apply the following well-known bounds on the extremal root radii:

$$|x_d| \le d \left| \frac{p(0)}{p'(0)} \right| \text{ and } |x_1| \ge \left| \frac{p'_{\mathrm{rev}}(0)}{d \ p_{\mathrm{rev}}(0)} \right|. \tag{9}$$

We can deduce these bounds from the well-known expression

$$\frac{p'(x)}{p(x)} = \sum_{j=1}^{d} \frac{1}{x - x_j}, \tag{10}$$

which we can obtain by differentiating the equation $p(x) = p_d \prod_{j=1}^{d} (x - x_j)$.

By extending these bounds to the polynomial $p_k(x)$ of Eq. (3) we obtain that

$$|x_d|^{2^k} \leq d/\Big|\Big(\frac{p'(x)}{p(x)}\Big)^{(k)}_{x=0}\Big| \text{ and } |x_1|^{2^k} \geq \frac{1}{d}\Big|\Big(\frac{p'_{\rm rev}(x)}{p_{\rm rev}(x)}\Big)^{(k)}_{x=0}\Big|. \tag{11}$$

Under the assumptions (4) for $i = 2$ for $p(x)$ and for $p_{\rm rev}(x)$, respectively, the latter two bounds become sharp as $k$ increases, by virtue of (9), and next we argue informally that it tends to be sharp with a high probability under random root models. Indeed,

$$\frac{1}{|x_d|} \leq \frac{1}{d}\Big|\frac{p'(c)}{p(c)}\Big| = \frac{1}{d}\Big|\sum_{j=1}^{d}\frac{1}{c - x_j}\Big| \tag{12}$$

by virtue of (9), and so the approximation to the root radius $|x_d|$ 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 a cancellation only occurs for a narrow class of polynomials $p(x)$, with a low probability if we assume a random root model.

Next we prove, however, that estimates (9) and (11) are extremely poor for worst case inputs.

**Theorem 1.** *The ratios* $|\frac{p(0)}{p'(0)}|$ *and* $|\frac{p_{\rm rev}(0)}{p'_{\rm rev}(0)}|$ *are infinite for* $p(x) = x^d - h^d$ *and* $h \neq 0$*, while* $|x_d|) = |x_1| = |h|$.

*Proof.* Observe that the roots $x_j = h\exp(\frac{(j-1)\mathbf{i}}{2\pi d})$ of $p(x) = x^d - h^d$ for $j = 1, 2, \dots, d$ are the $d$th roots of unity up to scaling by $h$.

The problem persists for the root radius $r_d(w, p)$ where $p'(w)$ and $p'_{\rm rev}(w)$ vanish; rotation of the variable $p(x) \leftarrow t(x) = p(ax)$ for $|a| = 1$ does not fix it but shifts $p(x) \leftarrow t(x) = p(x - c)$ for $c \neq 0$ can fix it, thus *enhancing the power of estimates (9) and (11).*

### 1.4 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. [7], [10], and [12]) and the two parameters

$$\tilde{r}_- := \min_{i\geq 1}\Big|\frac{p_0}{p_i}\Big|^{\frac{1}{i}}, \tilde{r}_+ := \max_{i\geq 1}\Big|\frac{p_{d-i}}{p_d}\Big|^{\frac{1}{i}} \tag{13}$$

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

$$\frac{1}{d}\tilde{r}_+ \leq r_1 < 2\tilde{r}_+, \frac{1}{2}\tilde{r}_- \leq r_d \leq d\tilde{r}_-, \tag{14}$$

$$\tilde{r}_+\sqrt{\frac{2}{d}} \leq r_1 \leq \frac{1+\sqrt{5}}{2}\tilde{r}_+ < 1.62\tilde{r}_+ \text{ if } p_{d-1} = 0, \tag{15}$$

$$0.618\tilde{r}_- < \frac{2}{1+\sqrt{5}}\tilde{r}_- \le r_d \le \sqrt{\frac{d}{2}}\tilde{r}_- \text{ if } p_1 = 0, \tag{16}$$

$$r_1 \le 1 + \sum_{i=0}^{d-1} \left|\frac{p_i}{p_d}\right|, \frac{1}{r_d} \le 1 + \sum_{i=1}^{d} \left|\frac{p_i}{p_0}\right|. \tag{17}$$

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

$$r_1^2 \le \frac{M(p)^2}{|p_d|} \le \max_{i=0}^{d-1} \left|\frac{p_i}{p_d}\right|^2, \frac{1}{r_d^2} \le \frac{M(p_{\mathrm{rev}})^2}{|p_0|^2} \le \max_{i=1}^{d} \left|\frac{p_i}{p_0}\right|^2 \tag{18}$$

It is shown in [6] that we can get a very fast approximation of all root radii of $p$ at the origin at a very low cost, which complements the estimates 13, 14 15, 16, 17, and 18.

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. 2 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. [7],[4],[11],[3], and [2]) 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 $r_d \le \rho_{i,-} := \left(\binom{d}{i}\left|\frac{p_0}{p_i}\right|\right)^{\frac{1}{i}}, \frac{1}{r_1} \le \frac{1}{\rho_{i,+}} := \left(\binom{d}{i}\left|\frac{p_d}{p_{d-i}}\right|\right)^{\frac{1}{i}}$ and therefore, since $p^{(i)}(0) = i!p_i$ for all $i > 0$, that

$$r_d \le \rho_{i,-} = \left(i!\binom{d}{i}\left|\frac{p(0)}{p^{(i)}(0)}\right|\right)^{\frac{1}{i}}, \frac{1}{r_1} \le \frac{1}{\rho_{i,+}} = \left(i!\binom{d}{i}\left|\frac{p_{\mathrm{rev}}(0)}{p_{\mathrm{rev}}^{(i)}(0)}\right|\right)^{\frac{1}{i}} \tag{19}$$

for all $i$; from which we obtain relations 9 for $i = 1$.

### 1.5 Higher order derivatives of $\frac{p'(0)}{p(0)}$

Recall that the ratios $\frac{p'(0)}{p(0)}$ and $\frac{xp'(0)}{p(0)}$ play a key role in the root-finders and root-radii computations. We evaluate the ratio $p'(x)/p(x) = p_0'(x)/p_0(x)$ by applying the recurrence

$$\frac{p_{i+1}'(x)}{p_{i+1}(x)} = \frac{1}{2\sqrt{x}}\left(\frac{p_i'(\sqrt{x})}{p_i(\sqrt{x})} - \frac{p_i'(-\sqrt{x})}{p_i(-\sqrt{x})}\right), \ i = 0, 1, \ldots \tag{20}$$

Recurrences (3) and (20) reduce the evaluation of $p_\ell(c)$ to the evaluation of $p(c)$ at $q = 2^\ell$ points $c^{1/q}$ and for $c \ne 0$ reduce the 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 (20) to support fast convergence to the convex hull of the roots.

Equations (5) and (6) enable us to strengthen upper estimates in Eq. 9 and more generally Eq. 19 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. 9); we can approximate the higher order derivatives $\left(\frac{p^{(g)}(x)}{p^{(g-1)}(x)}\right)^{(\ell)}$ at $x = 0$ by using a divided difference (See Alg. 5 of [companion papper]). Besides the listed applications of root-squaring, one can apply DLG iterations to randomized exclusion tests for sparse polynomials. 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 as the additional cost incurred 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 (20), and one can apply recurrence (3) by using a crude approximation to $p_d$ and if needed can scale polynomials $p_i(x)$ for some $i$.

We can treat $\frac{p'(x)}{p(x)}$ as another blackbox that runs the same order of cost since the same blackbox oracle that evaluates $p(x)$ can be used to evaluate $p'(x)$ as well, based on the following theorem:

**Theorem 2.** *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.* [8] and [1] prove the theorem for any function $f(x_1,\dots,x_s)$ that has partial derivatives in all its $s$ variables $x_1,\dots,x_s$.

## 2 Our Algorithm for Evaluating $\left(\frac{p'(x)}{p(x)}\right)^{(\ell)}$

In this section we present the design of the general algorithm for approximating the root radius given by Eq. 5. 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. 3. 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 $r, \theta$ where $\theta \approx \frac{p}{q} = \frac{p}{2^\epsilon}$, *i.e.,* perform Alg. 4, 2) 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) perform the second pass of the going-down algorithm where we compute the values

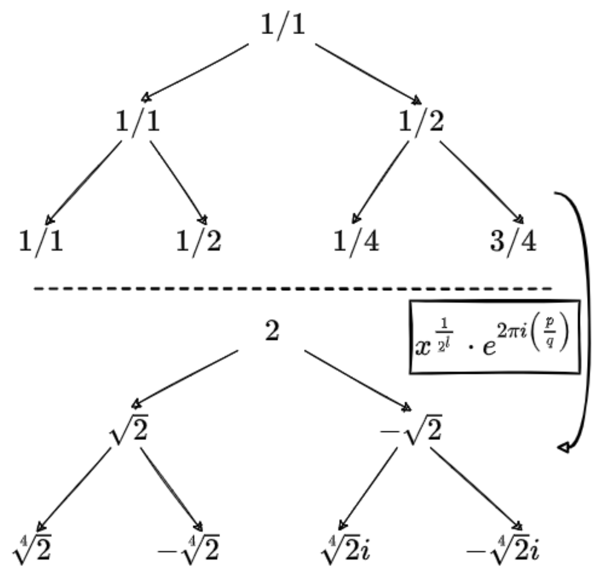


**Fig. 1.** The upper tree depicts the steps of CIRCLE_ROOTS_RATIONAL_FORM$(p, q, l)$ in Alg.1 for $l = 2$, $p = 1$, and $q = 1$. The lower tree depicts the steps of ROOTS$(r, t, u, l)$ in Alg.2 for $r = 2$, $l = 2$, $p = 1$, and $q = 1$

**Fig. 2.**

$|x|^{\frac{1}{2^m}} \exp(2\pi i\frac{p}{q})$ at the $m^{\text{th}}$ level, and 4) finally compute the values given by Eq. 20 going back up the rational root tree.

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

$$p\%q \neq 0 \implies \sqrt{\exp\left(2\pi i\frac{p}{q}\right)} = \exp\left(2\pi i\frac{p}{2q}\right) \tag{21}$$

and

$$p\%q = 0 \implies \sqrt{\exp\left(2\pi i\frac{1}{1}\right)} = \exp\left(2\pi i\frac{1}{1}\right) = 1, \tag{22}$$

and the negation operation satisfies

$$p\%q \neq 0 \implies -\exp\left(2\pi i\frac{r}{s}\right) = \exp\left(2\pi i\frac{2r+s}{2s}\right) \tag{23}$$

and

$$p\%q = 0 \implies \sqrt{\exp\left(2\pi i\frac{1}{1}\right)} = \exp\left(2\pi i\frac{1}{2}\right) = -1; \tag{24}$$

therefore, the first four lines of Alg. 1 compute 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:

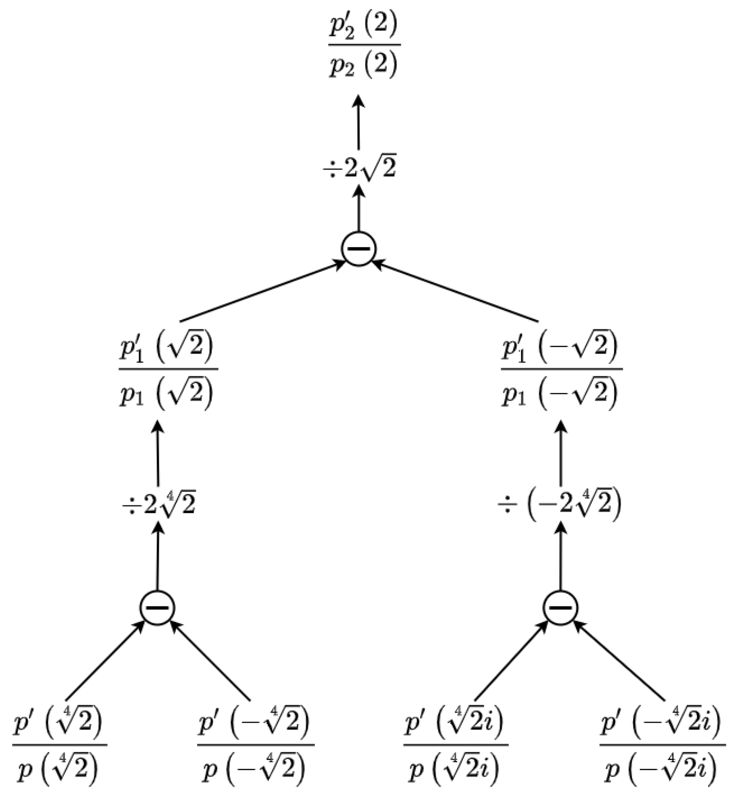


**Fig. 3.** The steps of DLG_RATIONAL_FORM$(p, p', r, t, u, l)$ in Alg.3 for $r = 2$, $l = 2$, $t = 1$, and $u = 1$.

**Fig. 4.**

**Theorem 3.** *For a complex number $x$ with a rational angle $\frac{p}{q}$*, i.e., $x = |x| \exp\left(2\pi i \frac{p}{q}\right)$, *Alg. 1 correctly computes the roots in Eq. 20. In particular, for a rational angle in fractional form, it does so with exact precision.*

*Proof.* Equations 21, 22, 23, and 24 give the base case and the theorem follows by a straightforward induction.

Alg. 2 and Alg. 4 are both straightforward, so for the rest of this section we focus on the intuition behind Alg. 3.

Alg. 3 is a dynamic programming on Equation 20. Therefore, by Thm. 3, we have that Alg. 3 correctly computes Eq. 20. Specifically, Alg. 3 does the following: 1) it computes the last layer of the recursion Eq. 20 (*i.e.,* it computes $p'(x^{\frac{1}{2^l}})/p(x^{\frac{1}{2^l}}))$ and then 2) it recursively applies Eq. 20 via dynamic programming until it finally computes $\frac{p'_\ell(x)}{p_\ell(x)}$ which is the desired quantity. The

**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
```

**Algorithm 2** ROOTS$(r, t, u, l)$

root_tree = CIRCLE_ROOTS_RATIONAL_FORM$(p, q, l)$
circ_root = [exp $\left(2 \cdot \pi \cdot i \cdot \frac{r}{s}\right)$ for $r, s$ in root_tree]
roots =[ $\sqrt[2^l]{r}$·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** $r_i \in$ root **do**
  base_step[$i$] := $\frac{p'(r_i)}{p(r_i)}$
**end for**
diff[0] := base_step
**for** $i \leq l$ **do**
  **for** $j \leq 2^{l-i-1}$ **do**
    diff[$i+1$][$j$]:= $\frac{1}{2}\,\frac{\text{diff}[i][2j]-\text{diff}[i][2j+1]}{\text{root}[2j]}$
    root = roots$(r, t, u, l-1-i)$
  **end for**
**end for**
**return** diff[$l$][0]

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.*, $p'(x^{\frac{1}{2^l}})/p(x^{\frac{1}{2^l}})$) is given by the line "diff[$l$][0]".

**Algorithm 4** DLG$(p, p', l, x, \epsilon)$

angle := $\frac{1}{2\pi i}\log(x)$
$u := 2^{\epsilon}$
$t :=$ (angle $\cdot\, u$)%1
$r := |x|$
**return** DLG_Rational_Form$(p, p', r, t, u, l)$

The final step in the algorithm is to use Alg. 4 to compute root radius approximations $r_d$ and $r_1$. The procedure is given by Alg. 5. The rationale for

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

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

generating a random $x$ is that there 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

$$\lim_{\delta,\epsilon\to\infty} \text{DLG}(p, p', l, x\cdot 2^{-\delta}, \epsilon) = \frac{p'_\ell(0)}{p_\ell(0)} = \left(\frac{p'_{\ell-1}(x)}{p_{\ell-1}(x)}\right)'_{x=0} = \left(\frac{p'(x)}{p(x)}\right)^{(\ell)}_{x=0}, \quad (25)$$

for any $x$, if $p(0) \neq 0$ (See Thm 4).

## 3 Theoretical Analysis

We now give some theoretical guarantees: Thm. 4 proves the correctness of Alg. 5, and Thm. 5 and Thm. 6 give computational complexity bounds.

### 3.1 Correctness of the output

**Theorem 4.** *If $p(0) \neq 0$, then Alg. 5 computes the bounds given by Eq. 9 with probability 1.*

*Proof.* By Lem. 1 we have that the limit in Eq. 1 is well defined. Since there are at most finitely many roots for $p_\ell$ with a high probability Alg. 5 computes the correct approximation to the bounds in Eq. 9.

### 3.2 Complexity

**Theorem 5.** *Alg. 3 performs $q$ floating point subtractions, divisions, and multiplications and $q$ applications of* sin *and* cos*, where $q = 2^l$; furthermore, Alg. 3 performs at most $Cq$ integer additions, "multiplications-by-2", and $\%2^\epsilon$ (*i.e., *mod $2^\epsilon$) operations, where $C = 1, 3, 2$ respectively.*

*Proof.* Looking at Fig. 1 and Fig. 3, we can see that the computational tree for the Alg. 3 is a binary tree with $2^l = q$ nodes; the proof for the constants $C = 1, 3, 2$ follows similarly follows from the inspection of the operations performed in Alg. 1.

**Theorem 6.** *The $Cq$ integer additions, "multiplications-by-2", and $\%2^\epsilon$ (*i.e., *mod $2^\epsilon$) operations in Alg. 3 have negligible overhead. More precisely, integer additions are always additions of $2\,\epsilon \log \ell$-bit integers and "multiplications-by-2" and $\%2^\epsilon$ (*i.e., *mod $2^\epsilon$) 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 $\cdot$ operations are in fact "multiplications-by-2", and $\%2^\epsilon$ (*i.e.,* mod $2^\epsilon$) operations by a straightforward proof similar to the one in Alg. 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; therefore, it suffices to prove that $\log r$ is bounded by $\epsilon \log \ell$. However, this once again follows by induction on the binary computation tree: since this tree has depth $\ell$ we see that any denominator is bounded by $\epsilon \log \ell$ by a simple induction.

In order to prove Thm. 4 we must first prove Lem. 1.

**Lemma 1.** *If $p(0) \neq 0$, then the limit $\lim_{x\to 0} \frac{p'_\ell(x)}{p_\ell(x)}$ is always well-defined.*

*Proof.* By applying induction on Eq. 3, we have that $p_l(0) \neq 0$ if $p(0) \neq 0$, and thus it suffices to consider the behavior of the numerator in Eq. 20. The case $\ell = 1$ follows from an application of L'Hopital's rule. For $\ell \neq 1$, there are two cases: either $\sqrt{x}$ divides the numerator of $\frac{p'_\ell(\sqrt{x}\,)}{p_\ell(\sqrt{x}\,)}$ or it does not. If it does, then we are done since we can once again apply L'Hopital's rule. Otherwise, we get that $p'_\ell(\sqrt{x}\,) = c_0 + c_1\sqrt{x}\, + ... + c_k(\sqrt{x}\,)^k$ for some $c_i$ with $c_0 \neq 0$. But then we have

$$\begin{aligned}
\frac{p'_{\ell+1}(\sqrt{x}\,)}{p_{\ell+1}(\sqrt{x}\,)} &= \frac{1}{2\sqrt{x}}\Big(\frac{p'_\ell(\sqrt{x}\,)}{p_\ell(\sqrt{x}\,)} - \frac{p'_\ell(-\sqrt{x}\,)}{p_\ell(-\sqrt{x}\,)}\Big) \\
&= \frac{1}{2\sqrt{x}} \frac{b_0c_0 + \sqrt{x}\cdot N_1(\sqrt{x}) - b_0c_0 - \sqrt{x}\cdot N_2(\sqrt{x})}{p'_\ell(\sqrt{x}\,)p'_\ell(-\sqrt{x}\,)} \\
&= \frac{1}{2}\frac{N_1(\sqrt{x}) - N_2(\sqrt{x})}{p'_\ell(\sqrt{x}\,)p'_\ell(-\sqrt{x}\,)},
\end{aligned}$$

for some polynomials $N_1$ and $N_2$ with $b_0 = p'_\ell(0)$. Therefore the limit at zero is once again well-defined.

### 3.3 Stability

**Lemma 2.** *The (relative) condition number operator $\kappa$ satisfies the following properties:*

1. $\kappa\{f\}(x) = |x \log'(f(x))|$
2. $\kappa\left\{\frac{f}{g}\right\}(x) = ||\kappa\{f\}(x)| - |\kappa\{g\}(x)||$
3. $\kappa\{x^d\}(x) = d$

*Proof.* This follows from the definition of the condition number.

**Theorem 7.** *If $p_\ell(0) \neq 0$, then $\frac{p_\ell'(x)}{p_\ell(x)}$ is well-conditioned at any point sufficiently close to 0.*

*Proof.* The proof of Lem. 1 gives us that $\frac{p_\ell'(x)}{p_\ell(x)}$ is a well behaved rational function at any point close to zero and thus the condition number $\kappa\{\frac{p_\ell'}{p_\ell}\}(x)$ is well defined at this point since the condition number of arithmetic operations of functions are themselves arithmetic operations in those same functions and the conditions number for a polynomial of degree $d \in \mathbb{R}$ is exactly $d$; therefore, $\kappa\{\frac{p_\ell'}{p_\ell}\}(x)$ has a well defined/bounded condition number in the limit to zero.

*Remark 2.* Even though Thm. 7 states that $\frac{p_\ell'(x)}{p_\ell(x)}$ is highly stable, in practice, computating this function requires the use of trigonometric functions (*i.e.,* the roots in Alg. 2); therefore, our added precision in Alg.. 1 helps with the instability associated with Alg. 2. Intuitively this helps the instability because the trigonometric functions are the only subroutines in the algorithm that have a non constant condition number and thus, special care must be taken with them.

## 4 Experimental Results

### 4.1 Setup

### 4.2 Tables

### 4.3 Observations

## 5 Conclusion

## References

1. Baur, W., Strassen, V.: The complexity of partial derivatives. Theoretical computer science **22**(3), 317–330 (1983)
2. Bini, D.A., Robol, L.: Solving secular and polynomial equations: A multiprecision algorithm. Journal of Computational and Applied Mathematics **272**, 276–292 (2014)
3. Bini, D.A., Fiorentino, G.: Design, analysis, and implementation of a multiprecision polynomial rootfinder. Numerical Algorithms **23**(2), 127–173 (2000)

4. Carstensen, C.: Inclusion of the roots of a polynomial based on gerschgorin's theorem. Numerische Mathematik **59**(1), 349–360 (1991)
5. Householder, A.S.: Dandelin, lobacevskii, or graeffe. The American Mathematical Monthly **66**(6), 464–466 (1959), http://www.jstor.org/stable/2310626
6. Imbach, R., Pan, V.Y.: Root radii and subdivision for polynomial root-finding. CoRR **abs/2102.10821** (2021)
7. Kerimov, M.K.: 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), 1330–1331 (1977)
8. Linnainmaa, S.: Taylor expansion of the accumulated rounding error. BIT Numerical Mathematics 1976 16:2 **16**, 146–160 (1976). https://doi.org/10.1007/BF01931367, https://link.springer.com/article/10.1007/BF01931367
9. Malajovich, G., Zubelli, J.P.: On the geometry of graeffe iteration. J. Complex. **17**, 541–573 (2001)
10. Mignotte, M., Stefanescu, D.: Polynomials: An algorithmic approach. Springer (1999)
11. Pan, V.Y.: Approximating complex polynomial zeros: modified weyl's quadtree construction and improved newton's iteration. journal of complexity **16**(1), 213–264 (2000)
12. Yap, C.K., et al.: Fundamental problems of algorithmic algebra, vol. 49. Oxford University Press Oxford (2000)