Fast evaluation and root finding for polynomials with floating-point coefficients

Rémi Imbach
Université de Lorraine, CNRS, Inria, LORIA
F-54000 Nancy, France

Guillaume Moroz
guillaume.moroz@inria.fr
Université de Lorraine, CNRS, Inria, LORIA
F-54000 Nancy, France

ABSTRACT

Evaluating or finding the roots of a polynomial $f(z) = f_0 + \cdots + f_d z^d$ with floating-point number coefficients is a ubiquitous problem. By using a piecewise approximation of f obtained with a careful use of the Newton polygon of f, we improve state-of-the-art upper bounds on the number of operations to evaluate and find the roots of a polynomial. In particular, if the coefficients of f are given with m significant bits, we provide for the first time an algorithm that finds all the roots of f with a relative condition number lower than 2^m, using a number of bit operations quasi-linear in the bit-size of the floating-point representation of f. Notably, our new approach handles efficiently polynomials with coefficients ranging from 2^{-d} to 2^d, both in theory and in practice.

Evaluating or finding the roots of a polynomial with coefficients represented as floating-point numbers is widely used. Given two positive integer bounds m and τ, a floating number can be represented as $a2^b$ where a and b are integers with $|a| \leq 2^m$ and $|b| \leq \tau$. This representation notably allows to use a constant size $m + \log_2 \tau$ to represent numbers with different orders of magnitude [6, 14, 22].

In the literature, analysing the complexity for approximating the roots of a polynomial is usually done by considering a fixed-point representation for the coefficients ([2, 20, 21, 25] and references therein). In those analyses, the coefficients are either integers or represented as fixed-point numbers with a uniform error on all the coefficients. The drawback of those approaches is that finding the root of the polynomial $2^\tau - 2^{-\tau} z = 0$ will require $\widetilde{O}(\tau)$ bits operations where $\widetilde{O}(\cdot)$ means that we omit the logarithmic factors. On the other hand, this simple equation can be solved in $\widetilde{O}(\log \tau)$ bit operations if we use floating-point arithmetic.

Polynomial where the coefficients have different orders of magnitude appear in several applications. For example, truncating a series expansion of a generalized hypergeometric function such as e^z to order d may return a polynomial where the k-th coefficient has a magnitude in $\Theta(1/k!)$. The characteristic polynomial of a matrix also has coefficients with different order of magnitude. Solving multivariate polynomial systems can be done by eliminating variables and reducing the problem to a univariate polynomial having coefficients with different orders of magnitude [1].

Some work in the literature address the problem of handling large orders of magnitudes for the root-finding problem, notably when using methods based on the Newton diagram and Graeffe iterations ([8, 18, 23, 24, 27] among others). In these approaches, intermediate values can have large order of magnitudes and a process of normalisation can be applied to reduce their sizes [11, 13, 19]. However, the algorithm they describe is quadratic in the degree of the input polynomial.

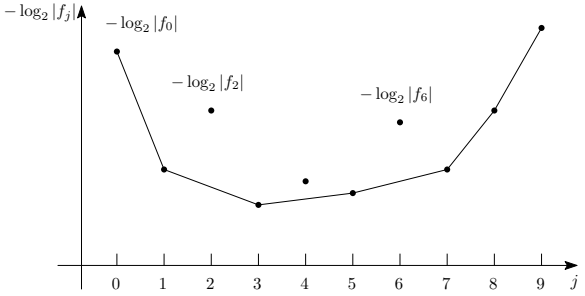

Figure 1: The Newton polygon of the polynomial f, using the valuation $-\log_2|f_j|$ for the j-th coefficient.

Using the Newton polygon of f we develop a new method adapted to the working relative precision of the input coefficients. Let f be a polynomial of degree d with floating-point coefficients of magnitude less than 2^τ and represented with m significant bits. We will compute a piecewise approximation of f by $O(d/m)$ polynomials g of degree $O(m)$, where the coefficients of each polynomial g is represented with $O(m)$ significant digits. This representation will allow us to improve state-of-the-art bounds on the problems of evaluating f on d points and of finding its roots.

After stating formally our main results in Section 1, we will describe in Sections 2 and 3 how to compute such a piecewise approximation in $\widetilde{O}(d(m + \log \tau))$ bit operations. Then we will show how to use this data-structure to evaluate f (Section 4) and to approximate or isolate the roots of f (Section 5). Finally we describe experimental results of our prototype implementations on different families of polynomials (Section 6).

1 PRELIMINARIES

In the article, our input is the polynomial $f(z) = f_0 + \cdots + f_d z^d$. We assume that the coefficients of f are represented with floating-point numbers and our goal is to design fast algorithms for the evaluation or the root-finding problem, with the same error bounds as if we had used classical algorithms with floating-point arithmetic.

For that we will first compute in Sections 2 and 3 a piecewise polynomial approximation of f defined on a partition of the complex plane. Before stating our main result, we recall two mathematical tools that will be fundamental in our algorithms: the *condition number* and the *Newton polygon*.

Notations. We will use the following notations to describe our piecewise polynomial approximation. For a complex number γ and a positive real ρ, we denote by $D(\gamma, \rho)$ the disk of center γ and radius ρ. For two positive real number r_1, r_2, we denote by $R(r_1, r_2)$

the ring $D(0, r_2) \setminus D(0, r1)$. Moreover, for a polynomial f we let:

$$\widetilde{f}(|z|) = |f_0| + \cdots + |f_d||z|^d, \qquad \|f\|_1 = |f_0| + \cdots + |f_d|$$
$$\hat{f}(|z|) = \max_{0 \le j \le d}(|f_j||z|^j), \qquad f_{\ell,u}(z) = f_l + \cdots + f_u z^{u-\ell}.$$

1.1 Relative condition number

In floating-point representation, the errors are relative. Let f_ε be the polynomial obtained by adding a relative error bounded by ε on the coefficients of f. This amounts to multiply all the coefficients f_j by $(1 + \varepsilon_j)$ with $|\varepsilon_j| \le \varepsilon$.

In the *evaluation problem*, for any complex point z we have

$$|f(z) - f_\varepsilon(z)| \le \varepsilon \widetilde{f}(|z|)$$

and this bound is tight. In Theorem 1.1 we focus on the problem of finding the approximate value of $f(z)$ with the same error bound.

For the *root-finding problem*, we can also define the relative condition number [10],[14, §1.6] associated to a relative perturbation of the coefficients of f. More precisely, if ζ is a root of f, and $\dot{\zeta}$ is the closest root of f_ε, the relative condition number of ζ denoted by $\mathrm{cond}(f, \zeta)$ is defined as $\lim_{\varepsilon \to 0} |\zeta - \dot{\zeta}|/(|\zeta|\varepsilon)$. Using the Taylor expansion $f_\varepsilon(\zeta) = (\zeta - \dot{\zeta}) f'_\varepsilon(\zeta) + O(|\zeta - \dot{\zeta}|^2)$ and $|f(\zeta) - f_\varepsilon(\zeta)| \le \varepsilon \widetilde{f}(|\zeta|)$, this leads to:

$$\mathrm{cond}(f, \zeta) = \frac{\widetilde{f}(|\zeta|)}{|\zeta||f'(\zeta)|}.$$

This means that the number of bits of precision required to compute the first significant bit of ζ is in $O(\log(\mathrm{cond}(f, \zeta))$. In Section 5, we focus on the problem of finding the roots of f with a number of bit operations quasi-linear in $\log(\mathrm{cond}(f, \zeta))$ and quasi-linear in d.

1.2 Newton polygon

Given a non-zero complex coefficient f_j of the polynomial f, we can associate it with the value $-\log_2 |f_j|$. Then for each index $0 \le j \le d$, let P_j be the point $(j, -\log_2 |f_j|)$. The *Newton polygon* of f is the lower convex hull of the set of points P_j, as illustrated on Figure 1. It has been used to give a rough first estimation of the modules of the roots of a polynomial [3, 4, 8, 15, 18, 23, 24, 27].

We will denote by H the Newton polygon of f, and use it to partition the complex plane in rings where the variation of $\widetilde{f}$ is bounded (Section 2).

1.3 Main results

We can now state our main results, with some reasonable assumptions on the precision such as $m > \log d$ for a simpler presentation.

The first theorem shows that we can evaluate a polynomial of degree d on d points with a complexity quasi-linear in d times the size of a floating-point representation of the coefficients. In particular, if the magnitudes of the coefficients are bounded by 2^τ, this means that our algorithm will be quasi-linear in $d \log \tau$. This is an improvement over a recent result [21] where the complexity was quasi-linear in $d\tau$.

Theorem 1.1 (Evaluation). *Let f be a polynomial of degree d with complex floating-point coefficients of magnitudes less than 2^τ and let m be a positive integer. It is possible to compute in $\widetilde{O}(d(m + \log \tau))$ bit operations a piecewise polynomial approximation f_{pw} such that*

$$|f(z) - f_{pw}(z)| \le 2^{-m} \widetilde{f}(z)$$

for all $z \in \mathbb{C}$. Moreover, assuming that $m > \log d$, and given d complex points $z_1, \ldots, z_d$ with $|z_j| \le 2^\tau$, it is possible to evaluate:

(i) *$f_{pw}(z_1)$ in $\widetilde{O}(m(m + \log \tau))$ bit operations,*
(ii) *$f_{pw}(z_1), \ldots, f_{pw}(z_d)$ in $\widetilde{O}(d(m + \log \tau))$ bit operations.*

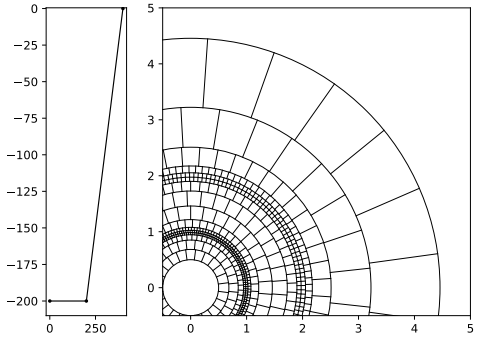

Figure 2: Newton polygon (left) and sector partition (right) associated to the polynomial $(z^{200} - 1)(z^{200} - 2^{200})$

The second theorem allows us to find the roots of polynomials with a number of bit operations quasi-linear in $\log \tau$, and to handle polynomials with large coefficients such as resultant polynomials of two bivariate polynomials (see Section 6.2).

Theorem 1.2 (Root finding). *Let f be a polynomial of degree d with complex floating-point coefficients of magnitudes less than 2^τ and let m be a positive integer. It is possible to compute the m most significant bits of the roots ζ of f such that $\mathrm{cond}(f, \zeta) \le 2^m$ in $\widetilde{O}(d(m + \log \tau))$ bit operations.*

Moreover, if $\kappa = \max_{\zeta | f(\zeta) = 0} \mathrm{cond}(f, \zeta)$, it is possible to compute the m most significant bits of all the roots of f in $\widetilde{O}(d(m + \log \tau + \log \kappa)$ bit operations.

Our approach consists in computing a piecewise approximation in two steps. A first approximation is computed on a partition of the complex plane in concentric rings, and is described in Section 2. Then we refine our piecewise approximation by partitioning each ring uniformly in angular sectors, as described in Section 3.

2 PIECEWISE APPROXIMATION OVER RINGS

Our first piecewise approximation is based on a subdivision of the complex plane with concentric rings R_n centered at the origin, as illustrated in Figure 2. In Algorithm 1, we compute the rings and we select dominant monomials of f. This returns a list of rings $R_n = R(r_n, r_{n+1})$ and a pair of indices (ℓ_n, u_n) associated to each ring such that evaluating $z^{\ell_n} f_{\ell_n, u_n}(z)$ on R_n is sufficient to evaluate f up to a given error bound. One of the goal of this algorithm is to choose the rings such that the sum of the number of monomials of the polynomials f_{ℓ_n, u_n} is linear in d. The main result of this section is summarized in the following Lemma.

Lemma 2.1. *Given a polynomial f of degree d with floating-point coefficients with m-bits mantissa and magnitude less than 2^τ, it is*

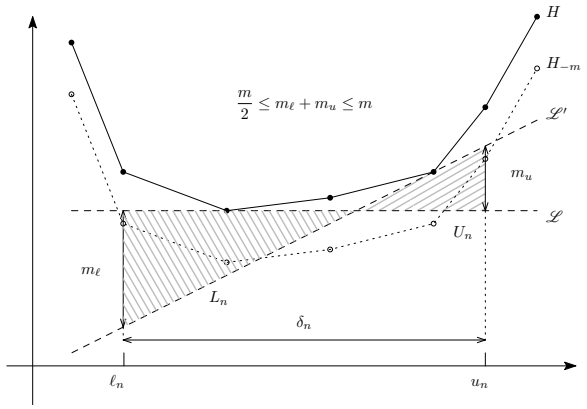

Figure 3: One step to sweep the line in Algorithm 1.

possible to compute in $\widetilde{O}(d(m+\log\tau))$ *bit operations* N *rings* $R_n = R(r_n, r_{n+1})$ *and extract* N *polynomials* f_{ℓ_n,u_n} *of degrees* δ_n *such that:*

(i) $$2^{\frac{m}{2}} r_n^{\delta_n} \le r_{n+1}^{\delta_n} \le 2^m r_n^{\delta_n} \quad \text{if } \delta_n \ge 1, \text{ and}$$

(ii) $$\sum_{n=0}^{N-1} (\delta_n + 1) \le 65d + 1.$$

And for all $z \in R_n$:

(iii) $$|f(z) - z^{\ell_n} f_{\ell_n,u_n}(z)| \le (d-\delta_n) 2^{-m} \hat{f}(|z|).$$

We use the Newton polygon associated to f to compute such approximation polynomials, which allows us to reduce computing rings and indices that satisfy Lemma 2.1 to geometrical arguments.

In Section 2.1 we start by detailing Algorithm 1 and prove the inequalities (i) and (iii) of Lemma 2.1. Then in Section 2.2 we prove the inequality (ii) that is fundamental to bound the complexity of all the subsequent algorithms.

2.1 Algorithm

Our algorithm takes as input the Newton polygon H of the polynomial f. The Newton polygon is the lower convex hull of the points $(j, -\log_2|f_j|)$ for $0 \le j \le d$ and in the general case, it can be computed in $O(d \log d)$ arithmetic operations [9], or $O(d)$ arithmetic operations when the points are sorted, which is the case here.

The main idea of Algorithm 1 is to swipe a line tangent to H, as illustrated in Figure 3. Given a line $\mathscr{L}$ of slope s tangent to H, the next line $\mathscr{L}'$ of slope s' tangent to H, with $s' > s$, is computed informally as follow:

1. Let H_{-m} be the polygon H shifted vertically by $-m$.
2. Let ℓ be the abscissa of the leftmost point of H_{-m} below $\mathscr{L}$.
3. For the new line $\mathscr{L}'$, we denote by
   - u the abscissa of the rightmost point of H_{-m} below $\mathscr{L}'$,
   - m_ℓ the vertical distance between $\mathscr{L}$ and $\mathscr{L}'$ at ℓ,
   - m_u the vertical distance between $\mathscr{L}$ and $\mathscr{L}'$ at u.
4. Starting from the slope of $\mathscr{L}$, increase the slope of $\mathscr{L}'$ until
$$m/2 \le m_\ell + m_u \le m.$$

In Algorithm 1, line 10 addresses step 1 above. In line 11 we compute the maximal abscissa u such that $m_\ell + m_u \le m$. Then we choose $s' = s + m/(u-l+1)$ for the new slope of $\mathscr{L}'$; this implies that $m_l + m_u = m(u-l)/(u-l+1) \le m$, and as soon as $u > l$, we have $m/2 \le m_l + m_u \le m$. Thus, if $\delta_n \ge 1$, we have $(r_{n+1}/r_n)^{\delta_n} = 2^{m(u_n-\ell_n)/(u_n-\ell_n+1)}$ is between $2^{m/2}$ and 2^m, which proves the inequality (i) of Lemma 2.1.

For the inequality (iii), let z be such that $r_n \le |z| \le r_{n+1}$. This means that $\log|z|$ is between s_n and s_{n+1}. Let $\mathscr{L}''$ be the line of slope $\log|z|$ tangent to H. It is below $\mathscr{L}$ for the abscissae less than ℓ_n, and below $\mathscr{L}'$ for the abscissae greater than u_n. In particular, let $P = (k, y_k)$ be a vertex of H contained in $\mathscr{L}''$, and let $Q = (j, y_j)$ be a point of H_{-m} such that $j < \ell_n$ or $j > u_n$ and $|f_j| \ne 0$. By construction, we have $y_k = -\log_2|f_k|$ and $y_j \le -\log_2|f_j| - m$. Moreover we know that the line $\mathscr{L}''$ is below P_j. Since $\mathscr{L}''$ is a line of slope $\log|z|$ passing through P_k, this implies that $y_k + (j-k)\log_2|z| \le y_j$. Equivalently, this means that $-\log_2|f_k| + (j-k)\log_2|z| \le -\log_2|f_j| - m$. Taking the power of two, this implies that $|z|^{j-k}/|f_k| \le 2^{-m}/|f_j|$, which can be rewritten as $|f_j||z|^j \le 2^{-m}|f_k||z|^k \le 2^{-m}\hat{f}(|z|)$. This implies that for all $z \in R_n$ and all $j < \ell_n$ and all $j > u_n$, we have $|f_j||z|^j \le 2^{-m}\hat{f}(|z|)$, which in turn implies the inequality (iii).

Algorithm 1: Rings and dominant monomials

Input: m: a positive integer
H: list of points (j, h_j) on the Newton polygon of f for $0 \le j \le d$
Output: Satisfying inequalities (i), (ii), (iii) of Lemma 2.1
$(r_n)_{0\le n<N}$: list of circle radii surrounding the rings
$(\ell_n)_{0\le n<N}$: lower indices of the dominant monomials
$(u_n)_{0\le n<N}$: upper indices of the dominant monomials

1 **for** *j from* 1 *to* $d-1$ **do**
2 $\mathcal{L} \leftarrow$ lines tangent to H passing through $(j, h_j - m)$
3 $S_j^0 \leftarrow$lower slope of the 2 lines in $\mathcal{L}$
4 $S_j^1 \leftarrow$upper slope of the 2 lines in $\mathcal{L}$
5 start $\leftarrow$ slope of line through $(0, h_0)$ and $(1, h_1 - m)$
6 end $\leftarrow$ slope of line through $(d-1, h_{d-1})$ and $(d, h_d - m)$
7 $n, s_n \leftarrow 1$, start
8 $r_0, r_1, \ell_0, u_0 \leftarrow 0, 2^{\text{start}}, 0, 0$
9 **while** $s_n <$ *end* **do**
10 $\ell_n \leftarrow$ minimal index ℓ such that $S_\ell^1 > s_n$
11 $u_n \leftarrow$ maximal index u such that $(u-\ell_n)(S_u^0 - s) < m$
12 $s_{n+1} \leftarrow s_n + \frac{m}{u_n - \ell_n + 1}$ **if** $u_n > \ell_n$ **else** $S_{u_n+1}^0$
13 $r_{n+1} \leftarrow 2^{s_{n+1}}$
14 $n \leftarrow n+1$
15 $r_{n+1}, \ell_n, u_n \leftarrow +\infty, d, d$
16 $N \leftarrow n+1$
17 **return** $(r_n)_{0\le n\le N}, (\ell_n)_{0\le n<N}, (u_n)_{0\le n<N}$

2.2 Complexity

We can now prove the inequality (ii) that will allow us notably to bound the number of bit operations of Algorithm 1 in Section 2.2.2.

2.2.1 Bound on cumulated number of monomials. By the way the δ_n are computed, it is not obvious that their sum is linear in d. The proof comes from a geometrical argument. For each step, we can

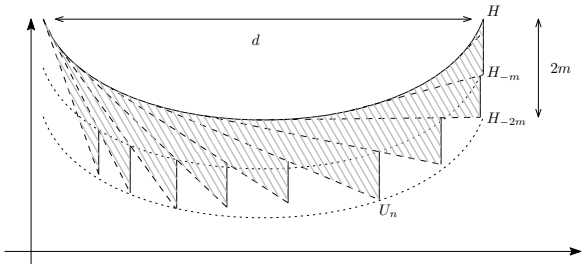

Figure 4: N steps to sweep the line in Algorithm 1.

define an area that has a size in $\Omega(\delta_n m)$, and we will show that the sum of the area is in $O(dm)$, as illustrated on Figure 4. More precisely, for $0 \leq n < N$ such that $\delta_n \geq 1$, let V_n be the vertical band between the abscissae ℓ_n and u_n. Letting $\mathscr{L}$ and $\mathscr{L}'$ be the lines of slopes s_n and s_{n+1} respectively, we define two regions:

- U_n the set of points of V_n above $\mathscr{L}$ and below $\mathscr{L}'$
- L_n the set of points of V_n below $\mathscr{L}$ and above $\mathscr{L}'$

First, since V_n has width δ_n, and the difference of the slopes between $\mathscr{L}$ and $\mathscr{L}'$ is $m/(\delta_n+1) \geq m/(2\delta_n)$, the sum of the areas of L_n and U_n is at least $\delta_n m/8$.

Then, let U be the union of the U_n for $0 \leq n < N$ and $\delta_n \geq 1$. By construction, for different indices n, the U_n are pairwise distinct, such that the area of U is greater than the sum of the area of U_n. Similarly, letting L be the union of the L_n we have that the area of L is greater than sum of the area of L_n.

Finally, let H_{-2m} be the polygon obtained by shifting the Newton polygon H vertically by $-2m$. By construction, the index u_n has been chosen such that the vertical distance between $\mathscr{L}$ and the point of abscissa u_n of H_{-m} is at most m. Thus the points where $\mathscr{L}$ exits V_n on the right is above H_{-2m}. Moreover, at abscissa ℓ_n, the line $\mathscr{L}$ is above H_{-m}, so that in the vertical band V_n, the line $\mathscr{L}$ lies entirely above H_{-2m}. Similarly, we can prove that $\mathscr{L}'$ is above H_{-2m} since it is above $\mathscr{L}$ at abscissa u_n, and at abscissa ℓ_n, its vertical distance to $\mathscr{L}$ is $m\ell_n \leq m$. Thus U_n and L_n are contained in the band between H and H_{-2m}. This implies that U (resp. L) has an area less than $2dm$.

Gathering all the geometrical constraints, and letting $|U|$, $|L|$, $|U_n|$, $|L_n|$ be the areas of the corresponding regions, we have:

$$\sum_{0\leq n<N \mid \delta_n \geq 1} \frac{\delta_n m}{8} \leq \sum_{0\leq n<N \mid \delta_n \geq 1} |U_n| + |L_n| \leq |U| + |L| \leq 4dm$$

such that $\sum_{0\leq n<N} \delta_n \leq 32d$. And for $\delta_n \geq 1$ we have $\delta_n + 1 \leq 2\delta_n$, which implies $\sum_{0\leq n<N, \delta_n \geq 1}(\delta_n+1) \leq 64d$.

To conclude the proof of the inequality (ii) of Lemma 2.1, we need to take also into account the case where $\delta_n = 0$. This case happens when $\ell_n = u_n$. The values ℓ_n such that $\delta_n = 0$ are all distinct, such that $\sum_{0\leq n<N, \delta_n=0}(\delta_n+1) \leq d+1$.

2.2.2 Bound on the complexity of Algorithm 1. We prove here that the number of bit operations of Algorithm 1 is in $\widetilde{O}(d(m+\log\tau))$. Let N be the number of iterations of the **while** loop.

First, the lists S^0 and S^1 can be computed with $O(d)$ arithmetic operations. Second, since the indices ℓ and u are non-decreasing, computing the sequences $(\ell_n)_{n\leq N}$ and $(u_n)_{n\leq N}$ amount to $O(N+d)$ comparisons. Third, the terms of the sequence $(r_n)_{n\leq N}$ can be computed using Arithmetic-Geometric means [5]. This method is quasi-linear in required number of correct significant bits. The h_i have a magnitude bounded by τ. Thus, the variable s has a magnitude in $O(\tau)$ and it is sufficient to perform the arithmetic operations and the exponentiation with $O(m+\log d+\log\tau)$ correct significant digits in order to get the required precision for a term r_n.

Finally, using inequality (ii) of Lemma 2.1, N is in $O(d)$ whichconcludes the proof.

3 PIECEWISE APPROXIMATION OVER ANGULAR SECTORS

In the previous section, we computed a first piecewise approximation of f by polynomials $z^{\ell_n} f_{\ell_n,u_n}(z)$ over a partition of the complex plane in concentric rings R_n. Even though the total number of monomials of those polynomials is linear in d, it is possible that a single approximating polynomial f_{ℓ_n,u_n} has a degree d. Thus we need to further truncate the polynomials so that subsequent evaluations or root finding methods are faster. This is done by dividing uniformly each ring R_n in angular sectors $A_{n,k}$, and by computing a Taylor approximation of f_{ℓ_n,u_n} in each sector. More precisely, let $\gamma_{n,k}$ be the center and $|\rho_{n,k}|$ be the radius of a disk that contains $A_{n,k}$. Then we approximate f_{ℓ_n,u_n} by computing the first $4m+1$ coefficients of the polynomial f_{ℓ_n,u_n} composed with the polynomial $\gamma_{n,k}+\rho_{n,k}t$. We denote by $g_{n,k}$ the resulting polynomial. The main result of this section is summarized in the following Lemma.

Lemma 3.1. *Given a polynomial f_{ℓ_n,u_n} satisfying the conditions of Lemma 2.1, it is possible to compute in $\widetilde{O}(\delta_n(m+\log\tau))$ bit operations the parameters of K affine changes of variable $Z = \gamma_{n,k} + \rho_{n,k}T$ and K polynomials $g_{n,k}$ of degree $\min(\delta_n, 4m)$ such that:*

(i) *R_n is included in the union of the disks $D(\gamma_{n,k}, |\rho_{n,k}|)$,*

(ii) $\|f_{\ell_n,u_n}(\gamma_{n,k}+\rho_{n,k}T) - g_{n,k}(T)\|_1 \leq (\delta_n+1)2^{-m}\hat{f}(r_n)r_n^{-\ell_n}$.

We use Algorithm 2 to compute those approximations. We describe it and provide its complexity analysis in Section 3.1, and we prove the inequalities (i) and (ii) in Section 3.2.

3.1 Algorithm and complexity analysis

In Algorithm 2 we focus on one ring R_n defined by two circles of radii r_n and r_{n+1}. The algorithm is a variant of the algorithm used in a previous work where the rings were given by explicit formula [21, §3]. The main differences are that we need to handle arbitrary rings, and we need to guarantee a different error bound for the approximation polynomials $g_{n,k}$.

We start by computing the parameters of the disks covering the ring R_n, where γ is the middle point between r_n and r_{n+1}, whereas ρ is $3/2$ the radius of the interval $[r_n, r_{n+1}]$. With those parameters, we can check that the disk $D(\gamma,\rho)$ contains an angular sector of R_n of angle $2\rho/\gamma$ (Lemma 3.2). The number of disks necessary to cover R_n is thus bounded by $K \leq 3(\delta/m)+2$, which is in $O(\max(\delta,m)/m)$. Then we want to compute the first $4m+1$ coefficients of the polynomials in t obtained after the change of variable $z = (\gamma+\rho t)e^{i2\pi k/K}$ for all $0 \leq k < K$ in the polynomial f_{ℓ_n,u_n}; next we show how to do it with bit complexity $\widetilde{O}(\delta m)$ using a baby-step giant-step or rectangular splitting approach [6].

For that, remark that when computing the fast Fourier transforms, the $e^{2\pi j/K}$ are equal for indices that are equal modulo K.

Algorithm 2: Piecewise polynomial approximation

Input: m: a positive integer
f: input polynomial
r_n, r_{n+1}: two consecutive radii surrounding a ring
ℓ_n: lower index of the dominant monomials
u_n: upper index of the dominant monomials
Output: Satisfying inequalities (i),(ii) of Lemma 3.1
$(\gamma_{n,k}, \rho_{n,k})_{0\le k<K_n}$: change of variable parameters: $z = \gamma_{n,k} + \rho_n t$
$(g_{n,k})_{0\le k<K_n}$: list of approximation polynomials

```
# Computes truncated Taylor shifts using a
# baby-step giant step approach

# A. Parameters for the change of variable
#      z = (γ + ρt)e^{i2π k/K}
```
1 $\gamma, \rho, \leftarrow \frac{r_{n+1}+r_n}{2}, \frac{3}{2}\frac{r_{n+1}-r_n}{2}$
2 $\delta \leftarrow u_n - \ell_n$
3 $K \leftarrow \lceil 2\pi\frac{\gamma}{\rho}\rceil$ # $K \le 3\delta/m + 2$
4 $m' \leftarrow \lfloor \frac{\delta}{K} \rfloor$

```
# B. Baby steps.
#      Compute (γ + ρT)^k mod T^{4m+1}, normalized
```
5 $q_0(T) \leftarrow \frac{\gamma+\rho T}{\gamma+\rho}$
6 **for** k *from* 0 *to* $K-1$ **do**
7 $\quad q_{k+1}(T) \leftarrow q_k(T)\cdot\frac{\gamma+\rho T}{\gamma+\rho} \mod T^{4m+1}$

```
# C. Giant steps
# C.1 Gather monomials with same indices modulo K
```
8 $M \leftarrow m'\frac{\hat f(\gamma)}{\gamma^{\ell_n}}(1+\frac{\rho}{\gamma})^{\delta}$
9 **for** k *from* 0 *to* $K-1$ **do**
10 $\quad p_k(Z) \leftarrow \frac{1}{M}\sum_{j=0}^{m'} f_{\ell_n+k+jK}(\gamma+\rho)^{k+jK}Z^j$

```
# C.2 Fast composition with absolute error 2^{-4m}
```
11 **for** k *from* 0 *to* $K-1$ **do**
12 $\quad r_k(T) \leftarrow \sum_{j=0}^{4m} r_{k,j}T^j = p_k\,(q_K(T))\cdot q_k(T) \mod T^{4m+1}$

```
# D. Spread approximations to all angular sectors
```
13 **for** j *from* 0 *to* $\min(\delta, 4m)$ **do**
14 $\quad r_j^T(W) \leftarrow \sum_{k=0}^{K-1} r_{k,j}W^k$
15 $\quad g_{0,j},\dots,g_{K-1,j} \leftarrow r^T(e^{i2\pi\frac{k}{K}})$ for $0\le k<K$ with FFT
16 **return** $(\gamma e^{i2\pi\frac{k}{K}}, \rho e^{i2\pi\frac{k}{K}})$ for $0\le k<K$
17 $\quad \sum_{j=0}^{4m} M g_{k,j}T^j$ for $0\le k<K$

Thus for a given k between $0 \le k < K$, we can gather the terms of index k modulo K in the polynomial $r_k = \sum_{j=0}^{\delta/K} h_{k+jK}$ before computing the fast Fourier transform. This will allow us to reorder our computations to compute the $g_{n,k}$ in $\widetilde{O}(\delta m)$ bit operations.

More precisely, in step `B.` (the baby-step) we compute the powers of the polynomials $\gamma + \rho T$ modulo T^{4m+1} up to the exponent K which is in $O(\delta/m+1)$. At each step of the loop, normalizing the linear factor can be done in $\widetilde{O}(m+\log\tau)$ bit operations using floating-point arithmetic since γ and ρ have a magnitude in $O(2^\tau)$. Then each polynomial multiplication can be done in $\widetilde{O}(\min(\delta,m)m)$ bit operations. Thus all the polynomials q_k can be computed in $\widetilde{O}(\min(\delta,m)Km + K(m+\log\tau)))$, or $\widetilde{O}(\delta m + K\log\tau)$ since Km is in $O(\max(\delta,m))$.

Then in step `C.` (the giant-step) we compute the coefficients of the polynomials r_k of $q_k\cdot(p_k\circ q_K) \mod T^{4m+1}$ by using a combination of fast Taylor shift ([27, Theorem 8.4]) and fast composition ([26, Theorem 2.2]). This can be done with an absolute error on the coefficients less than 2^{-4m} in $\widetilde{O}(\min(\deg(p_k)\deg(q_k), m)m)$ bit operations. Since the degree of q_k is K and the degree of p_k is $m' \le \delta/K$, the degree $\deg(p_k)\deg(q_k)$ of $p_k\circ q_k$ is in $O(\delta)$ and all the r_k can be computed in $\widetilde{O}(\min(\delta,m)Km)$ bit operations. Finally, Km is in $O(\max(\delta,m)$, such that step `C.2` can be done in $\widetilde{O}(\delta m)$ operations. Moreover, as in step `B.`, normalizing the coefficients of p_k in step `C.1` costs $\widetilde{O}(\delta(m+\log\tau))$.

Finally in step `D.` we compute the polynomials $g_{n,k}$ using $\min(\delta, 4m)$ fast Fourier transforms with an absolute error less than 2^{-4m} ([27]). This can be done in $\widetilde{O}(\min(\delta,m)Km)$ bit operations, that is in $\widetilde{O}(\delta m)$.

Thus, the total number of bit operations of Algorithm 2 is in $\widetilde{O}(\delta(m+\log\tau))$.

3.2 Correctness

We prove here inequalities (i) and (ii) of Lemma 3.1.

3.2.1 Ring cover. For the inequality (i) of Lemma 3.1, the ring R_n is contained in the union of the K_n disks $D(\gamma_{n,k}, |\rho_{n,k}|)$ if the following Lemma holds.

Lemma 3.2. *The disk $D(\gamma,\rho)$ contains an angular sector of R_n of angle $\frac{\rho}{\gamma}$. Furthermore, $\frac{\gamma}{\rho} \le 1 + 3\frac{\delta}{m}$.*

Proof. The angular sector covered by $D(\gamma,\rho)$ has an angle 2β, where β is the angle between two sides of lengths γ and r_{n+1} in a triangle with sides of lengths γ, r_{n+1} and ρ. We have $\beta^2 \ge 2(1-\cos(\beta))$. Then using arguments from the triangle geometry to bound $\cos(\beta)$, we deduce $2\beta > (2/3)\sqrt{5/2}\rho/\gamma > \rho/\gamma$.

Then for the bound on $\frac{\gamma}{\rho}$, we remark that it is equal to $(2/3)(r_{n+1}+r_n)/(r_{n+1}-r_n) = (2/3)(1+1/(r_{n+1}/r_n-1))$. Moreover, $r_{n+1}/r_n \ge 2^{m/(2\delta)} \ge 1 + (\log(2)/2)m/\delta$. With $2/3 < 1$ and $4/(3\log(2)) < 3$, this allows us to conclude the proof. □

3.2.2 Approximation bound. To prove the bound on the approximation inequality (ii) of Lemma 3.1, we need to bound two errors: the error appearing while truncating at order $4m$ the polynomials $f_{\ell_n,u_n}(\gamma+\rho T)$, and the error appearing on the coefficients while computing with an absolute error in 2^{-4m} in Algorithm 2.

First we bound the error coming from the error on the coefficients. In steps `B` and `C.1`, we normalize the polynomials such that the sum of the absolute value of their coefficients is less than 1. Thus, at the end, the absolute error on each coefficient of $g_{n,k}$ is less than $(\delta+1)2^{-4m}M$. We show that it is bounded by $(\delta+1)2^{-m-1}\hat f(r_n)/r_n^{\ell_n}$ using the following bound on $M = m'(\hat f(\gamma)/\gamma^\ell)(1+\rho/\gamma)^\delta$.

Lemma 3.3. *For all $m \ge 1$:*

$$\|g_{n,k}\|_1 \le M \le (\delta+1)2^{3m-1}\hat f(r_n)/r_n^{\ell_n}$$

Proof. Let j be the index such that $\hat{f}(\gamma) = |f_j|\gamma^j$. We have $|f_j|\gamma^{j-\ell_n} = |f_j|r_n^{j-\ell_n}(\gamma/r_n)^{j-\ell_n} \leq (\hat{f}(r_n)/r_n^{\ell_n})(r_{n+1}/r_n)^{\delta}$. With $(r_{n+1}/r_n)^{\delta} \leq 2^m$, this leads to $M \leq m'(\hat{f}(r_n)/r_n^{\ell_n})2^m(1+\rho/\gamma)^{\delta}$. Then, we can notice that $(1+\rho/\gamma)^{\delta} \leq 2^{(1/\log(2))(\delta\rho/\gamma)}$. Moreover, $\rho/\gamma = (3/2)\tanh((s_{n+1}-s_n)\log(2)/2)$. Thus $\rho/\gamma \leq 3\log(2)(s_{n+1}-s_n)/4 \leq (3\log(2)/4)(m/\delta)$. This leads to $(1+\rho/\gamma)^{\delta} \leq 2^{3m/4}$. This also leads to $K \geq (8\pi/(3\log(2)))(\delta/m)$, such that $m' \leq m$. Finally, this implies that M is bounded by $m2^{7m/4}\hat{f}(r_n)/r_n^{\ell_n}$. Moreover, for all $m \geq 1$, we have $m2^{7m/4} \leq 2^{3m-1}$. □

Then the bound on the error coming from the Taylor expansion is obtained by bounding for a given j the coefficients of the polynomial $|f_j|(\gamma+\rho T)^{j-\ell_n}$ of degree greater than $4m$. The k-th coefficient of this polynomial is $|f_j|\gamma^{j-\ell_n}\binom{j-\ell_n}{k}(\rho/\gamma)^k \leq |f_j|\gamma^{j-\ell_n}((e\delta/k)(\rho/\gamma))^k$. With the bound on ρ/γ above, this becomes less than $|f_j|\gamma^{j-\ell_n}((3e\log(2)/4)(m/k))^k$. As previously, we can bound $|f_j|\gamma^{j-\ell_n}$ by $2^m\hat{f}(r_n)/r_n^{\ell_n}$. The second factor can be bounded for all $k \geq 4m$ by $(3e\log(2)m/16)^k \leq 1/2^k$. Thus the sum of all the coefficients of degree $4m+1$ or more is bounded by $(\hat{2}^m f(r_n)/r_n^{\ell_n})2^{-4m}$. Adding the errors for j from ℓ_n to u_n, we get a bound for the error on the remainder of $(\delta+1)\hat{f}(r_n)/r_n^{\ell_n}2^{-m-1}$.

Summing the two errors lead to the required bound for the inequality (ii) of Lemma 3.1.

4 EVALUATION

As a consequence of Lemma 2.1 and 3.1, the polynomials $g_{n,k}$ returned by Algorithm 2 form a piecewise polynomial approximation over the angular sectors $A_{n,k}$ defined by the intersection between the ring $R(r_n, r_{n+1})$ and the disk $D(\gamma_{n,k}, |\rho_{n,k}|)$. This is formalized in the following corollary.

Corollary 4.1. *Let z be a complex point. Then there exists n and k such that $r_n \leq |z| \leq r_{n+1}$ and $z \in D(\gamma_{n,k}, |\rho_{n,k}|)$. Moreover, letting $t = (z-\gamma_{n,k})/\rho_{n,k}$, the polynomial $g_{n,k}$ satisfies:*

$$|f(z) - z^{\ell_n} g_{n,k}(t)| \leq (d+1)2^{-m}\hat{f}(z)$$

This leads to a straightforward method to evaluate f on one point z with an error in $O(d2^{-m}\widetilde{f}(|z|))$. First find the angular sector $A_{n,k}$ it belongs to, then compute $t = (z-\gamma_{n,k})/\rho_{n,k}$ and then evaluate $z^{\ell_n}g_{n,k}(t)$. To evaluate f at d points, we use fast multipoint evaluation [17, 28] to evaluate the $g_{n,k}$.

Complexity analysis. Assume that z has a magnitude less than 2^{μ}. Finding the angular sector $A_{n,k}$ can be done with a binary search on the most significant bits for a total number of bit operations in $\widetilde{O}(\log d + m + \log\mu)$. Then the change of variable $t = (z-\gamma_{n,k})/\rho_{n,k}$ can be done within the same complexity.

Finally, we can evaluate $g_{n,k}(t)$ in $\widetilde{O}(m^2+\log\tau)$ bit operations with a classical Hörner scheme, and we can evaluate $z^{\ell_n} = e^{\ell_n \log z}$ in $\widetilde{O}((m+\log\mu+\log d)$ bit operations using arithmetico-geometric means for the logarithm and the exponential. Thus the total number of bit operations for one evaluation is in $\widetilde{O}(m^2+\log d+\log\tau+\log\mu)$.

Furthermore, to evaluate f on a set of d points of magnitude less than $\log\mu$, after gathering the points by angular sectors, using a fast multipoint evaluation algorithm [17, 28] to evaluate $g_{n,k}$ on each sector, leads to a number of bit operations in $\widetilde{O}(d(m+\log\mu+\log\tau))$.

Computation of the error bound. We can also compute the error bound efficiently with the explicit formula $(d+1)2^{-m}\hat{f}(z)$. Evaluating $\hat{f}(|z|)$ requires first to find the index j such that $\hat{f}(|z|) = |f_j||z|^j$. This amounts to find the vertex of the Newton polygon H belonging to the line of slope $\log|z|$ tangent to H. If the slopes of the edges of H are computed beforehand this can be done with a binary search $\widetilde{O}(\log d+m+\log\mu)$ bit operations. Then evaluating the formula can be done in $\widetilde{O}(m+\log\mu+\log d+\log\tau)$ bit operations. Thus, the error bound can be evaluated within the same number of operations as the approximate evaluation of f itself.

5 ROOT FINDING

Our algorithm to isolate the roots of f consists in computing first a piecewise approximation of f, then finding the approximations $\dot{\zeta}$ to the roots of each approximate polynomial, and finally, for each approximate root $\dot{\zeta}$, computing a disk centered on $\dot{\zeta}$ that contains a root ζ of f if ζ is well-conditioned. It is described in more details in Algorithm 3.

The main lemma that guarantees that our algorithm approximates correctly all the well-conditioned roots of f is the following.

Lemma 5.1. *With the notations of Algorithm 3:*

(i) *each $D(\dot{\zeta}, r) \in \dot{Z}$ contains a unique root ζ of f and the Newton algorithm starting at any point in the disk converges toward ζ*
(ii) *each root ζ of f such that $2\log(\mathrm{cond}(f,\zeta)) + 3\log_2(d+1) + 11 < m$ is contained in a disk $D(\dot{\zeta}, r) \in \dot{F}$*

To ensure that we find all the root ζ such that $\mathrm{cond}(f,\zeta) \leq 2^m$, it suffices to change m by $2m + 3\log_2(d+1) + 11$ before running the algorithm. We prove inequality (i) in Section 5.1 and inequality (ii) in Section 5.2

For the complexity of the algorithm, the dominating parts are: the computation of the piecewise approximation in step A, the $O(d/m)$ root approximations in step B, and the approximate evaluations $\dot{F}$ and $\dot{F}'$ in step C. The computation of the piecewise approximation and the evaluation can be done in $\widetilde{O}(d(m+\log\tau))$ bit operations, and each root approximation can be done in $\widetilde{O}(m^2)$ bit operations, which prove the complexity stated in Theorem 1.2.

5.1 Guarantee that the each returned disk isolate a root of f

Given a root ζ of f, the Newton bassin of ζ is the set of initial points such that the Newton method converges toward ζ. Using a bound on the second derivative of f, the Kantorovich's theory allows us to give a bound on the radius of a disk centered at $\dot{\zeta}$ that is included in the Newton bassin of ζ [7, §3.2],[21, Lemma 1]. Namely, let $r > 2|f(\dot{\zeta})/f'(\dot{\zeta})|$ and $K > |f''(y)/f'(\dot{z})|$ for all $y \in D(\dot{\zeta}, 4r)$. If $5rK \leq 1$ then, f has a unique root ζ in $D(\dot{\zeta}, r)$, and for all $z \in D(\dot{\zeta}, r)$, the Newton sequence starting from z converges to ζ.

In our case, remark that for all z such that $|z| \leq |\dot{\zeta}|2^{1/d}$, we have $|f''(z)| \leq d^2\widetilde{f}_{2,d}(|z|) \leq 2d^2\widetilde{f}_{2,d}(|\dot{\zeta}|) \leq 2d^2\widetilde{f}(|\dot{\zeta}|)/|\dot{\zeta}|^2 \leq 2d^3\hat{f}(|\dot{\zeta}|)/|\dot{\zeta}|^2$. This ensures that the number K computed on line 13 of Algorithm 3 is an upper bound on $\max_{z\in D(\dot{\zeta},r)}|f''(z)/f'(\dot{\zeta})|$.

Algorithm 3: Root isolation

Input: m a positive integer and a polynomial f of degree d
Output: List of isolating disks of the roots ζ of f such that $2\log_2(\mathrm{cond}(f,\zeta)) + 3\log_2(d+1) + 11 \le m$

```
# A. Compute the piecewise approximation
```
1 $A_m(f) \leftarrow$ the piecewise approximation of f

```
# B. Compute the approximate roots
```
2 $Z, \dot{Z} \leftarrow \{\}$
3 **for** (g, γ, ρ) *in* $A_m(f)$ **do**
4 $\quad \dot{g} \leftarrow$ Approximate factorization of g such that
5 $\quad\quad \|g - \dot{g}\|_1 \le 2^{-4m}\|g\|_1$
6 $\quad \dot{Z} \leftarrow \dot{Z} \cup \{\gamma + \rho\dot{t} \mid \dot{t}$ root of $\dot{g}\}$

```
# C. Evaluate f and f' on the approximate roots
```
7 $\varepsilon(z) \leftarrow z \mapsto (d+1)2^{-m}\hat{f}(|z|)$
8 $\varepsilon'(z) \leftarrow z \mapsto d^2 2^{-m}\hat{f}(|z|)/|z|$
9 $\dot{F} \leftarrow$ Approximate evaluations of f on $\dot{Z}$ with error ε
10 $\dot{F}' \leftarrow$ Approximate evaluations of f' on $\dot{Z}$ with error ε'

```
# D. Compute the isolating disks
```
11 **for** $(\dot{\zeta}, \dot{f}, \dot{f}') \in \dot{Z} \times \dot{F} \times \dot{F}'$ *s. t.* $\dot{f} \simeq f(\dot{\zeta})$ *and* $\dot{f}' \simeq f'(\dot{\zeta})$ **do**
12 $\quad r \leftarrow 2\frac{|\dot{f}| + \varepsilon(\dot{\zeta})}{|\dot{f}'| - \varepsilon'(\dot{\zeta})}$
13 $\quad K \leftarrow 2d^3 \frac{\hat{f}(|\dot{\zeta}|)}{|\dot{\zeta}|^2(|\dot{f}'| - \varepsilon'(\dot{\zeta}))}$
14 $\quad$ **if** $4r < |\dot{\zeta}|(2^{1/d} - 1)$ *and* $5rK < 1$
15 $\quad\quad Z \leftarrow Z \cup \{D(\dot{\zeta}, r)\}$
16 **return** L

Thus using the Kantorovich's theory, this ensures that when the criterion on line 14 of Algorithm 3 is satisfied, the returned disk isolates a root of f. Moreover, if two such disks intersect, then the root is in their intersection since the Newton method starting from any point in their intersection will converge toward a unique root in the their intersection.

5.2 Guarantee that each well-conditioned root is contained in a returned disk

Given a root ζ of f, the proof of inequality (ii) of Lemma 5.1 is done in two steps. First we show that $\dot{F}$ contains a root close enough to ζ. Then we show that the Kantorovich criterion on line 14 is satisfied.

In this section, we assume that ζ belongs to an angular sector $A_{n,k}$, and we let g be the corresponding approximate polynomial, and γ and ρ be the parameters for the change of variable $z = \gamma + \rho t$. Then $\dot{g}$ denotes the approximate polynomial of g obtained by factorization in line 4 of Algorithm 3. Finally, let $\dot{\zeta}$ be the closest point to ζ such that $\dot{g}((\dot{\zeta} - \gamma)/\rho) = 0$, and let $\theta = (\zeta - \gamma)/\rho$ and $\dot{\theta} = (\dot{\zeta} - \gamma)/\rho$, such that $f(\zeta) = 0$ and $\dot{g}(\dot{\theta}) = 0$.

5.2.1 Bound on the distance of the approximated roots. In this section we compute an upper bound on the distance between ζ and $\dot{\zeta}$. First it is known ([12, Theorem 6.4e], [3, Theorem 9]) that for a given point θ, the distance between θ and the closest root $\dot{\theta}$ of $\dot{g}$ is bounded by $\deg(\dot{g})|\dot{g}(\theta)/\dot{g}'(\theta)| \le d|\dot{g}(\theta)/\dot{g}'(\theta)|$. Then by construction, $\dot{g}$ satisfies $\left|g(\theta) - \dot{g}(\theta)\right| \le 2^{-4m}\|g\|_1$ and $\left|g'(\theta) - \dot{g}'(\theta)\right| \le d2^{-4m}\|g\|_1$.

Using Lemma 3.3 we have $\|g\|_1 \le (\delta+1)2^{-3m-1}\hat{f}(\|\zeta|)/|\zeta|^{\ell_n}$. With the triangular inequality, combining the inequality on $|g(\theta) - \dot{g}(\theta)|$ with the inequalities (iii) of Lemma 2.1 and (ii) of Lemma 3.1, leads to $\left|\dot{g}(\theta)\right| \le 2(d+1)2^{-m}\widetilde{f}(|\zeta|)/|\zeta|^{\ell_n}$.

For the derivative, we can show with similar arguments that $|\rho f'_{\ell_n,u_n}(\gamma+\rho\theta) - \dot{g}'(\zeta)| \le 2(d+1)^2\widetilde{f}(|\zeta|)/|\zeta|^{\ell_n}$. We can also deduce from inequality (iii) of Lemma 2.1 that $\rho|f'(\zeta) - \zeta^{\ell_n} f'_{\ell_n,u_n}(\zeta)| \le \rho d^2\widetilde{f}(|\zeta|)/|\zeta|$. Finally, remark that $|\zeta|/\rho \le r_{n+1}/((3/4)(r_{n+1} - r_n)) \le (4/3)(1/(1 - r_n/r_{n+1})) \le (4/3)(1/(1 - 2^{-m/2d})) \le (4/3)(2d/m + 1)$. This implies that for $m \ge 4$ and $d \ge 1$ we have $|\zeta|/\rho \le d+1$ and $\left|\dot{g}'(\theta) - \rho f'(\zeta)/|\zeta|^{\ell}_n\right| \le 3\rho(d+1)^3 2^{-m}\widetilde{f}(|\zeta|)/|\zeta||\zeta|^{\ell_n}$.

Combining all the previous inequalities, this implies that for $m \ge \log_2(\mathrm{cond}(f,\zeta)) + 3\log_2(d+1) + 4$ we have:

$$\frac{|\zeta - \dot{\zeta}|}{|\zeta|} \le 4(d+1)2^{-m}\,\mathrm{cond}(f,\zeta)$$

5.2.2 Guarantee that each well-conditioned root is returned. Finally, to prove the correctness of Algorithm 3, it remains to prove that if $2\,\mathrm{cond}(f,\zeta) + 3\log_2(d+1) + 11 \le m$, then the criterion on line 14 applied to $\dot{\zeta}$ evaluates to true. Although the proof is technical, the main idea is that the variable r is roughly proportional to $|\zeta - \dot{\zeta}|$. Thus, as soon as $|\zeta - \dot{\zeta}|$ is small enough, rK will become small enough to validate the criterion.

First, if $m > \mathrm{cond}(f,\zeta) + 3\log_2(d+1) + 4$, the bound on $|\zeta - \dot{\zeta}|$ ensures that $|\zeta|2^{-1/d} \le |\dot{\zeta}| \le |\zeta|2^{1/d}$. In particular this implies $\widetilde{f}(|\dot{\zeta}|) \le 2\widetilde{f}(|\zeta|)$. Moreover for all points of module less than $|\zeta|2^{1/d}$ we have $|f'(z)| \le 2d\widetilde{f}(|\zeta|)/|\zeta|$ and $|f''(z)| \le 2d^2\widetilde{f}(|\zeta|)/|\zeta|^2$.

Thus we can bound $|f(\dot{\zeta})|$ and $|f'(\dot{\zeta})|$ using a Taylor expansion around ζ with $|f(\dot{\zeta})| \le |\zeta - \dot{\zeta}||f'(\zeta)| + d^2|\zeta - \dot{\zeta}|^2\hat{f}(|\zeta|)/|\zeta|^2$ and $|f'(\dot{\zeta})| \ge |f'(\zeta)| - 2d^2|\zeta - \dot{\zeta}|\widetilde{f}(|\zeta|)/|\zeta|^2$.

Letting $C = 6(d+1)\,\mathrm{cond}(f,\zeta)$, this leads to $|\dot{f}| + \varepsilon(\dot{\zeta}) \le |\zeta||f'(\zeta)|(2^{-m}C + d2^{-2m}C^3)$ and $|\dot{f}'| - \varepsilon'(\dot{\zeta}) \ge |f'(\zeta)|\left(1 - d2^{-m}C^2 - d2^{-m}C\right)$.

Thus as soon as $d2^{-m}C(C+1) < 1/2$, we have $r \le 6\cdot 2^{-m}C|\zeta|$ and $K \le 8d^3\widetilde{f}(|\zeta|)/(|\zeta|^2|f'(\zeta)|)$, such that $5rK \le 40d2^{-m}C^2$, which is below 1 when $40d2^{-m}C^2 < 1$, and $r/|\dot{\zeta}| \le 6\cdot 2^{-m}C2^{1/d}$, which is below $\log(2)/d < 2^{1/d} - 1$ when $18d\cdot 2^{-m}C < \log(2)$. All those constraints are satisfied for $1500(d+1)^3\,\mathrm{cond}^2(f,\zeta) < 2^m$, which is satisfied for $m > 2\log_2(\mathrm{cond}(f,\zeta)) + 3\log_2(d+1) + 11$.

6 BENCHMARKS AND APPLICATIONS

The approaches presented here are not only of theoretical interest, and can be applied to solve problems involving well-conditioned polynomials of large degrees with coefficients of different orders of magnitude, for instance truncated series expansions of Gaussian analytic functions, giving polynomials with repulsion at the first order, or truncated series expansions of hypergeometric functions.

We demonstrate their practical efficiency for several families of well-conditioned polynomials that are random polynomials associated with hyperbolic, elliptic and flat distributions, as well as more structured polynomials as resultant of random bivariate polynomials appearing when solving multi-variate systems with elimination.

For $d >= 0$ we define the hyperbolic, elliptic and flat bases as $(\mathcal{H}_i)_{0\leq i\leq d} := (1)_{0\leq i\leq d}$, $(\mathcal{E}_i)_{0\leq i\leq d} := \left(\sqrt{\binom{d}{i}}\right)_{0\leq i\leq d}$ and $(\mathcal{F}_i)_{0\leq i\leq d} := \left(1/\sqrt{i!}\right)_{0\leq i\leq d}$. A polynomial f can be decomposed as

$$f(z) = \sum_{i=0}^{d} a_i\mathcal{H}_i z^i = \sum_{i=0}^{d} b_i\mathcal{E}_i z^i = \sum_{i=0}^{d} c_i\mathcal{F}_i z^i.$$

f is said random hyperbolic (respectively elliptic, flat) when the a_i's (respectively the b_i's, c_i's) are random numbers.

6.1 Implementation

Our prototype implementation[1] in `C` of the algorithms presented in this article is based on the `C` libraries `Arb`[2] (which provides multi-precision ball arithmetic) and `MPSolve`[3] (see [4], which provides a solver for univariate polynomials relying on Ehrlich's, aka Aberth's, iterations with multiprecision floating point arithmetic).

`PWEval` implements the evaluation process described in Sec. 4 and `PWRoots` the root isolation algorithm 3; they take in input a polynomial an f and an m and compute a piecewise approximation of f using ball arithmetic. `PWRoots` outputs a set of d' pairwise disjoints complex discs. If $d' = d$, each root of f is isolated in a disc of the output. Step 4 of algorithm 3 is achieved by applying the solver using secular equations of `MPSolve`. `PWEval` also takes in input a set Z of points in $\mathbb{C}$ and output for each $z \in Z$ a disc containing $f(z)$ of radius less than $2^{-m}d\hat{f}(|z|)$. Points are evaluated one by one (without multi-point evaluation).

We introduced in our implementation a slight variation of Algo. 1: in step 12, we compute s_{n+1} as $s_n + c\frac{m}{u_n - \ell_n + 1}$ where c is a real positive parameter in $O(1)$. While preserving the bit complexity of the overall approach this changes the subdivision of the complex plane in sectors and the piecewise approximation in the following way: the greatest is c, the widest are the rings, the less are the numbers of angular sectors in rings and the degrees of approximations and the fastest is the computation of the piecewise approximation because there are less approximations to compute.

Figures 8 and 9 show the Newton polygons and the sectors partitions of the complex plane for hyperbolic, elliptic and flat random polynomials obtained with our implementation with $c = 1$.

We observed that the bottleneck in `PWEval` was the computation of the piecewise approximation; we choose $c = 7/2$ in `PWEval` which makes this step faster. In `PWRoots`, the dominant step is the approximation of the roots of the approximating polynomials with `MPSolve` which complexity is quadratic in their degrees; we choose $c = 2/5$ to produce approximations with smaller degrees.

6.2 Numerical results

All the times given below are sequential times in seconds on a `Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz` machine on Linux.

[1] available at https://gitlab.inria.fr/gamble/pwpoly
[2] https://arblib.org
[3] https://numpi.dm.unipi.it/scientific-computing-libraries/mpsolve/

The input polynomials we consider in our tests are 53-bits floating point approximations of random dense hyperbolic, elliptic and flat polynomials which integers coefficient factors with a uniform law in the interval $[2^{-8}, 2^8]$. For root isolation, we also consider 53-bits floating point approximations of resultant polynomials of two bivariate random dense hyperbolic polynomials.

We generate sets of random complex points with rational real and imaginary parts with numerator and denominators uniformly chosen in $[2^{-16}, 2^{16}]$.

6.2.1 Evaluation. We used `PWEval` to evaluate random dense hyperbolic, elliptic and flat polynomials of increasing degree d at d points. We compare the running time of `PWEval` with the time required to evaluate f at the same set of d points with the a rectangular splitting algorithm (see [6, 16]) available in `Arb`. We choose m (the output precision for `PWEval`) and the precision for `Arb` so that the medians of the $\log_2$ of the errors relative to $\widetilde{f}$ of the evaluations is about $-50 \pm 10\%$. In our tests, evaluations with Horner's rule were always slower than evaluations with rectangular splitting.

Figure 5 shows those running times in an histogram while detailing for `PWEval` the time for computing the piecewise approximation and the time for evaluating it at the d points (respectively called "pw comp" and "pw eval" in fig. 5). `PWEval` becomes faster than the evaluation with rectangular splitting of f for degrees above 40000. Notice also that once the piecewise approximation is computed, evaluating it at a single point is several orders of magnitude faster than one evaluation (at a single point) with rectangular splitting.

6.2.2 Root isolation. We used `PWRoots` to isolate the roots of the polynomials of our test suite with $m = 2(30 + \lceil \log_2(d+1) \rceil)$ which allowed `PWRoots` to always isolate all the roots.

We first compare the running times of `PWRoots` and `HCRoots` which is a `C` implementation of the root finding algorithm of [21][Sec 5, Algo. 3]. `HCRoots` computes a piecewise approximation at a given precision m which is dedicated to hyperbolic polynomials; the root isolation process is embedded in a loop where m is doubled while all the roots are not isolated. When m reaches d, a subdivision root isolator with a complexity quadratic in d is used. We give in table 1 running times of `PWRoots` and `HCRoots` for small values of d. For hyperbolic polynomials, the running times of both `HCRoots` and `PWRoots` are more or less linear in d; `HCRoots` is faster by a constant factor due to specificities of the implementations. `HCRoots` is not well adapted to other weights than the hyperbolic ones and in these cases its running time is dominated by the one of the quadratic solver for f, as depicted in table 1 (elliptic and flat cases).

We also compare `PWRoots` and `MPSolve` for hyperbolic, elliptic and flat random dense polynomials of degrees up to 20000 in fig. 6 where one can observe the linear complexity in d of `PWRoots` (for the well conditionned polynomials in consideration) in contrast to the one of `MPSolve` which is quadratic in d.

Figure 7 compares `PWRoots` and `MPSolve` for isolating the roots of a resultant polynomial of degree d of two bivariate random dense hyperbolic polynomials of degree $\sqrt{d}$, for $\sqrt{d}$ up to 100. `PWRoots` becomes faster than `MPSolve` for d above 60^2.

REFERENCES

[1] S. Basu, R. Pollack, and M.-R. Roy. *Algorithms in Real Algebraic Geometry.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[2] Ruben Becker, Michael Sagraloff, Vikram Sharma, and Chee Yap. A near-optimal subdivision algorithm for complex root isolation based on the pellet test and newton iteration. *Journal of Symbolic Computation*, 86:51–96, 2018.

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

[4] Dario A Bini and Leonardo Robol. Solving secular and polynomial equations: A multiprecision algorithm. *J. of Computational and Applied Mathematics*, 272:276–292, 2014.

[5] Richard P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In J.F. Traub, editor, *Analytic Computational Complexity*, pages 151–176. Academic Press, 1976.

[6] Richard P Brent and Paul Zimmermann. *Modern computer arithmetic*, volume 18. Cambridge University Press, 2010.

[7] Jean-Pierre Dedieu. *Points fixes, zéros et la méthode de Newton*. Mathématiques et Applications. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[8] Xavier Gourdon. Algorithmique du theoreme fondamental de l'algebre. Research Report RR-1852, INRIA, 1993.

[9] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[10] Stef Graillat. Accurate simple zeros of polynomials in floating point arithmetic. *Computers & Mathematics with Applications*, 56(4):1114–1120, 2008.

[11] A. A. Grau. On the reduction of number range in the use of the graeffe process. *J. ACM*, 10(4):538–544, oct 1963.

[12] Peter Henrici. *Applied and computational complex analysis, Vol. 1.* Wiley, New York, 1974.

[13] Peter Henrici. *Applied and computational complex analysis, Volume 3: Discrete Fourier analysis, Cauchy integrals, construction of conformal maps, univalent functions*, volume 41. John Wiley & Sons, 1993.

[14] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[15] Rémi Imbach and Victor Y Pan. Root radii and subdivision for polynomial root-finding. In *Computer Algebra in Scientific Computing: 23rd International Workshop, CASC 2021, Sochi, Russia, September 13–17, 2021, Proceedings 23*, pages 136–156. Springer, 2021.

[16] Fredrik Johansson. Evaluating parametric holonomic sequences using rectangular splitting. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 256–263, 2014.

[17] Alexander Kobel and Michael Sagraloff. Fast approximate polynomial multipoint evaluation and applications. *arXiv preprint arXiv:1304.8069*, 2013.

[18] Gregorio Malajovich and Jorge P Zubelli. On the geometry of graeffe iteration. *journal of complexity*, 17(3):541–573, 2001.

[19] Gregorio Malajovich and Jorge P Zubelli. Tangent graeffe iteration. *Numerische Mathematik*, 89:749–782, 2001.

[20] Kurt Mehlhorn, Michael Sagraloff, and Pengming Wang. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 66:34–69, 2015.

[21] Guillaume Moroz. New data structure for univariate polynomial approximation and applications to root isolation, numerical multipoint evaluation, and other problems. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1090–1099. IEEE, 2022.

[22] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, Serge Torres, et al. *Handbook of floating-point arithmetic*. Springer, 2018.

[23] Alexandre Ostrowski. Recherches sur la méthode de graeffe et les zéros des polynomes et des séries de laurent. *Acta Mathematica*, 72(1):157–257, 1940.

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

[25] Victor Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *Journal of Symbolic Computation*, 33(5):701–733, 2002.

[26] Peter Ritzmann. A fast numerical algorithm for the composition of power series with complex coefficients. *Theoretical Computer Science*, 44:1–16, 1986.

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

[28] Joris van der Hoeven. Fast composition of numeric power series. Technical Report 2008-09, Université Paris-Sud, Orsay, France, 2008.

A BENCHMARKS

	$d = 200$		$d = 400$		$d = 800$		$d = 1600$	
	PWRoots	HCRoots	PWRoots	HCRoots	PWRoots	HCRoots	PWRoots	HCRoots
hyperbolic	1.10	0.23	2.50	0.43	6.84	0.92	13.20	1.95
elliptic	1.63	10.8	3.48	70.2	11.5	611	17.8	> 999
flat	1.69	19.6	3.46	68.7	9.46	511	22.0	> 999

Table 1: Running times in seconds for isolating the complex roots of random dense hyperbolic, elliptic and flat polynomials of increasing degrees d with `PWRoots` and `HCRoots`.

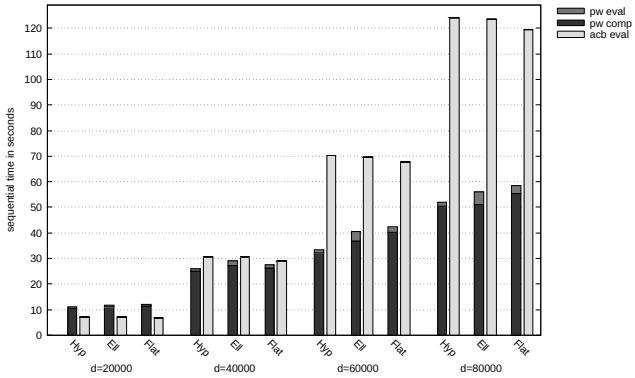

Figure 5: Evaluating random dense polynomials of degree d at d random points with `PWEval` and the rectangular splitting evaluation in `Arb`.

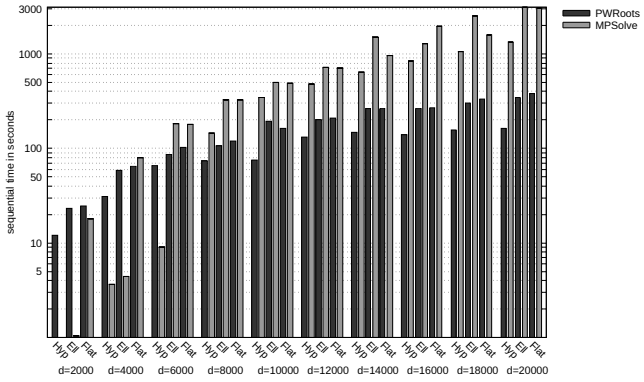

Figure 6: Complex roots isolation of random dense hyperbolic, elliptic and flat polynomials of increasing degrees d with `PWRoots` and `MPSolve`.

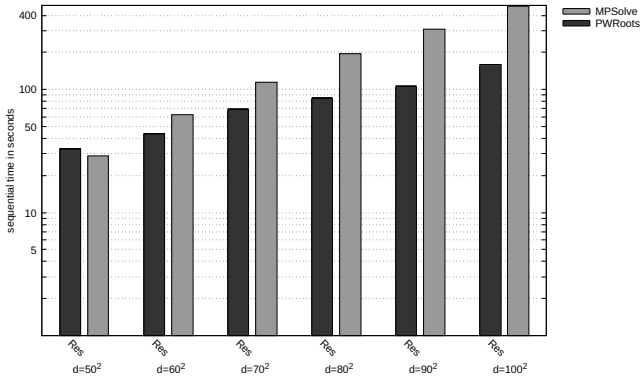

Figure 7: Complex roots isolation of the resultant polynomial of two random dense hyperbolic polynomials of increasing degrees $\sqrt{d}$ with `PWRoots` and `MPSolve`.

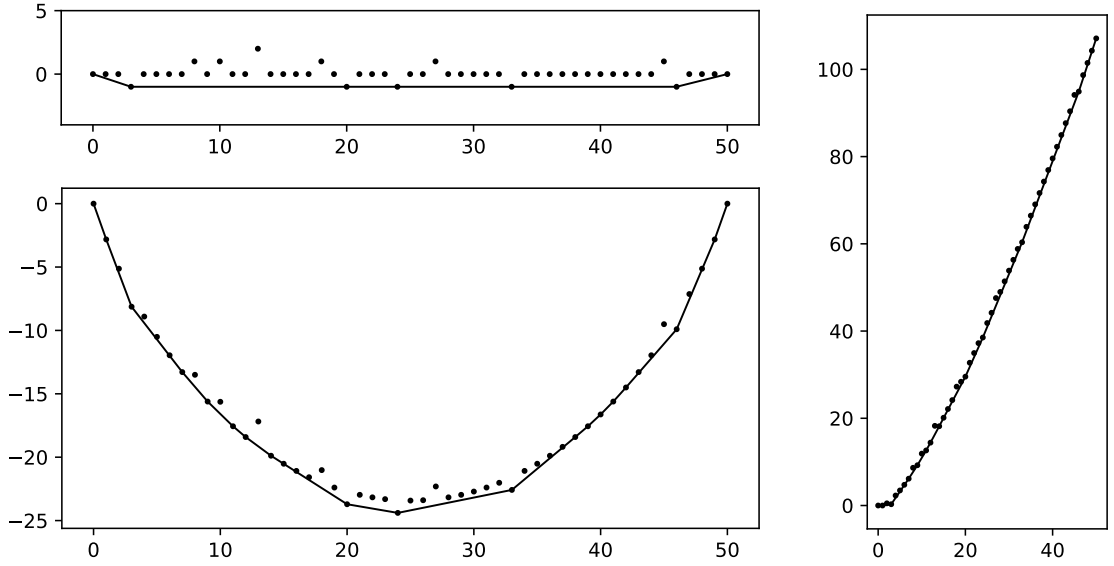

Figure 8: Newton polygon of a hyperbolic (top left), elliptic (bottom left), and flat (right) polynomial of degree 50

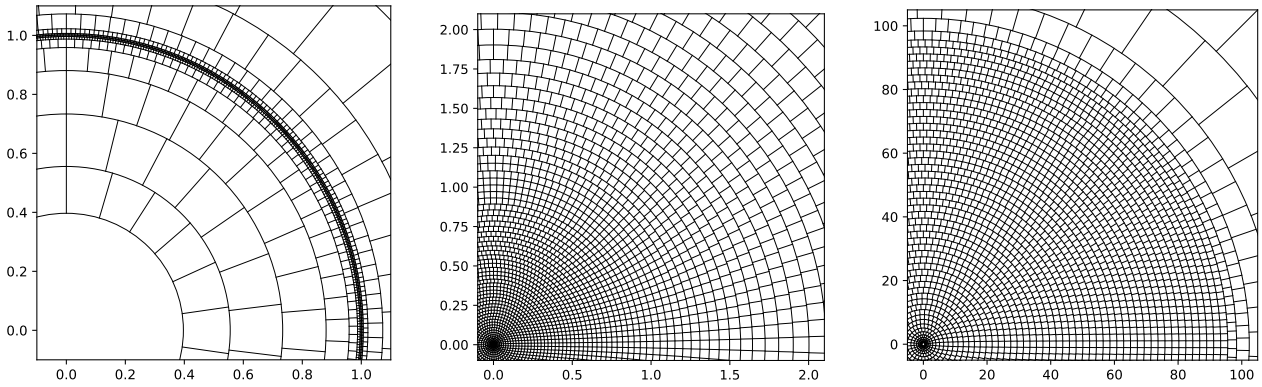

Figure 9: Sector partitions associated to a hyperbolic (left), elliptic (middle) and flat (right) polynomial of degree 10 000