New data structure for univariate polynomial approximation and applications to root isolation, numerical multipoint evaluation, and other problems

Guillaume Moroz

New data structure for univariate polynomial approximation and applications to root isolation, numerical multipoint evaluation, and other problems

Guillaume Moroz

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

November 17, 2021

Abstract

We present a new data structure to approximate accurately and efficiently a polynomial f of degree d given as a list of coefficients f_i. Its properties allow us to improve the state-of-the-art bounds on the bit complexity for the problems of root isolation and approximate multipoint evaluation. This data structure also leads to a new geometric criterion to detect ill-conditioned polynomials, implying notably that the standard condition number of the zeros of a polynomial is at least exponential in the number of roots of modulus less than $1/2$ or greater than 2.

Given a polynomial f of degree d with $\|f\|_1 = \sum |f_i| \leq 2^\tau$ for $\tau \geq 1$, isolating all its complex roots or evaluating it at d points can be done with a quasi-linear number of arithmetic operations. However, considering the bit complexity, the state-of-the-art algorithms require at least $d^{3/2}$ bit operations even for well-conditioned polynomials and when the accuracy required is low. Given a positive integer m, we can compute our new data structure and evaluate f at d points in the unit disk with an absolute error less than 2^{-m} in $\widetilde{O}(d(\tau+m))$ bit operations, where $\widetilde{O}(\cdot)$ means that we omit logarithmic factors. We also show that if κ is the absolute condition number of the zeros of f, then we can isolate all the roots of f in $\widetilde{O}(d(\tau+\log\kappa))$ bit operations. Moreover, our algorithms are simple to implement. For approximating the complex roots of a polynomial, we implemented a small prototype in `Python/NumPy` that is an order of magnitude faster than the state-of-the-art solver `MPSolve` for high degree polynomials with random coefficients.

Keywords: Polynomial evaluation, Complex root finding, Condition number

1 Introduction

One of the fundamental problem in computer algebra is the evaluation of polynomials. Since 1972, it is known that evaluating a univariate polynomial of degree d on d points can be done in a quasi-linear number of arithmetic operations [17]. Unfortunately, this bound doesn't hold if we consider the bit complexity, where the arithmetic operations performed with a precision of m bits costs $\widetilde{O}(m)$ bit operations. If we want to evaluate approximatively a polynomial on d points up to a constant absolute error, a direct application of Fiduccia algorithm leads to a bit-complexity bound in $\widetilde{O}(d^2)$ bit operations, and a more sophisticated algorithm provides a bound in $\widetilde{O}(d^{3/2})$ [45]. For almost 50 years, the following problem has remained open.

> Given a polynomial f of degree d with coefficients of constant size, and d complex points x_k in the unit disk, is it possible to compute all the $f(x_k)$ up to a constant absolute error with a number of bit operations quasi-linear in d ?

Nevertheless, the evaluation of polynomials on multiple points is used in many areas of computer science, such as polynomial system solving with the Newton method [43, 24, 12, 8, 5], homotopy continuation [11, 10, 4, 31, 8] or subdivision algorithms [36, 29, 34], visualisation of algebraic surfaces through raytracing or mesh computation [47], among others. Speeding up the numerical evaluation of polynomials may lead to an effortless practical improvement for many existing algorithms.

We introduce in Sections 1.1 and 3 a new data structure, that allows us to finally solve this problem. It approximates the input polynomial by a piecewise polynomial on a carefully chosen domain that depends solely on the degree of the input polynomial, and the required precision. The fact that the domain is fixed and not adaptive makes it easy to implement.

Moreover, we show that our new data structure improves not only the bound for the numeric multipoint evaluation problem (Sections 1.2.1 and 4), but also the bound for the root isolation problem (Sections 1.2.2 and 5), and the lower bound on the condition number of polynomials (Sections 1.2.3 and 6). We expect that our approach will lead further improvements for other related problems, notably multivariate polynomial evaluation and polynomial system solving. As a proof of concept, we also implemented the root isolation algorithm presented in this article. Even though our implementation is less than 100 lines of code written in `Python`, it is an order of magnitude faster than the state-of-the-art optimized implementation `MPSolve` for high degree polynomials with random coefficients (Section 1.2.4 and source code in Appendix A).

1.1 The data structure

Given a polynomial f of degree d and an integer $m > 1$, we will introduce in Definition 2 the so-called an *m-hyperbolic approximation* of f, which can be seen as a piecewise approximation of f by polynomials of degree $m-1$. The key that will allow us to improve the state-of-the-art complexity bounds of several classical problems related to univariate complex polynomials is the hyperbolic layout used to compute this piecewise approximation. We first define this layout so-called *hyperbolic covering*, illustrated in Figure 1. Roughly, a hyperbolic covering is a set of disks of radius exponentially smaller near the unit circle, and such that their union contains the unit disk.

Definition 1. *Given a positive integer N, an N-*hyperbolic covering *of the unit disk is the set of disks of centers $\gamma_n e^{i2\pi\frac{k}{K_n}}$ and radii ρ_n for $0 \leq n < N$ and $0 \leq k < K_n$, where γ_n, ρ_n and K_n are defined by:*

$$r_n = \begin{cases} 1 - \frac{1}{2^n} & \text{if } 0 \leq n < N \\ 1 & \text{if } n = N \end{cases}$$

$$\gamma_n = \frac{1}{2}(r_n + r_{n+1})$$

$$\rho_n = \frac{3}{4}(r_{n+1} - r_n)$$

$$K_n = \begin{cases} 4 & \text{if } n = 0 \\ \lceil \frac{3\pi}{\sqrt{5}} \frac{r_{n+1}}{\rho_n} \rceil & \text{otherwise} \end{cases}$$

Remark 1. *We can also write explicitly the corresponding sequences $(\gamma_n)_{n=0}^{N-1}$ and $(r_n)_{n=0}^{N-1}$:*

$$\gamma_n = \begin{cases} 1 - \frac{3}{4}\frac{1}{2^n} & \text{if } 0 \leq n \leq N-2 \\ 1 - \frac{1}{2}\frac{1}{2^n} & \text{if } n = N-1 \end{cases}$$

$$\rho_n = \begin{cases} \frac{3}{8}\frac{1}{2^n} & \text{if } 0 \leq n \leq N-2 \\ \frac{3}{4}\frac{1}{2^n} & \text{if } n = N-1 \end{cases}$$

We also have explicitly $2^{n+1} \leq K_n \leq 2^{n+4}$*, using* $\frac{1}{2} \leq r_{n+1} \leq 1$ *and* $\frac{3}{8}\frac{1}{2^n} \leq \rho_n \leq \frac{3}{4}\frac{1}{2^n}$.

We will see in Lemma 3 that for all integers $N \geq 1$, the union of the disks of a N-hyperbolic covering contains the unit disk.We can now define the m-hyperbolic approximation of a polynomial f, that can be seen as piecewise approximation of f by polynomials of degree lower than m.

Definition 2. *Given a polynomial* f *of degree* d *with* $\|f\|_1 \leq 2^\tau$*, and an integer* $m > 1$*, an* m-hyperbolic approximation *of* f *is a finite set of pairs* (g, a)*, where* g *is a polynomial of degree* $\widetilde{m} = \min(m-1, d)$*, with coefficients of bit size* $O(\tau + m)$*, and* a *is an affine transform, such that:*

- *the set of disks* $a(D(0,1))$ *is the* N*-hyperbolic covering with* $N = \lceil \log_2\left(\frac{3ed}{\widetilde{m}}\right) \rceil$
- $\|f \circ a - g\|_1 \leq 3\|f\|_1 2^{-m}$

Since we constrain the polynomials $g(X)$ approximating $f(a(X))$ to have a degree $m-1$, that can be significantly smaller than d, it is not obvious that an approximation satisfying the conditions defined above always exists. The following theorem proves that it always exists, and furthermore that it can be computed in quasi-linear time with respect to the degree of f.

Theorem 1. *Let* f *be a polynomial of degree* d *with* $\|f\|_1 \leq 2^\tau$*, and* $m > 1$ *be an integer. Algorithm 1 computes an* m*-hyperbolic approximation of* f*, denoted by* $H_{d,m}(f)$*, in* $\widetilde{O}(d(m+\tau))$ *bit operations.*

Remark also that the maximal precision required for the arithmetic operations in Algorithm 1 is in $O(\tau+m+\log d)$, which makes it suitable for an implementation with machine precision arithmetic.

Main idea of the proof of Theorem 1. Denoting the center and the radius of a disk in a m-hyperbolic approximation by $\gamma_{n,k} = \gamma_n e^{i2\pi k/K_n}$ and ρ_n , we need to prove that it is possible to compute a polynomial of degree $m-1$ that satisfies the bounds of Theorem 1. This comes from the fact that using the formula of Definition 1 and Remark 1, we can check that the coefficients of degree ℓ of the polynomial $f(\gamma_{n,k} + \rho_n X)$ have a modulus less than $\|f\|_1/2^\ell$ for all $\ell \geq m$. Then it remains to prove that we can approximate the first m coefficients of all the $f(\gamma_{n,k} + \rho_n X)$ in $\widetilde{O}(d(m+\tau))$ bit operations. For that, remark that $0 \leq n < N = O(\log d)$. Thus, it is sufficient to prove that for a fixed n, we can compute $f(\gamma_{n,k} + \rho_n X) \mod X^m$ for all k in $\widetilde{O}(d(m+\tau))$ bit operations. This can be done by using a combination of fast numerical composition of series (Proposition 3), and numerical fast Fourier transform (Proposition 2). More details can be found in Section 3.

1.2 Applications

Based on our new data structure, we describe three independent results that improve state-of-the-art solutions to long-standing problems. First we improve the complexity for evaluating numerically polynomials on multiple points. Then we improve the complexity of finding the roots of well-conditioned polynomials. Finally, we present a new lower bound on the condition number of the zeros of a polynomial, based on simple geometric properties of the distribution of its roots.

1.2.1 Numerical multipoint evaluation

In the literature [17, 2], [46, Chapter 10], a fast multipoint evaluation algorithm has been designed to evaluate d points of a degree d polynomial with a number of arithmetic operations quasi-linear in d. However, in the case of numerical evaluation with precision m after the binary point, this algorithm uses a number of bit operations quadratic in d, even for a constant m. This is due notably to the fact

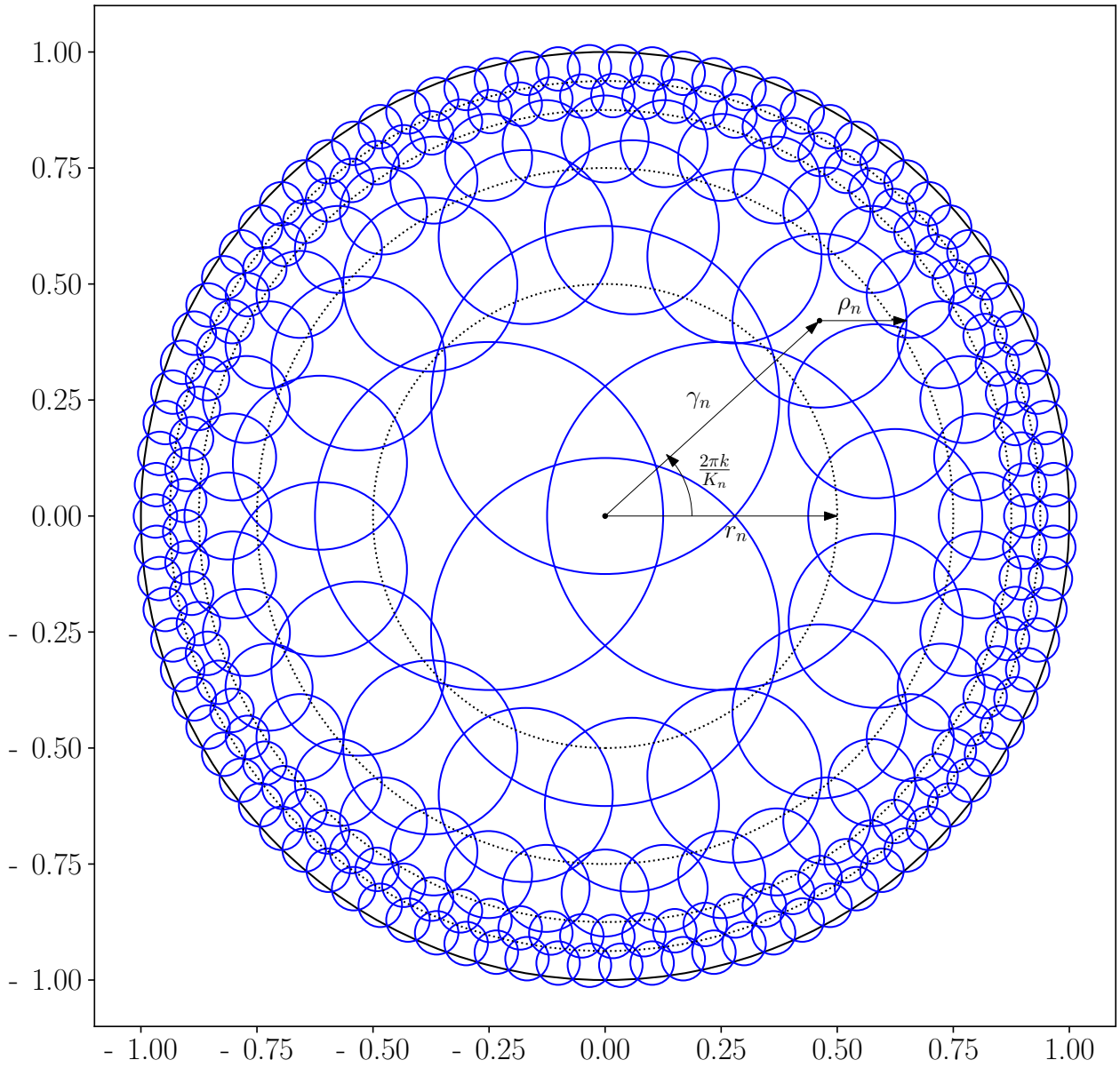

Figure 1: 5-hyperbolic covering

that this algorithm introduces intermediate polynomials with coefficients that may have a bit-size linear in d. A careful analysis of the bit-complexity of this algorithm for numerical evaluation ([45, Lemma 11]) shows that this algorithm uses $\widetilde{O}(d(m+d))$ bit operations.

Specific sets of points were found where the problem of numeric multipoint evaluation can be solved in a quasi-linear time in d. The most famous one is the set of roots of unity. Computing a numerical approximation of the $f(x_k)$ for $x_k = e^{i2\pi\frac{k}{d+1}}$ can be done in quasi-linear time [41] in d. Another family of points was used by Ritzmann for the problem of fast numeric composition of series [39, Proposition 3.4]. He showed that if the modulus of the x_k is lower than $\frac{1}{3d}$, then all the $f(x_k)$ can be approximated numerically in a quasi-linear number of bit operations.

In a more recent work [45, §3.2], van der Hoeven gave the first sub-quadratic bound to evaluate d points with modulus less than 1. More precisely, he showed that if $\|f\|_1 < 1$ it is possible to evaluate the $f(x_k)$ with an error less than 2^{-m} with $\widetilde{O}(d^{3/2}m^{3/2})$ bit operations. He achieved this bound by subdividing the unit disk in annuli of constant width. A drawback of this bound is that it increases the exponent on the required precision m. By contrast, in our data structure, we subdivide the unit disk with disks of width exponentially smaller near the unit circle. This approach allows us to finally derive an algorithm that is both quasi-linear in d and in m for the numerical multipoint evaluation problem.

Theorem 2. *Given a polynomial f of degree d with $\|f\|_1 \leq 2^\tau$, Algorithm 2 returns the evaluation of f on d points in the unit disk with an absolute error less than $\|f\|_1 2^{-m}$ in $\widetilde{O}(d(\tau+m))$ bit operations.*

Remark 2. *Using Algorithm 2 on the reverse polynomial $X^d f(1/X)$, we see that the same complexity bound holds for a set of d points in the complex plane, replacing f by the function*

$$\widetilde{f}(x) = \begin{cases} f(x) & \text{if } x \in D(0,1) \\ f(x)/x^d & \text{otherwise} \end{cases}.$$

Our algorithm is particularly well-suited for evaluating polynomials of high degree with fixed constant precision such as machine precision. This arises in many applications, notably in polynomial root approximation. One of the most famous method to approximate a root of a polynomial is the Newton method. Starting from an initial point x_0, it consists in computing iteratively the Newton map, yielding the sequence $x_{n+1} = x_n - f(x_n)/f'(x_n)$. It was shown that starting from an explicit set of $3.33d\log^2 d$ points, this approach is guaranteed to approximate all the roots of a given polynomial [24, 5]. In their work, the authors only show experiences with polynomial given by recursive formula, such that they can be evaluated with a number of operations logarithmic in their degree. Our data structure can improve their approach for dense polynomials given by their list of coefficients. In Section 1.2.2, we focus on the computation of disks isolating the roots of f, that is the computation of a set of disks that are pairwise disjoint and that contain a unique root of f each.

Main idea of the proof of Theorem 2. Once we have computed a m-hyperbolic approximation of the input polynomial f, the algorithm is rather straight-forward. For each disk of the corresponding hyperbolic covering, we can efficiently find all the input points that it contains, using a geometric data-structure such as range searching, recalled in Section 2.7. Then, using the state-of-the-art algorithm for multipoint evaluation (Proposition 4), we can evaluate the corresponding approximate polynomial g from the hyperbolic approximation on the selected points. Since g has a degree lower than m, evaluating g on the x_i up to precision m can be done with an amortized time $\widetilde{O}(m)$ per point if the number of points is greater than m and in a total time $\widetilde{O}(m^2)$ if there are less than m

points. Since the number of disks in a hyperbolic approximation is in $O(d/m)$, this leads to a bit complexity in $\widetilde{O}(dm)$. More details can be found in Section 4.

1.2.2 Root isolation

Our data structure also allows us to improve the state-of-the-art bound on the bit complexity for the problem of isolating all the complex roots of a given polynomial, with a bound that is adaptive in the condition number of the input. The condition number of the zeros of a polynomial measures the displacement of its roots with respect to a perturbation of its coefficients (Definition 3). For a square-free polynomial f and a root ζ, we let $\kappa_\zeta = \frac{\max(1,|\zeta|^d)}{|f'(\zeta)|}$. Then the absolute condition number of f is $\kappa = \max_{f(\zeta)=0}(\kappa_\zeta)$. Our goal is to provide for each root an isolating disk, that is a disk that contains a unique root and doesn't intersect any other isolating disk. In our case we consider so-called *projective disks*, that is either a disk or the inverse of a disk D defined as the set of points $1/x$ such that $x \in D$ and $x \neq 0$.

Theorem 3. *Given a square-free polynomial f of degree d with $\|f\|_1 \leq 2^\tau$, with absolute condition number κ, Algorithm 3 computes isolating projective disks of all the roots of f in $\widetilde{O}(d(\tau + \log\kappa))$ bit operations.*

Remark 3. *If all the coefficients of f are real numbers, then Algorithm 3 can be slightly modified to return all the real roots of f with the same bit-complexity, by selecting the isolating projective disks that intersects the real axis.*

In recent works on complex root isolation [33, 3], the best adaptive complexity bound, rewritten with our notation, is in $\widetilde{O}\left(d\sum_{f(\zeta)=0}[\tau + \max(1, \log(\kappa_\zeta)) + \max(1, \log(\sigma_\zeta^{-1}))]\right)$, where $\sigma_\zeta = \min_{f(\eta)=0,\eta\neq\zeta}(|\zeta - \eta|)$. Our bound removes the dependency in σ_ζ, and replaces the sum by a max, which improves the state-of-the-art bounds by a factor d if the condition numbers of the roots are logarithmic in d or evenly distributed. In adaptive algorithms that compute isolating disks, a criterion is used for early termination. It usually checks if a disk or a rectangle contains a unique root of f. Some examples of criteria are detailed in Section 2.6. All those criteria end up evaluating a polynomial of degree roughly d on each of the d roots ([33, §2.2.3],[3, Lemma 5], [26, Algorithms 2 and 3], ...), which leads to a bit-complexity at least quadratic in d if we use a naive evaluation algorithm, or in $d^{3/2}$ using state-of-the-art multipoint evaluation methods [45]. In our case, thanks to our new data structure, the criterion to check that a disk contains a unique root is replaced by the evaluation of a polynomial of degree $O(\tau + \log(\kappa))$, which explains partly how we avoid a cost quadratic in d.

Another approach in the literature consists in computing the roots up to a precision high enough such that we can guarantee that all the roots are approximated correctly. An explicit bound exists in the case where the input polynomial has integer coefficients. Schönhage showed [42, §20]) that if we can compute $\widetilde{f}$ an approximate factorization of f close enough, then the roots of f are isolated by disks centered on the roots of $\widetilde{f}$ with a radius depending on the condition number κ. Combined with a bound on κ from Mahler [32, last inequality] for polynomials with integer coefficients, and using the algorithm of Pan [37] to compute the approximate factorization $\widetilde{f}$, this leads to a root-isolating algorithm in $\widetilde{O}(d^2\tau)$ bit operations [16, §10.3.1]. Note that this algorithm requires d arithmetic operations performed with a precision in $\Omega(d\tau)$, and requires at least a quadratic number of bit operations. This method was also improved in practice for small degree polynomials [20].

In our case, the bound from Mahler on κ implies that the bit complexity of Algorithm 3 is also in $\widetilde{O}(d^2\tau)$, matching the bit-complexity of state-of-the art algorithms in the worst case for square-free polynomials with integer coefficients.

Main idea of the proof of Theorem 3. Our approach roughly follows the original approach of Schönhage [42, §20] in the sense that we will compute isolating disks centered on the roots of an approximation of f. It is also adaptive like more recent works ([33, 3, 26], among others), in the sense that we use a criterion for early termination.

The main novelty of our approach is that we start by computing an m-hyperbolic approximation of f, for a small initial constant integer m. This returns a set of $O(d/m)$ polynomials g of degree $O(m)$ defined on $O(d/m)$ disks covering the unit disk D. Then we compute the approximate roots of the g and we use a criterion to check if the corresponding approximate roots are the centers of disks isolating the roots of f. If we did not find all the roots of f, we double the parameter m and we start again. The criterion that we use to check if an approximate root is the center of an isolating disk is based on Kantorovich theory (recalled in Section 2.6) and it can be tested using the approximate polynomials of degree lower than m coming from the hyperbolic approximation. Moreover, this criterion will be satisfied for all roots of f for $m \geq c \log(\|f\|_1 d\kappa)$ for a universal constant c (Lemma 8). Further details can be found in Section 5.

1.2.3 Lower bound on condition number

Another insightful application of our data structure is a new geometrical interpretation of ill-conditioned polynomials, based on the distribution of their roots. Since the introduction of Wilkinson's polynomials $p(X) = (X-1)\cdots(X-d)$, it is known that the problem of polynomial root finding can be ill-conditioned even in the cases where the roots are well separated [48, 49]. More recently, it has been proved that the condition number of characteristic polynomials of $d \times d$ Gaussian matrices is in $2^{\Omega(d)}$ in average [9], where a Gaussian matrix is a matrix where the entries are independent, centered Gaussian random variables. Yet, no approaches provide a geometric explanation of this phenomenon. Our next theorem provides a geometric criterion that allows one to detect easily if a polynomial is ill-conditioned, based on the repartition of its roots. In these works, the authors consider the relative condition number defined as $\kappa^r(f) = \max_{f(\zeta)=0} \left(\frac{\|f\|_1}{|\zeta|}\kappa_\zeta\right)$.

Theorem 4. *Given a polynomial of degree d, let $N = \lceil \log_2(3ed) \rceil$ and let m be the maximal number of roots of f (resp. $X^d f(1/X)$) in a disk of the N-hyperbolic covering $\mathcal{H}_N$. The relative condition number $\kappa^r(f)$ of f is greater than $\frac{1}{4ed\sqrt{m}} 2^{5m/11}$.*

Remark 4. *In particular, the disk $D(0, 1/2)$ is covered by 4 disks in any N-hyperbolic covering. Thus, if m is the number of roots with absolute value less than $1/2$ or greater than 2, then $\kappa^r(f) \geq \frac{1}{8ed\sqrt{2m}} 2^{5m/88}$.*

As a direct consequence of this theorem, we recover the fact that the Wilkinson's polynomials have a condition number in $2^{\Omega(d)}$, since almost all their roots have a modulus larger than 2. Moreover, the set of eigenvalues of $d \times d$ Gaussian matrices are the Ginibre determinantal point process [19, 23], with eigenvalues roughly spread uniformly in the disk of radius $\sqrt{d}$ centered at 0, such that again, almost all the roots of the characteristic polynomial have a modulus greater than 2, which allows us to conclude that the expectation of the logarithm of its condition number is in $\Omega(d)$.

On the other hand, for a polynomial of degree d with random coefficients following a centered, Gaussian law of variance one, the expectation of the logarithm of the condition number of its real roots in $O(\log d)$ [13]. This is consistent with Theorem 4 since the roots of such polynomial, called Kac polynomials or hyperbolic polynomials, are roughly distributed evenly among the disks of an hyperbolic covering [14, 44, 38, 28, ...]. This means that our root solver algorithm is well-suited for polynomials with random coefficients of the same order of magnitude.

Main idea for the proof of Theorem 4. Given a polynomial f, we use Kantorovich theory to prove that for an integer m logarithmic in the condition number, the m-hyperbolic approximation returns polynomials of degree $m-1$ that have at least as many roots as f in the corresponding disk of the hyperbolic covering. Thus it implies that $m-1$ is greater than the number of roots of f, which provides a lower bound on a quantity logarithmic in the condition number.

1.2.4 Experimental proof of concept

Finally, we conclude with an experimental section, and we present a simple implementation of a root solver in the programming language `Python`, using the standard numerical library `NumPy` [21], and working with machine precision. Since our implementation is less than 100 lines of code, we include it in Appendix.

Our implementation is a simplified version of Algorithm 3. For the approximate factorization and the fast evaluation of the roots of unity, we use the standard polynomial root solver and the Fast Fourier Transform procedures of `NumPy`. For the data structure to detect duplicates, we simply round the solutions to a lower precision and sort the rounded solutions to detect the values with the same binary representation after rounding.

The current state-of-the-art implementation of a root solver for complex polynomials is the software `MPSolve` [6, 7], implemented in the `C` programming language, and based notably on the Aberth-Ehrlich method [15] . Its development started more than 20 years ago and it has received several improvements over time, making it the fastest current implementation to find all the complex roots of a polynomial. This software also uses multi-precision arithmetic when necessary. By contrast, our solver `HCRoots` is an early prototype written in `Python`, working in machine precision only, and depending solely on the `NumPy` library. Nevertheless, as we can see in Figure 2, for random polynomial that are known to be well-conditioned (see Section 1.2.3), our solver `HCRoots` called with a precision parameter $m = 30$ is an order of magnitude faster than `MPSolve`, which is very promising.

In our experiments, we focused on polynomials where the coefficients are centered, Gaussian random variables with variance 1. In this case, the solutions returned by our solver matched all the solutions returned by `MPSolve` with an error less than 2^{-25}, for polynomials up to degree 25000. Moreover, the linear complexity of our algorithm, combined with the fact that we don't need to use multi-precision arithmetic, allowed us to solve polynomials of degree 25000 an order of magnitude faster than `MPSolve`. In Figure 2, we show the timings to solve random polynomials of degree d with our solver, and `MPSolve`. Moreover, it is easy to parallelize our algorithm, and improve furthermore its practical efficiency.

2 Preliminaries

2.1 Notations

Given a polynomial or an analytic series f, we will denote by f' and f'' the derivative and the second derivative of f. If f is a polynomial, we will denote by $\|f\|_1$, $\|f\|_2$ and $\|f\|_\infty$ the classical norm 1, 2 and infinity on the vector of its coefficients. We will denote by $D(\gamma, \rho)$ the complex disk of radius ρ centered at γ. Finally, for a polynomial f and a root ζ of f, we let $\kappa_\zeta = \frac{\max(1,|\zeta|^d)}{|f'(\zeta)|}$ and $\kappa_1(f) = \max_{f(\zeta)=0}(\kappa_\zeta)$ is the absolute condition number of f. It is also denoted $\kappa(f)$ and referred as the condition number of f.

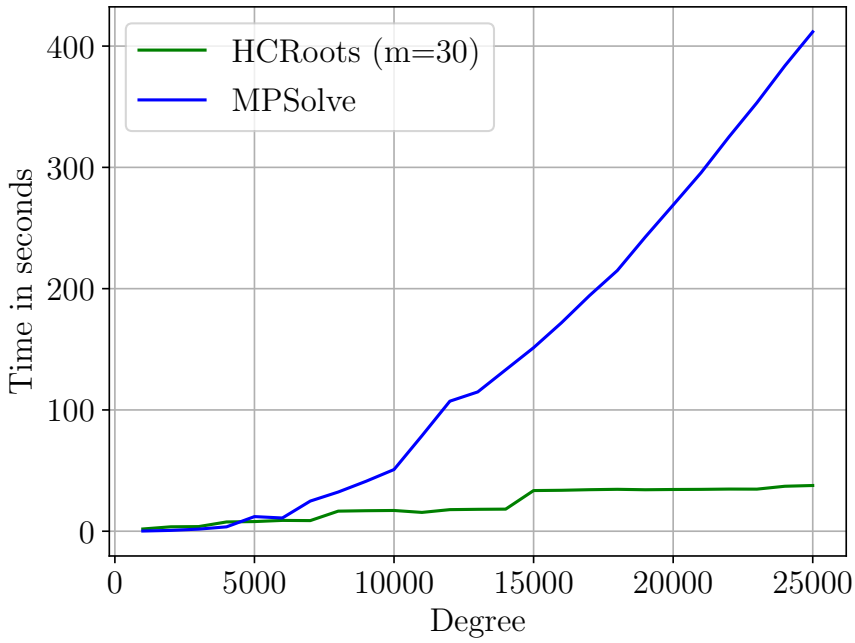

Figure 2: Time to approximate the roots of polynomials where the coefficients are random variable, centered Gaussian of variance 1

2.2 Fast elementary operations

We start with classical results on elementary operations, notably on the multiplication and the composition of polynomials.

Proposition 1 ([41, Theorem 2.2], [39, Proposition 3.2]). *Let f and g be two polynomials of degree d with $\|f\|_1$ and $\|g\|_1$ less than 2^τ, and an integer $m \geq \log(d+1)$. It is possible to compute the polynomial h such that $\|h - fg\|_1 \leq 2^{-m}$ in $\widetilde{O}(d(m+\tau))$ bit operations.*

Using Fast Fourier transform algorithm, we can also evaluate in a quasi-linear time a polynomial on the roots of unity. Note that in this case, even if the required precision m is smaller than d, the algorithm is still quasi-linear in dm.

Proposition 2 ([41, §3], [39, Proposition 3.3]). *Let f be a polynomial of degree d, and $\|f\|_1 \leq 2^\tau$ with $\tau \geq 1$, and an integer $m \geq \log(d+1)$. It is possible to compute the complex numbers $y_0, \ldots, y_{d-1}$ such that $\sum_{k=0}^{d-1} |y_k - f\left(e^{i2\pi k/d}\right)| \leq 2^{-m}$ in $\widetilde{O}(d(m+\tau))$ bit operations.*

Finally, another classical result that we will use is the fast composition of polynomials.

Proposition 3 ([39, Theorem 2.2]). *Let f and g be two polynomials of degree d with $\|f\|_1 \leq 2^\tau$ and $\|g\|_1 \leq 2^\nu$ where $\tau \geq 1$ and $\nu \geq 1$. Let m be a positive integer. It is possible to compute the polynomial h of degree $d-1$ such that $\|h(X) - f(g(X)) \mod X^d\|_1 \leq 2^{-m}$ in $\widetilde{O}(d(m+\tau+d\nu)))$ bit operations.*

Remark 5. *Ritzmann [39, Theorem 2.2] used the same bound for $\|f\|_1$ and $\|g\|_1$. Our proposition is a direct consequence of Ritzmann's theorem if we multiply f in the input by $2^{\nu-\tau}$, and the result by $2^{\tau-\nu}$. This reduction can be done in $\widetilde{O}(d(\tau+\nu+m))$ bit operations.*

2.3 Fast approximate multipoint evaluation

When we want to evaluate a polynomial on multiple points, it is possible to amortize the number of bit operations when the precision required is larger than the degree. In a recent work [45, §3.2], van der Hoeven showed that it can be done in $\widetilde{O}(d^{3/2}m^{3/2})$. However, this bound is not optimal when m is greater than d. For the case $m > d$, we recall here another state-of-the-art bound on the bit complexity for fast multipoint evaluation.

Proposition 4 ([45, Lemma 11], [30, Theorem 9])**.** *Let f be a polynomial of degree d, with $\|f\|_1 \leq 2^\tau$, with $\tau \geq 1$, and let $x_1, \ldots, x_d \in \mathbb{C}$ be complex points with absolute values bounded by 1. Then, computing y_k such that $|y_k - f(x_k)| \leq 2^{-m}$ for all k is possible in $\widetilde{O}(d(m + \tau + d))$ bit operations.*

Even though the bit complexity is quadratic in d, this approach is near optimal when m is greater than d, since its complexity matches the size of the output in this case. We reuse notably this result to bound the complexity of Algorithm 2, since our approach reduces the problem of evaluating a polynomial of degree d to the problem of evaluating several polynomials of degree m with a precision greater than m.

2.4 Condition number

Our root isolation algorithm has a bit complexity that depends on the condition number of the input polynomial. The absolute condition number is a measure of the displacement of its roots with respect to the displacement of its coefficients. More precisely, for a polynomial f with a vector of coefficients $c \in \mathbb{C}^{d+1}$, and a root ζ of f, there exists a neighborhood $U \subset \mathbb{C}^{d+1}$ of c, a neighborhood $V \subset \mathbb{C}$ of ζ and a differentiable function $\psi : U \to V$ that maps $c \in U$ to the unique zero in V of the corresponding polynomial. Letting $D\psi(\zeta)$ be the gradient of ψ at ζ, the condition number of ψ at ζ is the induced norm $|||D\psi(\zeta)|||_2 = \max_{\|\delta\|_2=1} |D\psi(\zeta) \cdot \delta| = \left(\sum_{k=0}^{d} |\zeta|^{2k}\right)^{1/2}$. If we consider the induced norm 1 instead, we have $|||D\psi(\zeta)|||_1 = \max_{\|\delta\|_1=1} |D\psi(\zeta) \cdot \delta| = \max_{k=0}^{d}(|\zeta|^k) = \max(1, |\zeta|^d)$.

Definition 3. *[8, §14.1.1] The* standard local absolute condition number *of polynomial f of degree d at a root ζ is $\kappa_2(f, \zeta) = \frac{1}{|f'(\zeta)|}\left(\sum_{k=0}^{d} |\zeta|^{2k}\right)^{1/2}$. Considering all the roots, we define the* standard absolute condition number *of f as $\kappa_2(f) = \max_{f(\zeta)=0} \kappa_2(f, \zeta)$.*

Remark 6. *The standard absolute condition number is obtained by considering the norm 2. If we consider the induced norm 1 instead, we get the condition number $\kappa_1(f, \zeta) = \frac{\max(1,|\zeta|^d)}{|f'(\zeta)|}$, such that $\kappa_1(f) \leq \kappa_2(f) \leq \sqrt{d}\kappa_1(f)$.*

As shown in Remark 6, the bound depending on the logarithm of the condition number for the norm 1 and the norm 2 will be the same up to a factor logarithmic in d. In the following, we will focus on the condition number κ_1 induced by the norm 1.

For square-free polynomial with integer coefficients, the condition number is finite. The following proposition bounds the condition number for square-free polynomials with integer coefficients.

Proposition 5 ([32, last inequality])**.** *Given a square-free polynomial f of degree d with integer coefficients, let τ be a real such that $\|f\|_1 \leq 2^\tau$. Then $\log(\kappa_1(f))$ is in $O(d\tau + d \log d)$.*

In particular, combined with Theorem 3, this proposition implies that for square-free polynomials, the bit-complexity of Algorithm 3 has the same worst-case bound as the state-of-the-art root-finding methods. We define also the relative condition number to represent the relative displacement of the roots, with respect to a relative displacement of the coefficients.

Definition 4 ([8, §14.1.1]). *The* standard local relative condition number *of a polynomial f at a root $\zeta \neq 0$ is $\kappa_2^r(f,\zeta) = \frac{\|f\|_2}{|\zeta|}\kappa_2(f,\zeta)$, and the* standard relative condition number *of f is $\kappa_2^r(f) = \max_{f(\zeta)=0,\zeta\neq 0} \kappa_2^r(f,\zeta)$. Similarly, we define $\kappa_1^r(f,\zeta) = \frac{\|f\|_1}{|\zeta|}\kappa_1(f,\zeta)$, and the associated* relative condition number *of f is $\kappa_1^r(f) = \max_{f(\zeta)=0,\zeta\neq 0} \kappa_1^r(f,\zeta)$.*

Remark 7. *An interesting property of the relative condition number at a nonzero root ζ of f is that it is equal to the relative condition number at the root $1/\zeta$ of the polynomial $g(X) = X^d f(1/X)$. Moreover for the absolute condition number we have $\kappa_1(g, 1/\zeta) \leq \kappa_1(f,\zeta)$ for ζ outside the unit disk.*

Proof of the remark. By construction, the inverse function is a bijection between the non-zero roots μ of g and the non-zero roots ζ of f. Computing the derivative of g at a root μ we have $\mu g'(\mu) = d\mu^d f(1/\mu) - \mu^{d-1} f'(1/\mu) = -\zeta f'(\zeta)/\zeta^d$. Thus $\kappa_1(g,\mu)/|\mu| = \frac{|\zeta|^d \max(1,|\mu|)^d}{|\zeta|\cdot|f'(\zeta)|} = \frac{\max(1,|\zeta|)^d}{|\zeta|\cdot|f'(\zeta)|} = \kappa_1(f,\zeta)/|\zeta|$. Finally, since $\|f\|_1 = \|g\|_1$ by construction, we have $\kappa_1^r(g,\mu) = \kappa_1^r(f,\zeta)$. □

This number is a standard way to measure to stability of the roots with respect to independent perturbations of the coefficients. In Theorem 4, we provide a geometric criterion to bound from below this condition number.

2.5 Fast approximate factorization

Another important result on univariate polynomials is a bound on the bit complexity to approximate all its roots. In the complex, approximating the roots is equivalent to compute an approximate factorization. We recall the state-of-the-art bound on the bit complexity for this problem.

Proposition 6 ([37, Theorem 2.1.1]). *Let f be a polynomial of degree d with leading coefficient c_d and all its roots ζ_k in the unit disk, and $m \geq d\log d$ a fixed real number. It is possible to compute complex numbers $z_1,\ldots,z_d$ such that $\|f(X) - c_d\prod_{k=1}^d (X - z_k)\|_1 \leq 2^{-m}\|f\|_1$ in $\widetilde{O}(dm)$ bit operations.*

Remark 8. *The theorem also holds with the same complexity for a polynomial h that has all its roots in the disk centered at the origin and of radius $c2^{m/d}$ for a constant $c \geq 1$, such as in the polynomials h in Algorithm 3 (Lemma 5).*

Proof of the remark. Let $f(Y) = h(c2^{m/d}Y)$. Then f has all its roots in the unit disk and $\|f\|_1 \leq c^d 2^m \|h\|_1$. Computing the approximate factorization $\widetilde{f}$ of f such that $\|f - \widetilde{f}\|_1 \leq 2^{-2m-d\log_2 c}\|f\|_1$ can be done in $\widetilde{O}(dm)$ since $m \geq d\log d$. Then with the change of variable $Y = X2^{-m/d}/c$, we have $\|h(X) - \widetilde{f}(X2^{-m/d}/c)\|_1 \leq 2^{-m}\|h\|_1$. □

This theorem does not directly give a bound on the distances between the roots. Schönhage shows [42, §19] that in the worst case $|\zeta_k - z_k| < 4\cdot 2^{-m/d}$. This bound can be improved for well-conditioned roots. In our case, we will also need a bound on the distance between some pairs of roots of two polynomials that have different degrees. We will use Kantorovich theory for the bounds in these cases (Section 2.6 and 5.1).

Note that Remark 8 requires a bound on the modulus of the roots of the polynomial. For that, we will use the classical Fujiwara bound.

Proposition 7 ([18]). *Let $f = \sum_{k=0}^d f_k X^k$ be a polynomial of degree d. Then the moduli of the roots of f are lower or equal to $2\max_{1\leq k\leq d} \sqrt[k]{\left|\frac{f_{d-k}}{f_d}\right|}$*

Applying this bound on the polynomial returned by an hyperbolic approximation, slightly perturbed, will allow us to verify that the assumptions of Remark 8 are satisfied (Lemma 5).

2.6 Certification of the roots

Several approaches in the literature guarantee that a neighborhood of a point contains a unique root of a given polynomial. We may cite notably Kantorovich criterion [12, §3.2], Smale's alpha theorem [12, §3.3], Newton interval method [36, Theorem 5.1.7], Pellet's test [33], Pellet's test combined with Graeffe iteration [3, 27], Cauchy's integral theorem [26, 25], and others [40], ... A first crude bound from the literature to guarantee that a disk centered at a point x contains a root of a polynomial f is the following.

Proposition 8 ([22, Theorem 6.4e], [6, Theorem 9])**.** *Let f be a polynomial of degree d, and x a complex point. Let $r_k = \sqrt[k]{k!\binom{d}{k}\left|\frac{f(x)}{f^{(k)}(x)}\right|}$. Then, for all $1 \leq k \leq d$, the disk $D(x, r_k)$ contains a root of f. In particular, the disk $D(x, d|f(x)/f'(x)|)$ contains a root of f.*

With some additional conditions, Kantorovich criterion provides a smaller radius to guarantee that a disk contains a root f.

Proposition 9 ([12, Theorem 88])**.** *Given a function f of class C^2, and a point x such that $f'(x) \neq 0$, let $\beta = |f(x)/f'(x)|$, and $K = \sup_{|y-x|\leq 2\beta} |f''(y)/f'(x)|$. If $2\beta K \leq 1$, then f has a root in the disk $D(x, 2|f(x)/f'(x)|)$.*

The previous propositions are useful to find a disk that contains a root of f, but they don't guarantee that the disk contains a unique root of f. In order to prove that Algorithm 3 terminates and to bound its complexity, we use the lower bound from Kantorovich theory on the size of the basin of attraction of the roots of f. More precisely, for each root ζ, we bound the size of a disk containing ζ where the Newton method always converges toward ζ.

Proposition 10 ([12, Theorem 85])**.** *Given a function f of class C^2, and a root ζ such that $f'(\zeta) \neq 0$. If there exists $r > 0$ such that $2rK \leq 1$, with $K = \sup_{|x-\zeta|\leq r} |f''(x)/f'(\zeta)|$, then ζ is the unique root of f in the disk $D(\zeta, r)$. Moreover, for any $x_0 \in D(\zeta, r)$, the Newton sequence defined by $x_{n+1} = x_n - f(x_n)/f'(x_n)$ converges toward ζ.*

Remark 9. *If f is a polynomial of degree d, and $s \geq \sup_{x\in D(0,1)} |f''(x)|$, then a consequence of Proposition 10 is that the set of disks $D(\zeta, 1/(2s\kappa_1(f)))$, for all roots ζ of f in the unit disk, are pairwise distinct.*

For any complex point x, we also prove the following lemma to have a criterion guaranteeing that a ball around x is included in a basin of attraction of a root of f.

Lemma 1. *Given a function of class C^2 and a point $x \in \mathbb{C}$ such that $f(x) \neq 0$. Let $r > 2|f(x)/f'(x)|$ and $K > |f''(y)/f'(x)|$ for all $y \in D(x, 4r)$. If $5rK \leq 1$ then, f has a unique root ζ in $D(x, r)$, and for all $x_0 \in D(x, r)$, the Newton sequence starting from x_0 converges to ζ.*

Proof. According to Proposition 9, f has a root ζ in $D(x, r)$. Then, $|f'(\zeta)| \geq |f'(x)| - |x - \zeta| \sup_{y\in D(x,r)} |f''(y)|$. This implies that $f'(\zeta) \geq |f'(x)|(1 - rK) \geq 4|f'(x)|/5$. Letting $\widetilde{K} = 5K/4$, we have that $\widetilde{K} \geq |f''(y)/f'(\zeta)|$ for all $y \in D(x, 4r) \supset D(\zeta, 2r)$. Remark that $4r\widetilde{K} \leq 1$, such that using Proposition 10, this implies that for all $x_0 \in D(\zeta, 2r) \supset D(x, r)$, the Newton sequence starting from x_0 converges toward ζ.

□

2.7 Geometric range searching

Our data structure can be seen as a piecewise polynomial approximation. As such, when doing multipoint evaluation, we will need to report the points that fall in a disk. This problem can be solved efficiently using classical range searching and point intersection searching algorithms.

Proposition 11 ([1, §5.2, Table 7]). *Given n points x_i in $\mathbb{C}$, it is possible to compute a data structure in $\widetilde{O}(n)$ operations such that for any disk D, returning the list of points x_i contained in D can be done in $O(k + \log n)$ operations, where k is the number of points in D.*

Moreover, when we isolate the roots of a polynomial f, we reduce the problem to isolate the roots in each disk of an N-hyperbolic covering. Because those disks overlap, we need to remove redundant boxes. For that, we use fast rectangle-rectangle searching techniques.

Proposition 12 ([1, §3.6]). *Given n rectangles r_i in the plane, it is possible to compute a data structure in $\widetilde{O}(n)$ such that for any rectangle r, returning the list of rectangle r_i intersecting r can be done in $\widetilde{O}(k + \log n)$ operations, where k is the number of rectangles intersecting r.*

Note that in an N-hyperbolic covering, each disk intersect at most 10 other disks of the covering. Indeed the disks centered in the disk $D(0, \frac{1}{2})$, it intersects at most 10 other disks. For $n \geq 1$ and a disk with a center between the circle of radius r_n and a circle of radius r_{n+1}, it intersects 2 disks of the hyperbolic covering that have their centers in the same ring. Then it can intersects at most 2 other disks coming from the inner adjacent ring, and 4 other from the outer adjacent ring. Thus, in Algorithm 3, this will guarantee that each query will be done in $\widetilde{O}(\log n)$ (see Section 5.2).

3 Computation of the hyperbolic approximation

3.1 Properties

First, given an integer m and a polynomial f of degree d, and a pair (g, a) from the hyperbolic approximation $H_{d,m}(f)$, we give a bound on the coefficients of the polynomials $f(a(X))$. That gives also a bound on the polynomial g, since g is an approximation of the polynomial $f(a(X))$ truncated to the degree $m - 1$.

Lemma 2. *Given a polynomial f of degree d and a integer $m > 1$, let a be an affine transformation appearing in the hyperbolic approximation $H_{d,m}(f)$. Letting $f(a(X)) = \sum_{k=0}^{d} c_k X^k$ and $\widetilde{m} = \min(m-1, d)$, we have $|c_k| \leq \begin{cases} \|f\|_1 \left(\frac{\widetilde{m}}{2k}\right)^k & \text{if } 0 \leq k \leq \widetilde{m} \\ \|f\|_1 / 2^k & \text{otherwise} \end{cases}$*

Proof. By construction, a is of the form $a(X) = (\gamma + \rho X)e^{i2\pi\alpha}$, with γ and ρ two positive real numbers. Letting $f(X) = \sum_{k=0}^{d} f_k X^k$, and expanding the $a(X)^k$, we have $|c_k| \leq \sum_{\ell=k}^{d} |f_\ell| \binom{\ell}{k} \gamma^{\ell-k} \rho^k$. We now distinguish two cases. First if $\gamma + \rho < 1$, then by construction of the sequences in Definition 1, we have $1 - \gamma = 2\rho$. Thus $|c_k| \leq \frac{1}{2^k} \sum_{\ell=k}^{d} |f_\ell| \binom{\ell}{k} \gamma^{\ell-k} (1-\gamma)^k \leq \|f\|_1 / 2^k$. This proves the desired bound for both the cases where $0 \leq k \leq \widetilde{m}$ and the case where $k \geq \widetilde{m}$.

Then, if $\gamma + \rho > 1$, then $\gamma = 1 - 1/2^N$ and $\rho = 3/2^{N+1}$, where $N = \lceil \log_2(3ed/\widetilde{m}) \rceil$. Then $|c_k| \leq \sum_{\ell=k}^{d} |f_\ell| \binom{\ell}{k} \gamma^{\ell-k} \rho^k \leq \|f\|_1 \rho^k \binom{d}{k}$. Using the inequalities $k! > (k/e)^k$ and $d \cdots (d-k+1) \leq d^k$ we have $|c_k| \leq \|f\|_1 (\rho e d/k)^k$. Moreover, $\rho \leq \widetilde{m}/(2de)$, such that $|c_k| \leq \|f\|_1 (\widetilde{m}/(2k))^k$. This also proves the desired bound for both the cases where $0 \leq k \leq \widetilde{m}$ and the case where $k \geq \widetilde{m}$. □

We also give a bound on the number of disks appearing in the decomposition.

Algorithm 1: Hyperbolic approximation data structure

Input: A polynomial $f(X) = \sum_{k=0}^{d} f_k X^k$ of degree d with $\|f\|_1 \leq 2^\tau$, $\tau \geq 1$, and an integer $m \geq 1$

Output: An m-hyperbolic approximation of f (see Definition 2)

$\widetilde{m} \leftarrow \min(m-1, d)$

$N \leftarrow \lceil \log_2(3ed/\widetilde{m}) \rceil$

for n *from* 0 *to* $N-1$ **do**

`# Compute` $(g_{n,k}, a_{n,k})$ `for the disks covering` $D(0, r_{n+1}) \setminus D(0, r_n)$

`# The precision of the arithmetic operations is in` $\Theta(m + \tau + \log d)$

`# A. Compute` r_n, γ_n, ρ_n `and` K_n `for the` $a_{n,k}(X) = (\gamma_n + \rho_n X) e^{i2\pi \frac{k}{K_n}}$

$r_n \leftarrow 1 - 1/2^n$

$r_{n+1} \leftarrow 1 - 1/2^{n+1}$ if $n \leq N-2$ else 1

$\gamma_n \leftarrow (r_n + r_{n+1})/2$

$\rho_n \leftarrow \frac{3}{4}(r_{n+1} - r_n)$

$K_n \leftarrow \lceil \frac{3\pi}{\sqrt{5}} \frac{r_{n+1}}{\rho_n} \rceil$

`# B. Compute` $g_{n,k}(X) \approx f\left((\gamma_n + \rho_n X) e^{i2\pi \frac{k}{K_n}}\right) \mod X^m$

`# B.1. Truncate` f `at` d_n `such that` $(\gamma_n + \rho_n)^{d_n+1} \leq 1/2^{m+1}$

$d_n \leftarrow \min\left(d, \lceil \frac{8}{3} \log(2)(m+1)2^n \rceil - 1\right)$ if $n < N-1$ else d

$p \leftarrow f_0 + \cdots + f_{d_n} X^{d_n}$

`# B.2. Gather the coefficients in` Y `of` $p(YZ) \mod Z^{K_n} - 1$`,`

`#  where` Y `and` Z `are symbolic variables`

for k *from* 0 *to* $K_n - 1$ **do**

  $p_k(Y^{K_n}) Y^k \leftarrow$ coefficients of Z^k of $p(YZ) \mod Z^{K_n} - 1$

`# B.3. Compute` $(\gamma_n + \rho_n X)^k \mod X^{\widetilde{m}}$

$q_0(X) \leftarrow 1$

for k *from* 1 *to* K_n **do**

  $q_k(X) \leftarrow q_{k-1}(X) \cdot (\gamma_n + \rho_n X) \mod X^{\widetilde{m}}$

`# B.4. Compute` $r_k(X) = p_k\left((\gamma_n + \rho_n X)^{K_n}\right) \cdot (\gamma_n + \rho_n X)^k \mod X^{\widetilde{m}}$

for k *from* 0 *to* $K_n - 1$ **do**

  $r_{k,0} + \cdots + r_{k,\widetilde{m}-1} X^{\widetilde{m}-1} \leftarrow p_k(q_{K_n}(X)) \cdot q_k(X) \mod X^{\widetilde{m}}$

`# B.5. Compute` $g_{n,k}(X) = r_0(X) + \cdots + r_{K_n-1}(X) e^{i2\pi \frac{k}{K_n}(K_n-1)}$

for ℓ *from* 0 *to* $\widetilde{m} - 1$ **do**

  $s_\ell(Z) \leftarrow r_{0,\ell} + \cdots + r_{K_n-1,\ell} Z^{K_n-1}$

  $g_{n,0,\ell}, \ldots, g_{n,K_n-1,\ell} \leftarrow s_\ell(e^{i2\pi \frac{0}{K_n}}), \ldots, s_\ell(e^{i2\pi \frac{K_n-1}{K_n}})$

`# B.6. Append the pair to the result list`

for k *from* 0 *to* $K_n - 1$ **do**

  $g_{n,k}(X) \leftarrow g_{n,k,0} + \cdots + g_{n,k,\widetilde{m}-1} X^{\widetilde{m}-1}$

  $a_{n,k}(X) \leftarrow (\gamma_n + \rho_n X) e^{i2\pi \frac{k}{K_n}}$

  Append the pair $(g_{n,k}, a_{n,k})$ to the list L

return L

Figure 3: Illustration for the proof that the union of the disks in a N-hyperbolic covering contains the unit disk (Lemma 3)

Lemma 3. *Given two integers d and $m > 1$, let $\widetilde{m} = \min(m-1, d)$ and let $N = \lceil \log_2(3ed/\widetilde{m}) \rceil$. Then the number of disks in the N-hyperbolic covering is in $O(d/\widetilde{m})$. Moreover, the union of the disks contains the unit disk.*

Proof. First, remark that the total number t of disks in a N-covering is $\sum_{n=0}^{N-1} K_n$. Such that using Remark 1 we have $K_n \leq 2^{n+4}$, and $t \leq 2^{N+4} \leq 16 \cdot 3ed/\widetilde{m}$. Thus the number of disks is in $O(d/\widetilde{m})$.

Then, we need to prove that for any ring $R_n = D(0, r_{n+1}) \setminus D(0, r_n)$, the union of the disks centered at $\gamma_n e^{i2\pi \frac{k}{K_n}}$ with radius ρ_n contains R_n. For that, let D be the disk $D(\gamma_n, \rho_n)$ and let $R_n(\alpha)$ be the segment intersection of R_n with the half-line starting from 0 and with angle α. Then, let β be the smallest angle such that $R_n(\beta)$ is not included in D. Then if the angle $\frac{2\pi}{K_n} \leq 2\beta$ it implies that for all $0 \leq \alpha < 2\pi$, the segment $R_n(\alpha)$ is included in a disk of the N-hyperbolic covering.

Since $\sin(\beta) \leq \beta$, it is sufficient to prove that $\frac{\pi}{K_n} \leq \sin(\beta)$. Consider the triangle ABC with $A = \gamma_n$, $B = r_{n+1}$ and $C = r_{n+1}e^{i\beta}$ (see Figure 3). Letting $O = 0$, remark that $\frac{\beta}{2}$ is the angle $\angle BOC$.

Let h be the distance between the point C and the line (OB). By construction, $\sin(\beta) = \frac{h}{r_{n+1}}$. If we prove that the angles at vertex A and vertex B are each smaller than $\frac{\pi}{2}$, then we can conclude that $h^2 \geq |C-A|^2 - |B-A|^2 = \rho_n^2 - (r_{n+1} - \gamma_n)^2 = \rho_n^2 - \frac{4}{9}\rho_n^2 = \frac{5}{9}\rho_n^2$. Such that $\sin(\beta) \geq \frac{\sqrt{5}}{3}\frac{\rho_n}{r_{n+1}} \geq \frac{\pi}{K_n}$.

In order to prove that the angles at A and B are smaller than $\frac{\pi}{2}$, remark that the triangle OBC is isosceles, such that the angle $\angle OBC = \angle ABC$ is less than $\pi/2$. Moreover, considering the triangle OAC, the angle $\angle OAC$ is larger than $\pi/2$ if $|C-O|^2 - |A-O|^2 - |C-A|^2 \geq 0$. This inequality holds for $n \geq 1$ since $|C-O|^2 - |A-O|^2 - |C-A|^2 = r_{n+1}^2 - \gamma_n^2 - \rho_n^2 = (r_{n+1} - \gamma_n)(r_{n+1} + \gamma_n) - \rho_n^2 \geq \frac{2}{3}\rho_n - \rho_n^2 \geq 0$. Thus $\angle BAC = \pi - \angle OAC$ is less than $\frac{\pi}{2}$. Since the angle of the triangle ABC at the vertices B and C are less than $\frac{\pi}{2}$, we can conclude that for all n, $\frac{\pi}{K_n}$ is small enough to let the disks cover the unit disk.

□

Finally, we give a bound on the bit size of the coefficients of the polynomials g of an hyperbolic approximation, by bounding the size of the polynomials $f(a(X))$, leading also to a bound the its second derivative.

Lemma 4. *Given a polynomial $f = \sum_{k=0}^{d} f_k X^k$ of degree d and a an affine transform from an m-hyperbolic approximation of f, let $\varphi(X) = f(a(X))$. Letting $\widetilde{m} = \min(m-1, d)$, we have $\|\varphi\|_1 \leq \|f\|_1 2^{\widetilde{m}/11}$ and for all $x \in D(0,1)$, we have $|\varphi''(x)| \leq \|f\|_1 \widetilde{m}^2 2^{\widetilde{m}/11}$.*

Proof. Let a be of the form $a(X) = (\gamma + \rho X)e^{i2\pi\alpha}$. Let $f^+(X) = \sum_{k=0}^{d} |f_k| X^k$. Since γ and ρ are positive numbers, we have $\|\varphi\|_1 \leq f^+(a(1)) \leq f^+(\gamma + \rho)$. By construction, $\gamma + \rho < 1 + \frac{1}{4}\frac{1}{2^{N-1}}$ with $N \geq \log_2(3ed/\widetilde{m})$. This implies that $\gamma + \rho \leq 1 + \frac{\widetilde{m}}{6ed}$ and thus $\|\varphi\|_1 \leq \|f\|_1 (1 + \frac{\widetilde{m}}{6ed})^d \leq \|f\|_1 e^{\frac{\widetilde{m}}{6e}} \leq \|f\|_1 2^{\widetilde{m}/11}$. Then, we obtain the bound on the derivative and second derivative of φ by bounding

the absolute value of the derivative and second derivative of each term $f_k a(X)^k$. By construction, either $\gamma + 2\rho \leq 1$ or $\rho = \frac{3}{4}\frac{1}{2^{N-1}}$ and $\gamma + \rho = 1 + \frac{1}{4}\frac{1}{2^{N-1}}$.

In the former case, using binomial inequalities, we have $|f_k|k(k-1)\rho^2(\gamma+\rho)^{k-2} \leq 2|f_k|(\gamma+2\rho)^k \leq 2|f_k|$ such that $|\varphi''(x)| \leq 2\|f\|_1$. In the second case, with $N \geq \log_2(3ed/\widetilde{m})$, we have $|f_k|k(k-1)\rho^2(\gamma+\rho)^{k-2} \leq |f_k|d(d-1)\rho^2(\gamma+\rho)^{d-2} \leq |f_k|d(d-1)\left(\frac{\widetilde{m}}{2ed}\right)^2\left(1+\frac{\widetilde{m}}{6ed}\right)^{d-2} \leq |f_k|\widetilde{m}^2/(4e^2)2^{\widetilde{m}/11}$, which implies $|\varphi''(x)| \leq \|f\|_1\widetilde{m}^2 2^{\widetilde{m}/11}$.

□

3.2 Proof of Theorem 1

We can now prove that Algorithm 1 returns an hyperbolic approximation of a polynomial with a complexity quasi-linear in the degree and the required precision.

Correctness. First, for the correctness of the algorithm, we will prove that Algorithm 1 returns a list of pairs (g, a) satisfying the constraints of Definition 2. First the affine transforms computed in Algorithm 1 send the unit disk to the disks described in Definition 1 of a hyperbolic covering. Then, the polynomials g computed are approximation of the polynomials $f(a(X)) \mod X^{\widetilde{m}}$. We will show that the approximation satisfies the bound $\|g(x) - f(a_{n,k}(X))\|_1 \leq 3\|f\|_1 2^{-m}$.

In part $B.1$ of Algorithm 1, we start by truncating f to d_n. The resulting polynomial p satisfies $f - p = f_{d_n+1}X^{d_n+1} + \cdots + f_d X^d$. Let $a_{n,k}$ be of the form $a_{n,k}(X) = (\gamma_n + \rho_n X)e^{i2\pi\frac{k}{K}n}$ and let $e^+(X) = \sum_{k=d_n+1}^{d} |f_k|X^k$. Since γ_n and ρ_n are positive numbers, we have $\|f(a_{n,k}(X)) - p(a_{n,k}(X))\|_1 \leq \|e^+(a_{n,k}(X))\|_1 \leq e^+(\gamma+\rho)$. In the case where $n < N-1$ and $d_n < d$, we have $\gamma_n + \rho_n = 1 - \frac{3}{8}\frac{1}{2^n} < 1$. Thus $\|e^+(a_{n,k}(X))\|_1 \leq (\gamma_n+\rho_n)^{d_n+1}\|f\|_1$. Moreover, $(\gamma_n + \rho_n)^{d_n+1} \leq e^{(d_n+1)\log(1-\frac{3}{8}\frac{1}{2^n})} \leq e^{-(d_n+1)\frac{3}{8}\frac{1}{2^n}} \leq \frac{1}{2^{m+1}}$ since $d_n \geq \frac{8}{3}\log(2)(m+1)2^n$. This implies that $\|f(a_{n,k}(X)) - p(a_{n,k}(X))\|_1 \leq \|f\|/2^{m+1}$.

Then, the algorithm will evaluate $p(YZ)$ on $Y = \gamma_n + \rho_n X$ and $Z = e^{i2\pi\frac{k}{K_n}}$, modulo $X^{\widetilde{m}}$ and modulo $Z^{K_n} - 1$. The advantage is that composition modulo $X^{\widetilde{m}}$ can be done efficiently using Proposition 3 and evaluation modulo $Z^{K_n} - 1$ on $Z = e^{i2\pi\frac{k}{K_n}}$ can be reduced to Fast Fourier Transform and be done efficiently too using Proposition 2. The part $B.2$ up to $B.5$ can perform this computation with an error less than $\|f\|_1/2^{m+1+\log_2(\widetilde{m})}$ on each coefficients. More precisely, in $B.3$ the algorithm computes $q_k(X) = (\gamma_n + \rho_n X)^k \mod X^{\widetilde{m}}$. Then in $B.4$ we have $r_k(X) = p_k((\gamma_n+\rho_n X)^{K_n}) \cdot (\gamma_n + \rho_n X)^k \mod X^{\widetilde{m}}$. Letting $\omega = e^{i2\pi/K_n}$, the truncated polynomial $g_{n,k}$ associated to the disk of center $\gamma_n\omega_k$ and radius ρ_n is $\sum_{j=0}^{K_n-1} r_j(X)\omega^{kj}$. In step $B.5$, for a fixed n and a fixed ℓ between 0 and $\widetilde{m}-1$, the coefficient of X^ℓ of $g_{n,k}$, denoted by $g_{n,k,\ell}$, is $s_\ell(\omega^k)$. Those coefficients can be computed efficiently using the fast Fourier transform algorithm. Such that using the notation of Algorithm 1 we finally have $\|g_{n,k}(X) - p(a_{n,k}(X)) \mod X^{\widetilde{m}}\|_1 \leq \|f\|_1/2^{m+1}$.

Finally, letting $p(a_{n,k}(X)) = \sum_{k=0}^{d} c_k X^k$, we have by Lemma 2 that $|c_k| \leq \|p\|_1/2^k$ for all $k \geq \widetilde{m}$. Thus, we have $\|p(a_{n,k}(X)) - [p(a_{n,k}(X)) \mod X^{\widetilde{m}}]\|_1 \leq 2\|p\|_1/2^{\widetilde{m}} \leq 2\|f\|_1/2^{\widetilde{m}}$.

Gathering the norm inequalities, we have as required:

$$\begin{aligned}
\|f(a_{n,k}(X)) - g(X)\|_1 &\leq \|f(a_{n,k}(x)) - p(a_{n,k}(X))\|_1 + \|p(a_{n,k}(X)) - [p(a_{n,k}(X)) \mod X^{\widetilde{m}}]\|_1 \\
&\quad + \|p(a_{n,k}(X)) - g(X) \mod X^{\widetilde{m}}\|_1 \\
&\leq \|f\|_1/2^{m+1} + \|f\|_1/2^{m+1} + 2\|f\|_1/2^m \\
&\leq 3\|f\|_1/2^m.
\end{aligned}$$

Complexity. The number of loop iterations in Algorithm 1 is in $O(\log d)$. Thus, it is sufficient to prove that each iteration can be performed in $\widetilde{O}(d(m+\tau))$ to achieve the complexity in Theorem 1. First, part A can be done in $\widetilde{O}(\log d)$ operations. Then in part B, we will use the state-of-the-art complexity bounds on the elementary operations recalled in Section 2.2.

First, in part $B.1$ and $B.2$, we are reordering the coefficients, gathering together the coefficients of X^k with the same value $k \mod K_n$, which can be done in $\widetilde{O}(d)$ bit operations. Note that the polynomials p_k computed in this part have a degree less than d_n/K_n, with $d_n/K_n \leq \frac{8}{3}\log(2)(m+1)$ and $d_n/K_n \leq d$, such that the degree of p_k is in $O(\widetilde{m})$.

Then in part $B.3$, we do K_n multiplication of polynomials of degree in $\widetilde{O}(\widetilde{m})$ with an absolute error on the result in $2^{\Theta(m)}$. Moreover, we have $\|q_k\|_1 \leq \max(1, (\gamma_n+\rho_n)^{K_n})$. Since $\gamma_n+\rho_n \leq 1+\frac{1}{2^{N+1}}$, this implies $\|q_k\|_1 \leq e^{K_n/2^{N+1}} \leq e^{2^{N+3}/2^{N+1}} \leq e^2$. Using Proposition 1, each multiplication can be done in $\widetilde{O}(\widetilde{m}m)$ bit operations, and part $B.3$ requires $\widetilde{O}(K_n\widetilde{m}m) = \widetilde{O}(dm)$ bit operations.

Similarly, in part $B.4$, since $\|p_k\|_1 \leq \|p\|_1 \leq \|f\|_1 \leq 2^\tau$ and $\|q_{K_n}\|_1 \leq e^2$, using proposition 3 on fast composition, we can compute $p_k(q_{K_n}) \mod X^{\widetilde{m}}$ with an error less than $2^{-\Theta(m)}$ in $\widetilde{O}(\widetilde{m}(m+\tau))$. We also perform the multiplication by q_k in this part within the same complexity. Overall, since the composition and the multiplication is done K_n times, part $B.4$ can be done in $\widetilde{O}(d(m+\tau))$ operations.

Finally, in part $B.5$, we use the Fast Fourier Transform algorithm to evaluate s of degree K_n on the roots of unity $e^{i2\pi\frac{k}{K_n}}$. If $\|s_\ell\|_1 \leq 2^\nu$ for an integer $\nu \geq 1$, the evaluation of s on the K_n points with an error less than 2^{-m} can be done in $\widetilde{O}(K_n(m+\nu))$ using Proposition 2. Thus the bit complexity for part $B.5$ is in $\widetilde{O}(d(m+\nu))$. To bound $\|s_\ell\|_1$, remark that $\sum_{\ell=0}^{\widetilde{m}-1}\|s_\ell\|_1 \leq \sum_{k=0}^{K_n-1}\|p_k(q_{K_n}))\cdot q_k\|_1$. Let p_k^+, p^+, f^+ be the polynomial p_k, p, f where we replaced the coefficients by their absolute value. In this case $\|p_k(q_{K_n}))\cdot q_k\|_1 \leq p_k^+((\gamma_n+\rho_n)^{K_n})(\gamma_n+\rho_n)^k$. Moreover $\sum_{k=0}^{K_n-1} p_k^+((\gamma_n+\rho_n)^{K_n})(\gamma_n+\rho_n)^k = p^+(\gamma_n+\rho_n) \leq f^+(\gamma_n+\rho_n)$. In turn $f^+(\gamma_n+\rho_n) \leq \|f\|_1(1+\frac{1}{2^{N+1}})^d$, and $N \geq \log_2(3ed/\widetilde{m})$, such that $(1+\frac{1}{2^{N+1}})^d \leq (1+\frac{\widetilde{m}}{6ed})^d \leq e^{\frac{\widetilde{m}}{6e}} \leq 2^{\widetilde{m}/11}$. Finally, $\sum_{\ell=0}^{\widetilde{m}-1}\|s_\ell\|_1 \leq \|f\|_1 2^{\widetilde{m}/11} \leq 2^{\tau+m}$. Thus part $B.5$ can be computed in $\widetilde{O}(d(m+\tau))$ bit operations.

4 Multipoint evaluation

A direct application of our data structure is the fast evaluation of polynomials. The main idea is to approximate the input polynomial f with a piecewise polynomial, where each polynomial g_k has a degree with the same order of magnitude as the required precision. Then we can use state-of-the-art multipoint evaluation technique on each g_k.

Proof of Theorem 2. The correction of Algorithm 2 is ensured by the fact that for all x in a disk $a_k(D(0,1))$, letting $z = a_k^{-1}(x)$, we have $|f(x) - g_k(a_k^{-1}(x))| = |f(a_k(z)) - g_k(z)| \leq \|f(a_k(X)) - g_k(X)\|_1 \leq 3\|f\|_1 2^{-m-2}$. If we compute y the evaluation of $g_k(z)$ with an error less than $\varepsilon = \|g_k\|_1 2^{-12m/11-2}$, the result will have an error less than $\|f\|_1 2^{-m-2}$, using the bound on $\|g_k\|_1$ given in Lemma 4. So finally we have $|f(x)-y| \leq 3\|f\|_1 2^{-m-2} + \|f\|_1 2^{-m-2} = \|f\|_1 2^{-m}$.

First the data structure Q can be computed in $\widetilde{O}(d)$ using Proposition 11, and the hyperbolic approximation G in $\widetilde{O}(dm)$ using Theorem 1. Then in the loop, the algorithm queries the points $v_1, \ldots, v_{n_k}$ in $\widetilde{O}(n_k + \log d)$ using Proposition 11. Let $\widetilde{m} = \min(m+1, d)$ be the degree of g_k, and $q_k = \lceil n_k/\widetilde{m} \rceil$. We can evaluate g_k on n_k points using q times the fast multipoint evaluation method in Proposition 4. For an absolute error less than $\|g_k\|_1 2^{-12m/11-2}$, this can be done in $\widetilde{O}(q_k\widetilde{m}m)$ bit operations. Note that $q_k\widetilde{m} \leq n_k + \widetilde{m}$, such that the total complexity in an iteration of the `for` loop is in $\widetilde{O}(n_k m + \widetilde{m}m + \log d)$. Note also that the sum of the n_k is d. If t is the number of discs in the hyperbolic approximation, after adding the complexity of all the main loop iterations, Algorithm 2

Algorithm 2: Multipoint evaluation

Input: Polynomial f of degree d, d complex number x_i in the unit disk, and a precision m
Output: List of complex number y_i such that $|y_i - f(x_i)| \leq \|f\|_1 2^{-m}$

$L \leftarrow \{\}$
$Q \leftarrow$ data structure adapted to the x_i for fast disk range searching
$G \leftarrow H_{d,m+2}(f)$
for (g_k, a_k) *in* G **do**
    `# The precision of the arithmetic operations is in` $\Theta(\tau + m)$
    $v_1, \ldots, v_{n_k} \leftarrow$ query Q for list of points x_i in $a_k(D(0,1))$
    $y_1, \ldots, y_{n_k} \leftarrow g_k(a_k^{-1}(v_1)), \ldots, g_k(a_k^{-1}(v_{n_k}))$
    Append $y_1, \ldots, y_{n_k}$ to L
return L

requires $\widetilde{O}(dm + t(\widetilde{m}m + \log d))$. By Lemma 3, t is in $O(d/\widetilde{m})$, such that the total complexity of Algorithm 2 is in $\widetilde{O}(dm)$. □

5 Root isolation

5.1 Properties of the approximate roots

We start by describing the properties satisfied by the roots of the truncated polynomials g coming from an m-hyperbolic approximations. In particular, we show how to perturb them such that all their roots are contained in a small enough disks.

Lemma 5. *Let $m > \widetilde{m}$ be two positive integers. Let $g(X) = \sum_{k=0}^{\widetilde{m}} c_k X^k$ be a polynomial of degree $\widetilde{m}$ and c be a constant such that $c_0 \leq c$ and $|c_k| \leq c\left(\frac{\widetilde{m}}{2k}\right)^k$ for $k \geq 1$. Then for the roots of the polynomial $g(X) + \frac{c}{2^m}X^{2\widetilde{m}}$ are in the disk $D(0, e2^{m/\widetilde{m}})$.*

Proof. Using the Fujiwara bound on the modulus of the roots of a polynomial (Proposition 7) on the polynomial $g(X) + \frac{1}{2^m}X^{2\widetilde{m}}$, we have $\sqrt[k]{\left|\frac{c_{2\widetilde{m}-k}}{c_{2\widetilde{m}}}\right|} = 0$ for $k < \widetilde{m}$, and for $\widetilde{m} \leq k < 2\widetilde{m}$ we have $\sqrt[k]{\left|\frac{c_{2\widetilde{m}-k}}{c_{2\widetilde{m}}}\right|} \leq 2^{(m-2\widetilde{m}+k)/k}\left(\frac{\widetilde{m}}{2\widetilde{m}-k}\right)^{(2\widetilde{m}-k)/k} \leq 2^{m/\widetilde{m}-1}e^{\frac{2\widetilde{m}-k}{k}\log\left(1+\frac{k-\widetilde{m}}{2\widetilde{m}-k}\right)} \leq 2^{m/\widetilde{m}-1}e^{(k-\widetilde{m})/k} \leq e2^{m/\widetilde{m}-1}$. Finally for $k = 2\widetilde{m}$, $\sqrt[2d]{\left|\frac{c_0}{c_{2\widetilde{m}}}\right|} \leq 2^{m/(2\widetilde{m})} \leq 2^{m/\widetilde{m}-1}$. □

Then we will use a technical lemma that gives a bound on the derivative of the difference of an analytic function and a polynomial, given bounds on their coefficients and the difference of their coefficients.

Lemma 6. *Let $\varphi(x) = \sum_{k=0}^{\infty}\varphi_k x^k$ be an analytic series with radius of convergence greater than 2. Let g be a polynomial of degree m and c be a positive real number such that: $||\varphi - g||_1 \leq c/2^m$ and $|\varphi_k| < c/2^k$ for all $k > m$. Then, for all x in the unit disk we have*

$$|\varphi'(x) - g'(x)| \leq c(m+2)/2^m.$$

Proof. Using the bounds on the coefficients of φ and g, we have $|g'(x) - \varphi'(x)| \leq \|g' - \varphi'\|_1 \leq mc/2^m + \sum_{k=m+1}^{\infty}(k-m)c/2^k$. The sum $S = \sum_{k=m+1}^{\infty} k/2^k$ can be bounded using the function

Algorithm 3: Root isolation

Input: Squarefree polynomial f of degree d

Output: List of d disks isolating all the roots of f in the unit disk and a subset of the other roots

$L \leftarrow \{\}$
$m \leftarrow 1$
while $|L| < d$ **do**
  $L \leftarrow \{\}$
  $G \leftarrow H_{d,m}(f)$
  $G^* \leftarrow \{(g, \frac{1}{a}) \mid (g, a) \in H_{d,m}(X^d f(1/X))\}$
  for (g, a) *in* $G \cup G^*$ **do**
    # Reduce the upper bound on the disk containing the roots of g (Lemma 5)
    $\widetilde{m} \leftarrow \min(m-1, d)$
    $h \leftarrow g(X) + \frac{\|f\|_1}{2^m} X^{2\widetilde{m}}$
    # Compute an approximation of the roots of h
    $\widetilde{h} \leftarrow$ Approximate factorization of h such that $\|h - \widetilde{h}\|_1 \leq 2^{-1.1m}\|h\|_1$
    for z_j *root of* $\widetilde{h}$ **do**
      # Check root unicity of $f(a(X))$ in a neighborhood of z_j (Lemma 1)
      $\varepsilon \leftarrow 3\|f\|_1(m+2)/2^m$ *# bound on $|f-g|$ and $|f'-g'|$ (Lemma 6)*
      if $|z_j| \leq 1$ *and* $|g'(z_j)| > \varepsilon$ **then**
        $K \leftarrow \frac{\|f\|_1 \widetilde{m}^2 2^{\widetilde{m}/11}}{|g'(z_j)| - \varepsilon}$
        $\beta \leftarrow \frac{|g(z_j)| + \varepsilon}{|g'(z_j)| - \varepsilon}$
        if $10\beta K \leq 1$ *and* $D(z_j, 8\beta) \subset D(0,1)$ **then**
          $L \leftarrow L \cup \{a(D(z_j, 2\beta))\}$
  # Remove duplicate roots
  $B \leftarrow$ list of bounding box of disks in L
  $Q \leftarrow$ data structure adapted to squares in B optimized for rectangle-rectangle search
  $L \leftarrow$ sublist of L without duplicates
  $m \leftarrow 2m$
return L

$\varphi(y) = \sum_{k=m+1}^{\infty} y^k = y^{m+1}/(1-y)$ defined for y a real in $[0,1[$. We have $S = 1/2\varphi'(1/2)$ and $\varphi'(y) = y^m[(m+1)(1-y)+y]/(1-y)^2$, such that $S = (m+1-m/2)/2^{m-1} = (m+2)/2^m$. Also we have $\sum_{k=m+1}^{\infty} m/2^k = m/2^m$. This leads to $|g'(x) - \varphi'(x)| \leq c(m+2)/2^m$ □

For an analytic function φ, this allows us to prove that if a polynomial g is a good enough approximation of φ, each root of φ in the unit disk is near a root of g.

Lemma 7. *Let $\varphi(x) = \sum_{k=0}^{\infty} \varphi_k x^k$ be an analytic series with radius of convergence greater than 2. Let g be a polynomial of degree m and c be a positive real number such that: $||\varphi - g||_1 \leq c/2^m$ and $|\varphi_k| < c/2^k$ for all $k > m$. Let ζ be a root of φ in the unit disk such that $\varphi'(\zeta) \neq 0$ and let $\kappa \geq 1/|\varphi'(\zeta)|$.*

If $2^m/(m+2) \geq 2c\kappa$, then g has a root in $D(\zeta, 2c\kappa m/2^m)$.

Remark 10. *If $m \geq 10$, the inequality $2^m/(m+2) \geq 2c\kappa$ holds as soon as $m \geq 2\log_2(c\kappa)$.*

Proof. Using Proposition 8, if $g'(\zeta) \neq 0$, then g has a root in the disk $D(\zeta, mg(\zeta)/g'(\zeta))$. Since ζ is in the unit disk, $|g(\zeta)| = |g(\zeta) - \varphi(\zeta)| \leq \|g - \varphi\|_1 \leq c/2^m$. For the derivative, we have $|g'(\zeta)| \geq |\varphi'(\zeta)| - |g'(\zeta) - \varphi'(\zeta)| \geq 1/\kappa - |g'(\zeta) - \varphi'(\zeta)|$. The difference between the derivative of g and φ can be bounded using Lemma 6 by $|g'(\zeta) - \varphi'(\zeta)| \leq c(m+2)/2^m$. Since $2^m/(m+2) \geq 2c\kappa$, this implies $|g'(\zeta)| \geq 1/(2\kappa)$, which allows us to conclude.

□

Finally, to prove that Algorithm 3 terminates, we will need the following lemma that guarantees that the criterion of Lemma 1 will be satisfied for a small enough approximation.

Lemma 8. *Let $\varphi(x) = \sum_{k=0}^{\infty} \varphi_k x^k$ be an analytic series with radius of convergence greater than 2 and ζ be a root of φ in the unit disk such that $\varphi'(\zeta) \neq 0$. Let $\kappa = 1/|\varphi'(\zeta)|$ and s be positive real greater than $|\varphi''(y)|$ for all y in the disk $D(0,1)$. Then, for any positive real $\varepsilon \leq 1/[23(s\kappa^2 + \kappa)]$ and all $x \in D(\zeta, \kappa\varepsilon)$:*

$$q := 10\frac{s(|\varphi(x)| + \varepsilon)}{(|\varphi'(x)| - \varepsilon)^2} < 1.$$

Proof. Using Taylor expansion at ζ, we have $|x - \zeta| \leq \kappa\varepsilon$, and thus $s(|\varphi(x)| + \varepsilon) \leq s(\kappa\varepsilon\varphi'(\zeta) + \frac{1}{2}\kappa^2\varepsilon^2 s + \varepsilon)$. Similarly, $|\varphi'(x)| - \varepsilon > |\varphi'(\zeta)| - \kappa\varepsilon s - \varepsilon$. Factoring out ε in the numerator, and $|\varphi'(\zeta)|$ in the denominator, this leads to $q \leq 10s\kappa^2\varepsilon\frac{2+\frac{1}{2}\varepsilon\kappa^2 s}{(1-\varepsilon(\kappa^2 s+\kappa))^2}$. Since $\varepsilon \leq 1/[23(s\kappa^2+\kappa)]$, the numerator is less than $93/46$ and the denominator is greater than $22^2/23^2$, such that $q \leq \frac{10}{23}\frac{93\cdot 23^2}{46\cdot 22^2} \leq 1$.

□

5.2 Proof of Theorem 3

We can now prove the main theorem bounding the bit complexity of Algorithm 3. We split our proof in three part. First the correctness, then the termination and finally a bound on the complexity of Algorithm 3.

Correctness. First, the correctness of Algorithm 3 follows from Lemma 1. Indeed, using Lemma 4 and Lemma 6, each disk added to L satisfies the condition of Lemma 1 and contains a unique root of f. Then, if the algorithm terminates, it returns a list of d pairwise distinct disks, containing a root of f each, such that the result is correct.

Termination. For the termination of Algorithm 3, we fix m and we will use Lemma 7 and 8 to show that for m sufficiently large, Algorithm 3 terminates. First, using Lemma 7 to bound the distance between a root of $\varphi := f(a(X))$ and the closest root of $\widetilde{h}$, we need a bound $\|\varphi - \widetilde{h}\|_1$ and a bound on the condition number of φ. The first bound comes from $\|\varphi - \widetilde{h}\|_1 \leq \|\varphi - h\|_1 + \|h - \widetilde{h}\|_1 \leq \|\varphi - h\|_1 + \|h\|_1/2^{1.1m} \leq \|\varphi - h\|_1(1 + 1/2^{1.1m}) + \|\varphi\|_1/2^{1.1m}$. Using Lemma 4, we have $\|\varphi\|_1 \leq \|f\|_1 2^{m/11}$, such that $\|\varphi - \widetilde{h}\|_1 \leq \|\varphi - h\|_1(1 + 1/2^{1.1m}) + \|f\|_1/2^m$. Moreover, $\|\varphi - h\|_1 \leq \|\varphi - g\|_1 + \|g - h\|_1 \leq 3\|f\|_1/2^m + \|f\|_1/2^m$ and $1 + 1/2^{1.1m} \leq 5/4$ for $m \geq 2$. This leads to $\|\varphi - \widetilde{h}\|_1 \leq 6\|f\|_1/2^m$. For the bound on the condition number, since a is of the form $a(X) = (\gamma + \rho X)e^{i\alpha}$, this implies that for all x in the unit disk $|\varphi'(x)| = \rho|f'(x)| \geq \min(1, \frac{m}{2ed})|f'(x)|$, such that $\kappa_1(\varphi) \leq \max(1, \frac{2ed}{m})\kappa_1(f)$. Letting $\kappa = 2ed\kappa_1(f)$, we have that for each root ζ_j of φ, if $m > 2\log_2(6\|f\|_1\kappa)$, Lemma 7 implies that there exists a root z_j of $\widetilde{h}$ in the disk $D(\zeta_j, \mu\kappa)$, where $\mu = 12\|f\|_1 m/2^m$.

We will now use this property with Lemma 8 to show that the criterion computed on line 16 of Algorithm 3 will eventually be satisfied. Using the notations of Algorithm 3, we show that the criterion $10\beta K \leq 1$ will be satisfied for all roots of f for m large enough. Let $s = \|f\|_1 d^2 2^{m/10}$. Using the bound on $|f - g|$ given by Definition 2 and the bound on $|f' - g'|$ given by Lemma 6, we have $10\beta K \leq 10\frac{s(|f(z_j)|+2\varepsilon)}{(|f'(z_j)|-2\varepsilon)^2}$. Moreover z_j is in the disk $D(\zeta_j, \mu\kappa)$, with $\mu \geq 6\|f\|_1(m + 2)/2^m = 2\varepsilon$. From Lemma 8, we can conclude that $10\beta K$ is smaller than 1 for $\mu \leq 1/[23(s\kappa^2 + \kappa)]$, that is for $12\|f\|_1 m/2^m \leq 1/[23(\|f\|_1 m^2 2^{m/10}\kappa^2 + \kappa)]$, which holds as soon as $12\|f\|_1 m^3/2^{\frac{9}{10}m} \leq 1/[23(\|f\|_1\kappa^2 + \kappa)]$. Note that for $m \geq 40$, $m/2^{\frac{9}{10}m}$ is smaller than $1/2^{m/2}$, such that $10\beta\kappa \leq 1$ for all $m > 2\log_2(276(\|f\|_1^2 d^2\kappa^2 + \kappa))$ and the algorithm terminates after $O(\log(\|f\|_1\kappa_1(f)))$ iterations of the main loop.

Complexity bound. First, at each iteration of the main `while` loop, computing the m-hyperbolic approximation costs $\widetilde{O}(dm)$ bit operations. Then for a fixed $m \leq 2d$, the approximate factorization is called $O(d/m)$ times on polynomials of degree m, with $\widetilde{O}(m^2)$ bit operations for each call, using Proposition 6. With Remark 8, this bound holds for polynomial that have all their roots of modulus less than $e2^{m/d}$. By Lemma 2, the coefficients of the polynomial h satisfy the condition of Lemma 5 and we conclude that all its roots of h are included in $D(0, e2^{m/d})$. Thus the approximate factorization can be computed within $\widetilde{O}(dm)$. Thus, for all cases, the total cost for the approximate factorization in an iteration of the `while` loop is in $\widetilde{O}(dm)$ bit operations. After that, for each g, we need to evaluate K and β up to a precision in $O(\log(\|f\|_1 d) + m)$. This can be done using state-of-the-art fast approximate multipoint evaluation in $\widetilde{O}(m(m + \log(\|f\|_1 d))$ for all the approximate roots of $\widetilde{h}$ using Proposition 4. This amounts to a total of $\widetilde{O}(d(m + \log(\|f\|_1 d))$ bit operations for the steps 7 to 17. If $m \geq d$, then the factorization is called a constant number of times on polynomials of degree d, for a total cost in $\widetilde{O}(dm)$ bit operations, and all the multipoint evaluations will cost a total of $\widetilde{O}(d(m + \log(\|f\|_1\kappa))$ bit operations. Finally, removing duplicate solutions can be done in $\widetilde{O}(d)$ operations. Indeed, by construction, given a box of B in a disk D of the N-hyperbolic covering, the number of times that it appears in L is bounded by the maximal number of disks of the N-hyperbolic covering that intersects D, that is 10 (see Section 2.7). Thus, using Proposition 12, this ensures that each query to detect a duplicate will cost at most $\widetilde{O}(\log d)$ operations, and removing all the duplicates will cost at most $\widetilde{O}(d)$ bit operations. In total, the costs is $\widetilde{O}(dm)$ per iteration of the `while` loop. Since m is doubled at each iteration, the cost is the same as the cost of the last iteration, that is in $\widetilde{O}(d\log(\|f\|_1\kappa_1(f)))$ bit operations.

6 Lower bound on the condition number

The lower bound on the condition number is a consequence of Lemma 7, that gives a bound on the distance between the roots of two polynomials with close enough coefficients, applied on the polynomials from the adapted hyperbolic approximation. Essentially, the idea is that for any m greater or equal to function of $\kappa_1(f)$ and for any pair (g, a) of an m-hyperbolic approximation of f, the number of roots of g is greater than the number of roots of $f(a(X))$ in the disk $a(D(0,1))$. In particular, this property allows us to deduce a lower bound on $\kappa_1(f)$ depending on the number of roots of $f(a(X))$ in the disk $a(D(0,1))$.

Proof of Theorem 4. Let D be a disk where the number m of solutions of f is maximal. Without restriction of generality, using Remark 3, we can assume that the absolute value of the center of D is less than 1. Up to a change of variable $f(uX)$, where u is a complex number of modulus 1, we can assume that the center of D is a positive real number. Letting $N = \lceil \log_2(3ed/m) \rceil$, we can see easily that D is included in a disk $D(\gamma, \rho)$ of the N-hyperbolic covering of the unit disk. Let $\varphi(X) = f(\gamma + \rho X)$ and let $g(X)$ be the polynomial of degree $m-1$ obtained by truncating φ at order $m-1$. By Lemma 2, the coefficients of φ for degree $\ell \geq m$ are less than $\|f\|_1/2^\ell$. Using the bounds in Lemma 4, we have for all x in the unit disk $\|\varphi''(x)\|_1 \leq \|f\|_1 m^2 2^{m/11}$. Moreover $\varphi' = \rho f'$, and $\rho \geq \frac{m}{4ed}$, such that $\kappa_1(\varphi) \leq \frac{4ed}{m}\kappa_1(f)$. Let $c = \|f\|_1$, $\kappa = \frac{4ed}{m}\kappa_1(f)$ and $s = \|f\|_1 m^2 2^{m/11}$. We can now prove by contradiction that the number of roots of g exceeds its degree if $2^m \geq \max(2c\kappa(m+2), 4cs\kappa^2 m)$. Indeed in this case, by Lemma 7, for each root ζ of φ, the polynomial g has a root in $D(\zeta, 2c\kappa m/2^m)$. Moreover, using Remark 9, for all the roots ζ of φ in the unit disk, the disks $D(\zeta, 2c\kappa m/2^m) \subset D(\zeta, 1/(2s\kappa))$ are pairwise distinct, such that g has at least $m > m-1$ roots. Thus, to avoid this contradiction, we thus have $\max(2c\kappa(m+2), 4cs\kappa^2 m) > 2^m$. That is, we must have either $2\|f\|_1\kappa_1(f)(m+2)4ed/m > 2^m$ or $\kappa_1(f)^2\|f\|_1^2(4ed)^2 2^{m/11} m > 2^m$. Equivalently, this amounts to $\|f\|_1\kappa_1(f) > \min\left(\frac{1}{8ed}\frac{m}{m+2}2^m, \frac{1}{4ed\sqrt{m}}2^{5m/11}\right)$. For $m \geq 3$, this implies $\|f\|_1\kappa_1(f) \geq \frac{1}{4ed\sqrt{m}}2^{5m/11}$. Finally, for each root ζ of φ with $|\zeta| < 1$ we have $\|f\|_1\kappa_1(f)/|\zeta| > \|f\|_1\kappa_1(f)$, which allows us to conclude. □

Acknowledgement

The author thanks the anonymous reviewers for their thoughtful comments on this work and on a previous related work.

A Source code

For the reproducibility of our experiments, and to demonstrate the conciseness of our implementation, we report here the full source code of our root solver `HCRoots`, along with an implementation of the multipoint evaluation algorithm (`hceval.py`), both available on a public gitlab server [35].

```
#    Copyright (C) 2021 Guillaume Moroz <guillaume.moroz@inria.fr>
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 2 of the License, or
# (at your option) any later version.
import numpy as np

# Compute the disks of a hyperbolic covering
def disks(d, m):
```

```
    N = np.math.ceil(np.log2(3*np.e*d/min(m-1,d)))
    r = 1 - 1/2**(np.arange(N+1))
    r[-1] = 1
    gamma = 1/2*(r[1:] + r[:-1])
    rho = 3/4*(r[1:] - r[:-1])
    K = np.ceil(3*np.pi*r[1:]/(np.sqrt(5)*rho)).astype(int)
    K[0] = 4
    return gamma, rho, K

# Compute the m-hyperbolic approximation
def hyperbolic_approximation(coeffs, m=30):
    d = coeffs.shape[-1]
    shape = coeffs.shape[:-1]
    gamma, rho, K = disks(d, m)
    N = gamma.size
    Kmax = ((d-1)//K.max()+1)*K.max()
    r = rho/gamma
    D = np.arange(d)
    G = np.zeros(shape + (N, Kmax, m), dtype='complex128')
    P = gamma[:, np.newaxis]**D * coeffs[..., np.newaxis, :]
    G[...,0] = np.fft.fft(P, Kmax)
    for i in range(m-1):
        P *= (D-i)/(i+1) * r[:, np.newaxis]
        G[..., i+1] = np.fft.fft(P[...,i+1:], Kmax)
    return G, gamma, rho, K

# Solve polynomials of small degree
def solve_small(p, m=30, guarantee=True, e=0):
    result = [np.empty(0)]*p.shape[0]
    abs_p = np.abs(p)
    nosol = abs_p[:,0] > abs_p[:,1:].sum(axis=-1)
    unksol = ~nosol
    sols = list(map(np.polynomial.polynomial.polyroots, p[unksol]))
    for i,j in enumerate(np.flatnonzero(unksol)):
        result[j] = sols[i][np.abs(sols[i])<=1]
    if guarantee:
        validate(result, p, e)
    return result

# Guarantee that there is a unique solution nearby
def validate(sols, p, e):
    nonempty = [i for i,x in enumerate(sols) if x.size>0]
    p0 = p[nonempty]
    p1 = np.polynomial.polynomial.polyder(p0, axis=-1)
    p2 = np.polynomial.polynomial.polyder(p1, axis=-1)
    s = np.linalg.norm(p2, 1, axis=-1)
    for i, j in enumerate(nonempty):
       q = 10*s[i]*(np.abs(np.polynomial.polynomial.polyval(sols[j], p0[i]))+e)/\
                   (np.abs(np.polynomial.polynomial.polyval(sols[j], p1[i]))-e)**2
       sols[j] = sols[j][q <= 1]


# Solve using truncated polynomials
def solve_piecewise(G, gamma, rho, K, m=30, rtol=8, guarantee=True, e=0):
    result = np.array([],dtype='complex128')
    Kmax = G.shape[1]
    for p, g, r, Kn in zip(G,gamma,rho,K):
        step = (Kmax-1)//(Kn-1)  # step * (Kn-1) < Kmax
        w = np.exp(-2j*np.pi*np.arange(0,Kmax,step)/Kmax)
```

```
        sols = solve_small(p[::step], m, guarantee, e)
        for i in range((Kmax-1)//step + 1):
            sols[i] = g*w[i] + r*sols[i]
            result = np.append(result, sols[i])
    rounded = np.round(result, decimals=-int(np.log10(rtol)))
    _, ind = np.unique(rounded, return_index=True)
    return result[ind]

# truncate and solve a polynomial over the complex
def solve(p, m=30, rtol=None, guarantee=True):
    rtol =  max(3*2**(-m), 2**-35) if rtol is None else rtol
    dtype = p.dtype if hasattr(p, 'dtype') else 'complex128'
    p = np.trim_zeros(p, 'b')
    coeffs = np.zeros((2, len(p)), dtype=dtype)
    coeffs[0] = p
    coeffs[1] = coeffs[0,::-1]
    G, gamma, rho, K = hyperbolic_approximation(coeffs, m)
    e = 3*np.linalg.norm(coeffs[0], 1)*(m+2)/2**m
    sols = solve_piecewise(G[0], gamma, rho, K, m, rtol, guarantee, e)
    invsols = solve_piecewise(G[1], gamma, rho, K, m, rtol, guarantee, e)
    result = np.concatenate([sols , 1/invsols])
    rounded = np.round(result, decimals=-int(np.log10(rtol)))
    _, ind = np.unique(rounded, return_index=True)
    return result[ind]
```

hcroots.py

```
#    Copyright (C) 2021 Guillaume Moroz <guillaume.moroz@inria.fr>
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 2 of the License, or
# (at your option) any later version.
import numpy as np

# Compute the disks of a hyperbolic covering
def disks(d, m):
    N = np.math.ceil(np.log2(3*np.e*d/min(m-1,d)))
    r = 1 - 1/2**(np.arange(N+1))
    r[-1] = 1
    gamma = 1/2*(r[1:] + r[:-1])
    rho = 3/4*(r[1:] - r[:-1])
    K = np.ceil(3*np.pi*r[1:]/(np.sqrt(5)*rho)).astype(int)
    K[0] = 4
    return gamma, rho, K

# Compute the m-hyperbolic approximation
def hyperbolic_approximation(coeffs, m=30):
    d = coeffs.shape[-1]
    shape = coeffs.shape[:-1]
    gamma, rho, K = disks(d, m)
    N = gamma.size
    Kmax = ((d-1)//K.max()+1)*K.max()
    r = rho/gamma
    D = np.arange(d)
    G = np.zeros(shape + (N, Kmax, m), dtype='complex128')
    P = gamma[:, np.newaxis]**D * coeffs[..., np.newaxis, :]
    G[...,0] = np.fft.fft(P, Kmax)
    for i in range(m-1):
        P *= (D-i)/(i+1) * r[:, np.newaxis]
```

```
        G[..., i+1] = np.fft.fft(P[...,i+1:], Kmax)
    return G, gamma, rho, K

# Get the indices to match points to the corresponding disk
def get_indices(N, Kmax, points):
    module_indices = np.zeros(points.shape, int)
    angle_indices = np.zeros(points.shape, int)
    apoints = np.abs(points)
    big = apoints > 1-1/2**(N-1)
    small = apoints < 1/2
    middle = ~small & ~big
    module_indices[middle] = np.log2(1/(1-apoints[middle])).astype(int)
    module_indices[big] = N-1
    angle_indices[:] = (0.5 - np.angle(points)*(Kmax/(2*np.pi)) % Kmax).astype(int)
    return module_indices, angle_indices

# Evaluate the points in a unit disk
def eval_unitdisk(G, gamma, rho, points):
    N, Kmax, m = G.shape
    m_ind, a_ind = get_indices(N, Kmax, points)
    shift_points = (points - gamma[m_ind]*np.exp(-2j*np.pi*a_ind/Kmax))/rho[m_ind]
    res = np.polynomial.polynomial.polyval( shift_points, G[m_ind, a_ind].T, tensor=False)
    return res

# Evaluate the points in the complex plane
def eval_hyperbolic_approximation(covering, points):
    G, gamma, rho, d = covering
    points = np.array(points)
    apoints = np.abs(points)
    inpoints = points[apoints <= 1]
    outpoints = points[apoints > 1]
    res = np.zeros(points.size, dtype='complex128')
    res[apoints <= 1] = eval_unitdisk(G[0], gamma, rho, inpoints)
    res[apoints >  1] = eval_unitdisk(G[1], gamma, rho, 1/outpoints)*outpoints**d
    return res

# Compute the hyperbolic approximation used for multi-point evaluation
def get_hyperbolic_approximation(p, m=30):
    d = len(p)-1
    dtype = p.dtype if hasattr(p, 'dtype') else 'complex128'
    coeffs = np.zeros((2, d+1), dtype=dtype)
    coeffs[0] = p
    coeffs[1] = coeffs[0,::-1]
    G, gamma, rho, K = hyperbolic_approximation(coeffs, m)
    covering = G, gamma, rho, d
    return covering

# Compute the hyperbolic approximation and evaluate the points
def eval(p, points, m=30):
    covering = get_hyperbolic_approximation(p, m)
    res = eval_hyperbolic_approximation(covering, points)
    return res
```

hceval.py

References

[1] Pankaj K Agarwal and Jeff Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in discrete and computational geometry*, volume 223 of *Contemporary Mathematics*, pages 1 – 56. American Mathematical Society, Providence, RI, 1999.

[2] Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1974.

[3] 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.

[4] Carlos Beltrán and Luis Miguel Pardo. Fast linear homotopy to find approximate zeros of polynomial systems. *Foundations of Computational Mathematics*, 11(1):95–129, Feb 2011.

[5] Todor Bilarev, Magnus Aspenberg, and Dierk Schleicher. On the speed of convergence of newton's method for complex polynomials. *Mathematics of Computation*, 85(298):693–705, 2016.

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

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

[8] Peter Bürgisser and Felipe Cucker. *Condition: The Geometry of Numerical Algorithms*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[9] Peter Bürgisser, Felipe Cucker, and Elisa Rocha Cardozo. On the condition of the zeros of characteristic polynomials. *Journal of Complexity*, 42:72–84, 2017.

[10] Felipe Cucker, Teresa Krick, Gregorio Malajovich, and Mario Wschebor. A numerical algorithm for zero counting, i: Complexity and accuracy. *Journal of Complexity*, 24(5):582–605, 2008.

[11] Felipe Cucker and Steve Smale. Complexity estimates depending on condition and round-off error. *J. ACM*, 46(1):113–184, January 1999.

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

[13] Yen Do, Hoi Nguyen, and Van Vu. Real roots of random polynomials: expectation and repulsion. *Proceedings of the London Mathematical Society*, 111(6):1231–1260, 2015.

[14] Alan Edelman and Eric Kostlan. How many zeros of a random polynomial are real? *Bulletin of the American Mathematical Society*, 32(1):1–37, 1995.

[15] Louis W. Ehrlich. A modified newton method for polynomials. *Commun. ACM*, 10(2):107–108, February 1967.

[16] Ioannis Z. Emiris, Victor Y. Pan, and Elias Tsigaridas. Algebraic algorithms. Chapter 10 of Computing Handbook , Volume I: Computer Science and Software Engineering (Allen B. Tucker, Teo Gonzales, and Jorge L. Diaz-Herrera, editors), 2014.

[17] Charles M. Fiduccia. Polynomial evaluation via the division algorithm the fast fourier transform revisited. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, page 88–93, New York, NY, USA, 1972. Association for Computing Machinery.

[18] Matsusaburô Fujiwara. Über die obere schranke des absoluten betrages der wurzeln einer algebraischen gleichung. *Tohoku Mathematical Journal, First Series*, 10:167–171, 1916.

[19] Jean Ginibre. Statistical ensembles of complex, quaternion, and real matrices. *Journal of Mathematical Physics*, 6(3):440–449, 1965.

[20] Xavier Gourdon. Algorithmique du theoreme fondamental de l'algebre. Research Report RR-1852, INRIA, 1993. HAL: https://hal.inria.fr/inria-00074820.

[21] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

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

[23] J. Ben Hough, Majunath Krishnapur, Yuval Peres, and Bálint Virág. Determinantal processes and independence. *Probability Surveys*, 3:206–229 (electronic), 2006. 00000.

[24] John Hubbard, Dierk Schleicher, and Scott Sutherland. How to find all roots of complex polynomials by newton's method. *Inventiones mathematicae*, 146(1):1–33, Oct 2001.

[25] Rémi Imbach and Victor Y. Pan. New practical advances in polynomial root clustering. In Daniel Slamanig, Elias Tsigaridas, and Zafeirakis Zafeirakopoulos, editors, *Mathematical Aspects of Computer and Information Sciences*, pages 122–137, Cham, 2020. Springer International Publishing.

[26] Rémi Imbach and Victor Y. Pan. New progress in univariate polynomial root finding. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, ISSAC '20, page 249–256, New York, NY, USA, 2020. Association for Computing Machinery.

[27] Rémi Imbach, Victor Y. Pan, and Chee Yap. Implementation of a near-optimal complex root clustering algorithm. In James H. Davenport, Manuel Kauers, George Labahn, and Josef Urban, editors, *Mathematical Software – ICMS 2018*, pages 235–244, Cham, 2018. Springer International Publishing.

[28] Zakhar Kabluchko and Dmitry Zaporozhets. Asymptotic distribution of complex zeros of random analytic functions. *The Annals of Probability*, 42(4):1374 – 1395, 2014.

[29] R. Baker Kearfott. *Rigorous global search: continuous problems.* Nonconvex optimization and its applications. Kluwer Academic Publishers, Dordrecht, Boston, 1996.

[30] Alexander Kobel and Michael Sagraloff. Fast approximate polynomial multipoint evaluation and applications, 2016. arXiv:https://arxiv.org/abs/1304.8069v2 [cs.NA].

[31] Pierre Lairez. A deterministic algorithm to compute approximate roots of polynomial systems in polynomial average time. *Foundations of Computational Mathematics*, 17(5):1265–1292, Oct 2017.

[32] Kurt Mahler. An inequality for the discriminant of a polynomial. *Michigan Mathematical Journal*, 11(3):257 – 262, 1964.

[33] 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.

[34] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to interval analysis*. Siam, 2009.

[35] Guillaume Moroz. HCRoots: Hyperbolic Complex Root solver. `https://gitlab.inria.fr/gmoro/hcroots`, 2021.

[36] Arnold Neumaier. *Interval methods for systems of equations*. Cambridge University Press, 1990.

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

[38] Yuval Peres and Bálint Virág. Zeros of the i.i.d. Gaussian power series: a conformally invariant determinantal process. *Acta Mathematica*, 194(1):1 – 35, 2005.

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

[40] Siegfried M. Rump. Ten methods to bound multiple roots of polynomials. *Journal of Computational and Applied Mathematics*, 156(2):403 – 432, 2003.

[41] Arnold Schönhage. Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients. In Jacques Calmet, editor, *Computer Algebra*, pages 3–15, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[42] Arnold Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Departement of Mathematics, University of Tübingen, Germany, 1982. updated 2004.

[43] Steve Smale. The fundamental theorem of algebra and complexity theory. *Bull. Amer. Math. Soc. (N.S.)*, 4(1):1–36, 01 1981.

[44] Mikhail Sodin and Boris Tsirelson. Random complex zeroes, i. asymptotic normality. *Israel Journal of Mathematics*, 144(1):125–149, Mar 2004.

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

[46] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, U.K., 3 edition, 2013.

[47] Rephael Wenger. *Isosurfaces: geometry, topology, and algorithms*. CRC Press, 2013.

[48] James H. Wilkinson. The evaluation of the zeros of ill-conditioned polynomials. part ii. *Numerische Mathematik*, 1(1):167–180, Dec 1959.

[49] James H. Wilkinson. *Rounding errors in algebraic processes*. Englewood Cliffs, N.J: Prentice-Hall, 1964.