Computing Real Roots of Real Polynomials

Michael Sagraloff* Kurt Mehlhorn*

March 12, 2015

Abstract

Computing the roots of a univariate polynomial is a fundamental and long-studied problem of computational algebra with applications in mathematics, engineering, computer science, and the natural sciences. For isolating as well as for approximating all complex roots, the best algorithm known is based on an almost optimal method for approximate polynomial factorization, introduced by Pan in 2002. Pan's factorization algorithm goes back to the splitting circle method from Schönhage in 1982. The main drawbacks of Pan's method are that it is quite involved[1] and that all roots have to be computed at the same time. For the important special case, where only the real roots have to be computed, much simpler methods are used in practice; however, they considerably lag behind Pan's method with respect to complexity.

In this paper, we resolve this discrepancy by introducing a hybrid of the Descartes method and Newton iteration, denoted ANewDsc, which is simpler than Pan's method, but achieves a run-time comparable to it. Our algorithm computes isolating intervals for the real roots of any real square-free polynomial, given by an oracle that provides arbitrary good approximations of the polynomial's coefficients. ANewDsc can also be used to only isolate the roots in a given interval and to refine the isolating intervals to an arbitrary small size; it achieves near optimal complexity for the latter task.

Keywords. root finding, root isolation, root refinement, approximate arithmetic, certified computation, complexity analysis

1 Introduction

Computing the roots of a univariate polynomial is a fundamental problem in computational algebra. Many problems from mathematics, engineering, computer science, and the natural sciences can be reduced to solving a system of polynomial equations, which in turn reduces to solving a polynomial equation in one variable by means of elimination techniques such as resultants or Gröbner Bases. Hence, it is not surprising that this problem has been extensively studied and that numerous approaches have been developed; see [23, 24, 25, 32] for an extensive (historical) treatment. Finding all roots of a polynomial and the approximate factorization of a polynomial into linear factors are closely related. The most efficient algorithm for approximate factorization is due to Pan [30]; it is based on the splitting circle method of Schönhage [41] and considerably refines it. From an approximate factorization, one can derive arbitrary good approximations of

*MPI for Informatics, Campus E1 4, 66123 Saarbrücken, Germany. emails: msagralo@mpi-inf.mpg.de and mehlhorn@mpi-inf.mpg.de. The author ordering deviates from the default alphabetic ordering used in Theoretical Computer Science, because the first author contributed significantly more to the paper than the second author.

[1] In Victor Pan's own words: "Our algorithms are quite involved, and their implementation would require a non-trivial work, incorporating numerous known implementation techniques and tricks". In fact, we are not aware of any implementation of Pan's method.

all complex roots as well as corresponding isolating disks; e.g. see [12, 26]. The main drawbacks of Pan's algorithm are that it is quite involved (see Footnote 1) and that it necessarily computes all roots, i.e., cannot be used to only isolate the roots in a given region. It has not yet been implemented. In contrast, simpler approaches, namely Aberth's, Weierstrass-Durand-Kerner's and QR algorithms, found their way into popular packages such as MPSolve [4] or `eigensolve` [13], although their excellent empirical behavior has never been entirely verified in theory.

In parallel, there is steady ongoing research on the development of dedicated real roots solvers that also allow to search for the roots only in a given interval. Several methods (e.g. Sturm method, Descartes method, continued fraction method, Bolzano method) have been proposed, and there exist many corresponding implementations in computer algebra systems. With respect to computational complexity, all of these methods lag considerably behind the splitting circle approach. *In this paper, we resolve this discrepancy by introducing a hybrid of the Descartes method and Newton iteration, denoted* ANewDsc *(read approximate-arithmetic-Newton-Descartes). Our algorithm is simpler than Pan's algorithm, is already implemented with very promising results for polynomials with integer coefficients [20], and has a complexity comparable to that of Pan's method.*

1.1 Algorithm and Results

Before discussing the related work in more detail, we first outline our algorithm and provide the main results. Given a square-free univariate polynomial P with real coefficients, the goal is to compute disjoint intervals on the real line such that all real roots are contained in the union of the intervals and each interval contains exactly one real root. The Descartes or Vincent-Collins-Akritas[2] method is a simple and popular algorithm for real root isolation. It starts with an open interval guaranteed to contain all real roots and repeatedly subdivides the interval into two open intervals and a split point. The split point is a root if and only if the polynomial evaluates to zero at the split point. For any interval I, Descartes' rule of signs (see Section 2.3) allows one to compute an integer v_I, which bounds the number m_I of real roots in I and is equal to m_I, if $v_I \leq 1$. The method discards intervals I with $v_I = 0$, outputs intervals I with $v_I = 1$ as isolating intervals for the unique real root contained in them, and splits intervals I with $v_I \geq 2$ further. The procedure is guaranteed to terminate for square-free polynomials, as $v_I = 0$, if the circumcircle of I (= the one-circle region of I) contains no root of p, and $v_I = 1$, if the union of the circumcircles of the two equilateral triangles with side I (the two-circle region of I) contains exactly one root of I, see Figure 1 on page 11.

The advantages of the Descartes method are its simplicity and the fact that it applies to polynomials with real coefficients. The latter has to be taken with a grain of salt. The method uses the four basic arithmetic operations (requiring only divisions by two) and the sign-test for numbers in the field of coefficients. In particular, if the input polynomial has integer or rational coefficients, the computation stays within the rational numbers. Signs of rational numbers are readily determined. In the presence of non-rational coefficients, the sign-test becomes problematic.

The disadvantages of the Descartes method are its inefficiency when roots are clustered and its need for exact arithmetic. When roots are clustered, there can be many subsequent subdivision steps, say splitting I into I' and I'', where $\min(v_{I'}, v_{I''}) = 0$ and $\max(v_{I'}, v_{I''}) = v_I$. Such subdivision steps exhibit only linear convergence to the cluster of roots as an interval I is split into equally sized intervals. The need for exact arithmetic stems from the fact that it is crucial for the correctness of the algorithm that sign-tests are carried out exactly. It is known how to overcome each one of the two weaknesses separately; however, it is not known how to overcome them simultaneously. Our main result achieves this. That is, we present an algorithm ANewDsc

[2]Descartes did not formulate an algorithm for isolating the real roots of a polynomial but (only) a method for bounding the number of positive real roots of a univariate polynomial (Descartes' rule of signs). Collins and Akritas [7] based on ideas going back to Vincent formulated a bisection algorithm based on Descartes' rule of signs.

that overcomes both shortcomings at the same time. Our algorithm applies to real polynomials given through coefficient oracles[3], and our algorithm works well in the presence of clustered roots.

We are now ready for a high-level description of the algorithm. The correctness of the Descartes method rests on exact sign computations; however, the exact computation of the sign of a number does not necessarily require the exact computation of the number. The Descartes algorithm uses the sign-test in two situations: It needs to determine whether the polynomial evaluates to zero at the split point, and it determines v_I as the number of sign changes in the coefficient sequence of polynomials P_I, where P_I is a polynomial determined by the interval I and the input polynomial P. We borrow from [10] the idea of carefully choosing split points so as to guarantee that P is relatively large at split points. Our realization of the idea is, however, quite different and is based on a fast method for approximately evaluating a polynomial at many points [19, 21, 22, 34, 44]. We describe the details in Section 2.2. The choice of a split point where the polynomial has a large absolute value has a nice consequence for the sign change computation. Namely, the cases $v_I = 0$, $v_I = 1$, and $v_I > 1$ can be distinguished with approximate arithmetic. We give more details in Section 2.4.

The recursion tree of the Descartes method may have size $\Omega(n\tau)$. However, there are only $O(n)$ nodes where both children are subdivided further. This holds because the sum of the number of sign changes at the children of a node is at most the number of sign changes at the node. In other words, large subdivision trees must have long chains of nodes, where the interval that is split off is immediately discarded. Long chains are an indication of clustered roots. We borrow from [36] the idea of traversing such chains more efficiently by combining Descartes' rule of signs, Newton iteration, and a subdivision strategy similar to Abbott's quadratic interval refinement (QIR for short) method [1]. As a consequence, quadratic convergence towards the real roots instead of the linear convergence of pure bisection is achieved in most iterations. Again, our realization of the idea is quite different. For example, Newton iteration towards a cluster of k roots needs to know k (see equation 22). The algorithm in [36] uses exact arithmetic and, therefore, can determine the exact number v_I of sign changes in the coefficient sequence of polynomials P_I; v_I is used as an estimate for the size of a cluster contained in interval I. We cannot compute v_I (in the case $v_I > 1$, we only know this fact and have no further knowledge about the value of v_I) and hence have to estimate the size of a cluster differently. We tentatively perform Newton steps from both endpoints of the interval and determine a k such that both attempts essentially lead to the same iterate. Also, we use a variant of quadratic interval refinement because it interpolates nicely between linear and quadratic convergence. Details are given in Section 3. This completes the high-level description of the algorithm.

A variant of our algorithm using randomization instead of multi-point evaluation for the choice of split point has already been implemented for polynomials with integer coefficients [20]. First experiments are quite promising. The implementation seems to be competitive with the fastest existing real root solvers for all instances and much superior for some hard instances with clustered roots, e.g., Mignotte polynomials.[4]

The worst-case bit complexity of our algorithm essentially matches the bit complexity of the algorithms that are based on Pan's algorithm. More specifically, we prove the following theorems:

Theorem 1. *Let* $P(x) = P_n x^n + \ldots + P_1 x^1 + P_0 \in \mathbb{R}[x]$ *be a real, square-free polynomial of degree* n *with*[5] $1/4 \le P_n \le 1$. *The algorithm* ANewDsc *determines isolating intervals for all*

[3] A coefficient oracle provides arbitrarily good approximations of the coefficients.

[4] Mignotte [27][Section 4.6] considers the polynomial $x^n - 2(ax-1)^2$, where $n \ge 3$ and $a \ge 10$ is an integer. He shows that the polynomial has two real roots in the interval $(1/a - h, 1/a + h)$, where $h = a^{-(n+2)/2}$.

[5] If $P(x)$ is arbitrary, it suffices to determine an integer t with $2^t/4 \le P_n \le 2^t$ and to consider $2^{-t}P(x)$.

real roots of P with a number of bit operations bounded by[6]

$$\tilde{O}(n \cdot (n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1}))) \tag{1}$$
$$= \tilde{O}(n(n^2 + n \log \mathrm{Mea}(P) + \log M(\mathrm{Disc}(P)^{-1}))).$$

The coefficients of P have to be approximated with quality[7]

$$\tilde{O}(n + \tau_P + \max_i(n \log M(z_i) + \log M(P'(z_i)^{-1}))).$$

Here $M(x) := \max(1, |x|)$, z_1 to z_n are the roots of P, $\mathrm{Mea}(P) := |P_n| \cdot \prod_{i=1}^{n} M(|z_i|)$ *denotes the* Mahler Measure *of P,* $\mathrm{Disc}(P) := P_n^{2n-2} \prod_{1 \le i < j \le n} (z_j - z_i)^2$ *is the* discriminant *of P, $\tau_P := M(|\log(\max_i |P_i|)|)$, and P' is the derivative of P.*

For polynomials with integer coefficients, the bound can be stated more simply.

Theorem 2. *For a square-free polynomial $P \in \mathbb{Z}[x]$ with integer coefficients of absolute value 2^τ or less, the algorithm* ANewDsc *computes isolating intervals for all real roots of P with $\tilde{O}(n^3 + n^2\tau)$ bit operations. If P has only k non-vanishing coefficients, the bound improves to $\tilde{O}(n^2(k + \tau))$.*

For general real polynomials, the bit complexity of the algorithm ANewDsc matches the bit complexity of the best algorithm known ([26]). For polynomials with integer coefficients, the bit complexity of the best algorithm known ([12, Theorem 3.1]) is $\tilde{O}(n^2\tau)$, however, for the price of using $\Omega(n^2\tau)$ bit operations for every input.[8] Both algorithms are based on Pan's approximate factorization algorithm [30], which is quite complex, and always compute all complex roots.

Our algorithm is simpler and has the additional advantage that it can be used to isolate the real roots in a given interval instead of isolating all roots. Moreover, the complexities stated in the theorems above are worst-case complexities, and we expect a better behavior for many instances. We have some theoretical evidence for this statement. For sparse integer polynomials with only k non-vanishing coefficients, the complexity bound reduces from $\tilde{O}(n^2(n + \tau))$ to $\tilde{O}(n^2(k + \tau))$ (Theorem 2). Also, if we restrict the search for roots in an interval I_0, then only the roots contained in the one-circle region $\Delta(I_0)$ of I_0 have to be considered in the complexity bound (1) in Theorem 1. More precisely, the first summand n^3 can be replaced by $n^2 \cdot m$, where m denotes the number of roots contained in $\Delta(I_0)$, and the last summand $\sum_{i=1}^{n} \log M(P'(z_i)^{-1})$ can be replaced by $\sum_{i:z_i \in \Delta(I_0)} \log M(P'(z_i)^{-1})$. We can also bound the size of the subdivision tree in terms of the number of sign changes in the coefficient sequence of the input polynomial (Theorem 27). In particular, if P is a sparse integer polynomial, e.g., a Mignotte polynomial, with only $(\log(n\tau))^{O(1)}$ non-vanishing coefficients, our algorithm generates a tree of size $(\log(n\tau))^{O(1)}$. Our algorithm generates a tree of size $(\log(n\tau))^{O(1)}$, whereas bisection methods, such as the classical Descartes method, generate a tree of size $\Omega(n\tau)$, and the continued fraction method [6] generates a tree of size $\Omega(n)$.

A modification of our algorithm can be used to refine roots once they are isolated.

[6] The $\tilde{O}$-notation suppresses polylogarithmic factors, i.e., $\tilde{O}(T) = O(T(\log T)^k)$, where k is any fixed integer.

[7] Let $L \ge 1$ be an integer and let z be real. We call $\tilde{z} = s \cdot 2^{-(L+1)}$ with $s \in \mathbb{Z}$ an *approximation of z with quality L* if $|z - \tilde{z}| \le 2^{-L}$.

[8] More precisely, Pan's algorithm uses $O(n(\log^2 n)(\log^2 n + \log b))$ arithmetic operations carried out with a precision b of size $\Omega(n(\tau + \log n))$, and thus $O(n(\log^2 n + \log \tau)\mu(n(\tau + \log n))$ bit operations, where $\mu(b)$ denotes the cost for multiplying two b-bit integers. A straight forward, but tedious, calculation yields the bound $O((n^3 + n^2\tau)(\log^6 n)(\log^2(n\tau)))$ for the bit complexity of our method, when ignoring log log-factors. This is weaker than Pan's method, however, we have neither tried to optimize our algorithm in this direction nor to consider possible amortization effects in the analysis when considering log-factors.

Theorem 3. *Let $P = P_n x^n + \ldots + P_0 \in \mathbb{R}[x]$ be a real, square-free polynomial with $1/4 \leq |P_n| \leq 1$, and let κ be a positive integer. Isolating intervals of size less than $2^{-\kappa}$ for all real roots can be computed with*

$$\tilde{O}(n \cdot (\kappa + n^2 + n \log \mathrm{Mea}(P) + \log M(\mathrm{Disc}(P)^{-1})))$$

bit operations using coefficient approximations of quality

$$\tilde{O}(\kappa + n + \tau_P + \max_i (n \log M(z_i) + \log M(P'(z_i)^{-1}))).$$

For a square-free polynomial P with integer coefficients of size less than 2^τ, isolating intervals of size less than $2^{-\kappa}$ for all real roots can be computed with $\tilde{O}(n(n^2 + n\tau + \kappa))$ bit operations.

The complexity of the root refinement algorithm is $\tilde{O}(n\kappa)$ for large κ. This is optimal up to logarithmic factors, as the size of the output is $\Omega(n\kappa)$. The complexity matches the complexity shown in [26], and when considered as a function in κ only, it also matches the complexity as shown in [18] and as sketched in [31].

1.2 Related Work

Isolating the roots of a polynomial is a fundamental and well-studied problem. One is either interested in isolating all roots, or all real roots, or all roots in a certain subset of the complex plane. A related problem is the approximate factorization of a polynomial, that is, to find $\tilde{z}_1$ to $\tilde{z}_n$ such that $\|P(x) - P_n \prod_{1 \leq i \leq n}(x - \tilde{z}_i)\|$ is small. Given the number of distinct complex roots of a polynomial P, one can derive isolating disks for all roots from a sufficiently good approximate factorization of P; see [26]. In particular, this approach applies to polynomials that are known to be square-free.

Many algorithms for approximate factorization and root isolation are known, see [23, 24, 25, 32] for surveys. The algorithms can be roughly split into two groups: There are iterative methods for simultaneously approximating all roots (or a single root if a sufficiently good approximation is already known); there are subdivision methods that start with a region containing all the roots of interest, subdivide this region according to certain rules, and use inclusion- and exclusion-predicates to certify that a region contains exactly one root or no root. Prominent examples of the former group are the Aberth-Ehrlich method (used for MPSolve [4]) and the Weierstrass-Durand-Kerner method. These algorithms work well in practice and are widely used. However, a complexity analysis and global convergence proof is missing. Prominent examples of the second group are the Descartes method [7, 9, 10, 35], the Bolzano method [5, 39], the Sturm method [8], the continued fraction method [2, 42, 43], and the splitting circle method [41, 30].

The splitting circle method was introduced by Schönhage [41] and later considerably refined by Pan [30]. Pan's algorithm computes an approximate factorization and can also be used to isolate all complex roots of a polynomial. For integer polynomials, it isolates all roots with $\tilde{O}(n^2\tau)$ bit operations. It also serves as the key routine in a recent algorithm [26] for complex root isolation, which achieves a worst case complexity similar to the one stated in our main theorem. There exists a "proof of concept" implementation of the splitting circle method in the computer algebra system Pari/GP [14], whereas we are not aware of any implementation of Pan's method itself.

The Descartes, Sturm, and continued fraction methods isolate only the real roots. They are popular for their simplicity, ease of implementation, and practical efficiency. The papers [15, 17, 35] report about implementations and experimental comparisons. The price for the simplicity is a considerably larger worst-case complexity. We concentrate on the Descartes method.

The standard Descartes method has a complexity of $\tilde{O}(n^4\tau^2)$ for isolating the real roots of an integer polynomial of degree n with coefficients bounded by 2^τ in absolute value, see [11]. The size of the recursion tree is $O(n(\tau + \log n))$, and $\tilde{O}(n)$ arithmetic operations on numbers of

bitsize $O(n^2(\tau + \log n))$ need to be performed at each node. For $\tau = \Omega(\log n)$, these bounds are tight, that is, there are examples where the recursion tree has size $\Omega(n\tau)$ and the numbers to be handled grow to integers of length $\Omega(n^2\tau)$ bits.

Johnson and Krandick [16] and Rouillier and Zimmermann [35] suggested the use of approximate arithmetic to speed up the Descartes method. They fall back on exact arithmetic when sign computations with approximate arithmetic are not conclusive. Eigenwillig et al. [10] were the first to describe a Descartes method that has no need for exact arithmetic. It works for polynomials with real coefficients given through coefficient oracles and isolates the real roots of a square-free real polynomial $P(x) = P_n x^n + \ldots + P_0$ with root separation[9] ρ, coefficients $|P_n| \geq 1$, and $|P_i| \leq 2^\tau$, with an expected cost of $O(n^4(\log(1/\rho) + \tau)^2)$ bit operations. For polynomials with integer coefficients, it constitutes no improvement. Sagraloff [38] gave a variant of the Descartes method that, when applied to integer polynomials, uses approximate arithmetic with a working precision of only $\tilde{O}(n\tau)$ bits. This leads to a bit complexity of $\tilde{O}(n^3\tau^2)$; the recursion tree has size $O(n(\tau + \log n))$, there are $\tilde{O}(n)$ arithmetic operations per node, and arithmetic on numbers of length $\tilde{O}(n\tau)$ bits is required.

As already mentioned before, the recursion tree of the Descartes method may have size $\Omega(n\tau)$, and there are only $O(n)$ nodes where both children are subdivided further. Thus, large subdivision trees must have long chains of nodes, where one child is immediately discarded. Sagraloff [36] showed how to traverse such chains more efficiently using a hybrid of bisection and Newton iteration. His method reduces the size of the recursion tree to $O(n \log(n\tau))$, which is optimal up to logarithmic factors.[10] It only applies to polynomials with integral coefficients, uses exact rational arithmetic, and achieves a bit complexity of $\tilde{O}(n^3\tau)$. In essence, the size of the recursion tree is $O(n \log(n\tau))$, there are $\tilde{O}(n)$ arithmetic operations per node, and arithmetic is on numbers of amortized length $\tilde{O}(n\tau)$ bits. Other authors have also shown how to use Newton iteration for faster convergence after collecting enough information in a slower initial phase. For instance, Renegar [33] combines the Schur-Cohn test with Newton iteration. His algorithm makes crucial use of the fact that each (sufficiently small) cluster consisting of k roots of a polynomial P induces the existence of a single nearby ordinary root of the $(k-1)$-th derivative $P^{(k-1)}$, and thus, one can apply Newton iteration to $P^{(k-1)}$ in order to efficiently compute a good approximation of this root (and the cluster) if the multiplicity k of the cluster is known. Several methods have been proposed to compute k, such as approximating the winding number around the perimeter of a disk by discretization of the contour, numerically tracking a homotopy path near a cluster, or the usage of Rouché's and Pellet's Theorem. For a more detailed discussion and more references, we refer the reader to Yakoubsohn's paper [45]. Compared to these methods, our approach is more light-weight in the sense that we consider a trial and error approach that yields less information in the intermediate steps without making sacrifices with respect to the speed of convergence. More precisely, we use a variant of Newton iteration for multiple roots in order to guess the multiplicity of a cluster (provided that such a cluster exists), however, we actually never verify our guess. Nevertheless, our analysis shows that, if there exists a well-separated cluster of roots, then our method yields the correct multiplicity. From the so obtained "multiplicity", we then compute a better approximation of the cluster (again assuming the existence of a cluster) and finally aim to verify via Descartes' Rule of Signs that we have not missed any root. If the latter cannot be verified, we fall back to bisection.

The bit complexity of our new algorithm is $\tilde{O}(n^3 + n^2\tau)$ for integer polynomials. Similar as in [36], the size of the recursion tree is $O(n \log(n\tau))$ due to the combination of bisection and Newton steps. The number of arithmetic operations per node is $\tilde{O}(n)$ and arithmetic is on numbers of amortized length $\tilde{O}(n + \tau)$ bits (instead of $\tilde{O}(n\tau)$ as in [36]) due to the use of approximate multipoint evaluation and approximate Taylor shift.

[9] The root separation of a polynomial is the minimal distance between two roots

[10] As there might be n real roots, n is a trivial lower bound on the worst-case tree size.

Root refinement is the process of computing better approximations once the roots are isolated. In [18, 22, 19, 26, 31], algorithms have been proposed which scale like $\tilde{O}(n\kappa)$ for large κ. The former two algorithms are based on the splitting circle approach and compute approximations of all complex roots. The latter two solutions are dedicated to approximate only the real roots. They combine a fast convergence method (i.e., the secant method and Newton iteration, respectively) with approximate arithmetic and efficient multipoint evaluation; however, there are details missing in [31] when using multipoint evaluation. In order to achieve complexity bounds comparable to the one stated in Theorem 3, the methods from [18, 31] need as input isolating intervals whose size is comparable to the separation of the corresponding root, that is, the roots must be "well isolated". This is typically achieved by using a fast method, such as Pan's method, for complex root isolation first. Our algorithm does not need such a preprocessing step.

Very recent work [37] on isolating the real roots of a sparse integer polynomial $P \in \mathbb{Z}[x]$ makes crucial use of a slight modification of the subroutine Newton-Test as proposed in Section 3.2. There, it is used to refine an isolating interval I for a root of P in a number of arithmetic operations that is nearly linear in the number of roots that are close to I and polynomial in $m \cdot \log(n \cdot \tau_P)$, where $n := \deg P$ and m denotes the number of non-vanishing coefficients of P. This eventually yields the first real root isolation algorithm that needs only a number of arithmetic operations over the rationals that is polynomial in the input size of the sparse representation of P. Furthermore, for very sparse polynomials (i.e. $m = O(\log^c(n\tau_P))$ with c a constant), the algorithm from [37] uses only $\tilde{O}(n\tau_P)$ bit operations to isolate all real roots of P and is thus near-optimal.

1.3 Structure of Paper and Reading Guide

We introduce our new algorithm in Section 3 and analyze its complexity in Section 4. We first derive a bound on the size of the subdivision tree (Section 4.1) and then a bound on the bit complexity (Section 4.2). Section 5 discusses root refinement. Section 2 provides background material, which we recommend to go over quickly in a first reading of the paper. We provide many references to Section 2 in Sections 3 and 4 so that the reader can pick up definitions and theorems as needed.

2 The Basics

2.1 Setting and Basic Definitions

We consider a square-free polynomial

$$P(x) = P_n x^n + \ldots + P_0 \in \mathbb{R}[x], \quad \text{where } n \geq 2 \text{ and } 1/4 \leq P_n \leq 1. \tag{2}$$

We fix the following notations.

Definition 4.
(1) $M(z) := \max(1, |z|)$ for all $z \in \mathbb{C}$.
(2) $\|P\| := \|P\|_1 := |P_0| + \ldots + |P_n|$ denotes the 1-norm of P, and $\|P\|_\infty := \max_i |P_i|$ denotes the infinity-norm of P.
(3) $\tau_P := M(\log \|P\|_\infty)$.
(4) $z_1, \ldots, z_n \in \mathbb{C}$ are the complex roots of P.
(5) For each root z_i, we define the *separation of* z_i as the value $\sigma_i := \sigma(z_i, P) := \min_{j \in \{1,\ldots,n\} \setminus i} |z_i - z_j|$. The *separation of* P is defined as $\sigma_P := \min_i \sigma(z_i, P)$.
(6) $\Gamma_P := M(\log \max_i |z_i|)$ denotes the *logarithmic root bound* of P, and
(7) $\mathrm{Mea}(P) := |P_n| \cdot \prod_{i=1}^n M(|z_i|)$ denotes the *Mahler Measure* of P.

(8) For an interval $I = (a, b)$, $m(I) := \frac{a+b}{2}$ denotes the *midpoint* and $w(I) := b - a$ the *width of* I. The open disk in complex space with center $m(I)$ and radius $\frac{w(I)}{2}$ is denoted by $\Delta(I)$. We call $\Delta(I)$ the *one-circle region of* I.[11]
(9) $\mathcal{M}(I)$ denotes the set of roots of P which are contained in $\Delta(I)$.
(10) A dyadic fraction is any rational of the form $s \cdot 2^{-\ell}$ with $s \in \mathbb{Z}$ and $\ell \in \mathbb{Z}_{\geq 0}$.

We assume the existence of an oracle that provides arbitrary good approximations of the polynomial P. Let $L \geq 1$ be an integer and let z be a real. We call $\tilde{z} = s \cdot 2^{-(L+1)}$ with $s \in \mathbb{Z}$ an *(absolute) L-approximation* of z or an *approximation of z with quality L* if $|z - \tilde{z}| \leq 2^{-L}$. We call a polynomial $\tilde{P} = \tilde{P}_n x^n + \ldots + \tilde{P}_0$ an *(absolute) L-approximation* of P or an *approximation of quality L* if every coefficient of $\tilde{P}$ is an approximation of quality L of the corresponding coefficient of P. We assume that we can obtain such an approximation $\tilde{P}$ at $O(n(L + \tau_P))$ cost. This is the cost of reading the coefficients of $\tilde{P}$.

We have $\tau_P \leq M(\log(2^n \cdot \mathrm{Mea}(P))) \leq M(n + n\Gamma_P) = n(1 + \Gamma_P) \leq 2n\Gamma_P$. According to [26, Theorem 1] (or [38, Section 6.1]), we can compute an integer approximation $\tilde{\Gamma}_P$ of Γ_P with

$$\Gamma_P + 1 \leq \tilde{\Gamma}_P \leq \Gamma_P + 8 \log n + 1 \tag{3}$$

with $\tilde{O}(n^2 \Gamma_P)$ many bit operations. From $\tilde{\Gamma}_P$, we can then immediately derive a $\Gamma = 2^\gamma$, with $\gamma := \lceil \log \tilde{\Gamma}_P \rceil \in \mathbb{N}_{\geq 1}$, such that

$$\Gamma_P + 1 \leq \tilde{\Gamma}_P \leq \Gamma \leq 2 \cdot \tilde{\Gamma}_P \leq 2 \cdot (\Gamma_P + 8 \log n + 1). \tag{4}$$

Thus, $2^\Gamma = 2^{2^\gamma}$ is an upper bound for the modulus of all roots (in fact, we have $2^\Gamma \geq \max_i |z_i| + 1$ for all $i = 1, \ldots, n$), and $\Gamma = O(\Gamma_p + \log n)$.

2.2 Approximate Polynomial Evaluation

We introduce the notions *multipoint* (Definition 9) and *admissible point* (Definition 7). A point x^* in a set X is admissible if $|P(x^*)| \geq \frac{1}{4} \cdot |P(x)|$ for all $x \in X$. We show how to efficiently compute an admissible point in a multipoint (Corollary 11) and derive a lower bound on the value of P at such a point. Corollary 11 is our main tool for choosing subdivision points.

Theorem 5. *Let P be a polynomial as defined in (2), x_0 be a real point, and L be a positive integer.*

(a) The algorithm stated in the proof of part (a) computes an approximation $\tilde{y}_0$ of $y_0 := P(x_0)$ with $|y_0 - \tilde{y}_0| \leq 2^{-L}$ with

$$\tilde{O}(n(\tau_P + n \log M(x_0) + L)).$$

bit operations using approximations of the coefficients of P and the point x_0 of quality $O(\tau_P + n \log M(x_0) + L + \log n)$.

(b) Suppose $y_0 \neq 0$. The algorithm stated in the proof of part (b) computes an integer t with $2^{t-1} \leq |y_0| \leq 2^{t+1}$ with

$$\tilde{O}(n(\tau_P + n \log M(x_0) + \log M(y_0^{-1})))$$

bit operations. The computation uses fixed-precision arithmetic with a precision of $O(\tau_P + n \log M(x_0) + M(y_0^{-1}) + \log n)$ bits.

Proof. Part (a) follows directly from [18, Lemma 3], where it has been shown that we can compute a desired approximation $\tilde{y}_0$ via the Horner scheme and fixed-precision interval arithmetic, with a precision of $O(\tau_P + n \log M(x_0) + L + \log n)$ bits.

[11] The choice of name will become clear when we discuss Descartes' rule of signs in Section 2.3; see also Figure 1.

For (b), we consider $L = 1, 2, 4, 8, \ldots$ and compute absolute L-bit approximations $\tilde{y}_0$ of y_0 until we obtain an approximation $\tilde{y}_0$ with $|\tilde{y}_0| \geq 2^{2-L}$. Since $\tilde{y}_0$ is an absolute L-bit approximation of y_0, $|\tilde{y}_0 - y_0| \leq 2^{-L} \leq |\tilde{y}_0|/4$. Since $\tilde{y}_0$ is a dyadic fraction, we can determine $t \in \mathbb{Z}$ with $|t - \log|\tilde{y}_0|| \leq 1/2$. Then, $2^{t-1} \leq (3/4)2^{-1/2}2^t \leq (3/4)|\tilde{y}_0| \leq |y_0| \leq (5/4)|\tilde{y}_0| \leq (5/4)2^{1/2}2^t \leq 2^{t+1}$. Obviously, we succeed if $L \geq 2\log M(y_0^{-1})$, and since we double L in each step, we need at most $O(\log\log M(y_0^{-1}))$ many steps. Up to logarithmic factors, the total cost is dominated by the cost of the last iteration, which is bounded by $\tilde{O}(n(\tau_P + n\log M(x_0) + \log M(y_0^{-1})))$ bit operations according to Part (a). □

It has been shown [19, 21, 22, 34, 44] that the cost for approximately evaluating a polynomial of degree n at $N = O(n)$ points is is of the same order as the cost of approximately evaluating it at a single point.

Theorem 6 ([19, 21, 22]). *Let P be a polynomial as in (2), let $x_1, \ldots, x_N$ be real points with $N = O(n)$, and let L be a positive integer. The algorithm in [19, 21, 22] computes approximations $\tilde{y}_i$ of $y_i := P(x_i)$ with $|y_i - \tilde{y}_i| \leq 2^{-L}$, $i = 1, \ldots, N$, with*

$$\tilde{O}(n(n + \tau_P + n\log M(\max_i |x_i|) + L))$$

bit operations using approximations of the coefficients of P as well as the points x_i of quality $O(\tau_P + n\log M(\max_i |x_i|) + L + n\log n)$.

We frequently need to select a point x_i from a given set $X = \{x_1, \ldots, x_N\}$ of points at which $|P(x_i)|$ is close to maximal.

Definition 7. Let $X := \{x_1, \ldots, x_N\}$ be a set of N real points. We call a point $x^* \in X$ *admissible with respect to X* (or just *admissible* if there is no ambiguity) if $|P(x^*)| \geq \frac{1}{4} \cdot \max_i |P(x_i)|$.

Fast approximate multipoint evaluation allows us to find an admissible point efficiently.

Algorithm: Admissible Point

Input: A polynomial $P(x)$ as in (2), and a set $X = \{x_1, \ldots, x_N\}$ of points.

Guarantee: $\lambda := \max_i |P(x_i)| > 0$.

Output: An admissible point $x^* \in X$ and an integer t with $2^{t-1} \leq \lambda \leq 2^{t+1}$.

(1) $L := 1/2$.

(2) repeat

  (2.1) $L := 2 \cdot L$

  (2.2) Compute approximations $\tilde{\lambda}_i$ of quality L for the values $P(x_i)$ for $1 \leq i \leq N$.

  (2.3) $\tilde{\lambda} := \max_i |\tilde{\lambda}_i|$.

  until $\tilde{\lambda} \leq 2^{2-L}$.

(3) Let i_0 be an index with $\tilde{\lambda} := |\tilde{\lambda}_{i_0}|$ and t be an integer with $|t - \log|\tilde{\lambda}|| \leq \frac{1}{2}$.

(4) return x_{i_0} and t.

An argument similar to the one as in the proof of Part (b) in Lemma 5 now yields the following result:

Lemma 8. *Let $X := \{x_1, \ldots, x_N\}$ be a set of $N = O(n)$ real points. The algorithm* **Admissible Point** *applied to X selects an admissible point $x^* \in X$ and an integer t with*

$$2^{t-1} \leq |P(x^*)| \leq \lambda := \max_i |P(x_i)| \leq 2^{t+1}$$

using $\tilde{O}(n(n + \tau_P + n \log M(\max_i |x_i|) + \log M(\lambda^{-1})))$ bit operations. It requires approximations of the coefficients of P and the points x_i of quality $O(n + \tau_P + n \log M(\max_i |x_i|) + \log M(\lambda^{-1}))$.

We will mainly apply the Lemma in the situation where X is a set of $N = 2 \cdot \lceil n/2 \rceil + 1$ equidistant points. In this situation, we can prove a lower bound on λ in the Lemma above in terms of the separations of the roots z_i, the absolute values of the derivatives $P'(z_i)$, and the number of roots contained in a neighborhood of the points X.

Definition 9. For a real point m and a real positive value ϵ, the (m, ϵ)-*multipoint* $m[\epsilon]$ is defined as

$$m[\epsilon] := \{m_i := m + (i - \lceil n/2 \rceil) \cdot \epsilon \; ; \; i = 0, \ldots, 2 \cdot \lceil n/2 \rceil\}. \tag{5}$$

Lemma 10. *Let m be a real point, let ϵ be a real positive value, and let K be a positive real with $K \geq 2 \cdot \lceil n/2 \rceil$. If the disk $\Delta := \Delta_{K \cdot \epsilon}(m)$ with radius $K \cdot \epsilon$ and center m contains at least two roots of P, then each admissible point $m^* \in m[\epsilon]$ satisfies*

$$|P(m^*)| > 2^{-4n-1} \cdot K^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)| \quad \textit{for all roots } z_i \in \Delta,$$

where $\mu(\Delta)$ denotes the number of roots of P contained in Δ.

Proof. Since the number of points $m_i \in m[\epsilon]$ is larger than the number of roots of P and since their pairwise distances are ϵ, there exists a point $m_{i_0} \in m[\epsilon]$ whose distance to all roots of P is at least $\epsilon/2$. We will derive a lower bound on $|P(m_{i_0})|$. Let z_i be any root in Δ. For any different root $z_j \in \Delta$, we have $|z_i - z_j|/|m_{i_0} - z_j| < 2K\epsilon/(\epsilon/2) = 4K$, and, for any root $z_j \notin \Delta$, we have $|z_i - z_j|/|m_{i_0} - z_j| \leq 2K/(K - \lceil n/2 \rceil) \leq 4$. Hence, it follows that

$$\begin{aligned}
|P(m_{i_0})| &= |P_n| \cdot |m_{i_0} - z_i| \cdot \prod_{j \neq i} |m_{i_0} - z_j| > |P_n| \cdot \frac{\epsilon}{2} \cdot (4K)^{-\mu(\Delta)} \cdot 4^{-n+\mu(\Delta)} \cdot \prod_{j \neq i} |z_i - z_j| \\
&= |P'(z_i)| \cdot \frac{\epsilon}{2n} \cdot 4^{-n} \cdot K^{-\mu(\Delta)} = 2^{-(\log n + 1) - 2n} \cdot \epsilon \cdot K^{-\mu(\Delta)} \cdot |P'(z_i)| \\
&> 2^{-2n - \log n - 2} \cdot K^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)|,
\end{aligned}$$

where we used $\sigma(z_i, P) < 2K\epsilon$. Hence, for each admissible point $m^* \in m[\epsilon]$, it follows that $|P(m^*)| \geq \frac{|P(m_{i_0})|}{4} \geq 2^{-4n-1} \cdot K^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)|$. □

We summarize the discussion of this section in the following corollary.

Corollary 11. *Let m be a real point, let ϵ be a real positive value, and let K be a positive real with $K \geq 2 \cdot \lceil n/2 \rceil$, and assume that the disk $\Delta := \Delta_{K \cdot \epsilon}(m)$ contains at least two roots of P. Then, for each admissible point $m^* \in m[\epsilon]$,*

$$|P(m^*)| > 2^{-4n-1} \cdot K^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)| \quad \textit{for all roots } z_i \in \Delta. \tag{6}$$

The algorithm **Admissible Point** *applied to $m[\epsilon]$ selects an admissible point from the set with*

$$\tilde{O}(n(\mu(\Delta) \cdot \log K + n + \tau_P + n \log M(|m| + n\epsilon) + \log M(\max_{z_i \in \Delta}(\sigma(z_i, P) \cdot |P'(z_i)|)))). \tag{7}$$

bit operations. It requires approximations of the coefficients of P and the points m_i of quality

$$O(\mu(\Delta) \cdot \log K + n + \tau_P + n \log M(|m| + n\epsilon) + \log M(\max_{z_i \in \Delta}(\sigma(z_i, P) \cdot |P'(z_i)|)).$$

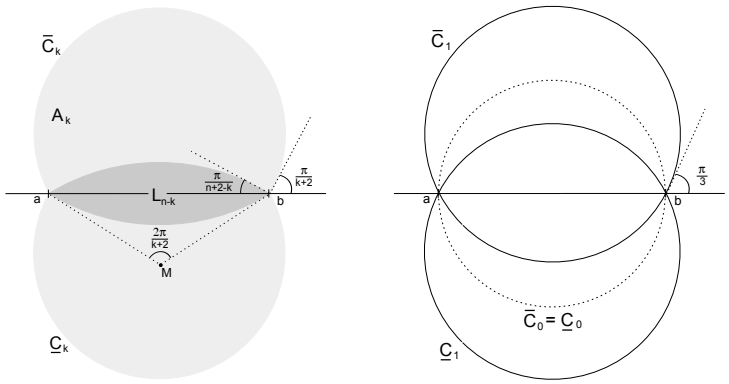

Figure 1: For any k, $0 \le k \le n$, the *Obreshkoff disks* $\overline{C}_k$ and $\underline{C}_k$ for $I = (a, b)$ have the endpoints of I on their boundaries; their centers see the line segment (a, b) under the angle $\frac{2\pi}{k+2}$. The *Obreshkoff lens* L_k is the interior of $\overline{C}_k \cap \underline{C}_k$, and the *Obreshkoff area* A_k is the interior of $\overline{C}_k \cup \underline{C}_k$. Any point (except a and b) on the boundary of A_k sees I under the angle $\frac{\pi}{k+2}$, and any point (except a and b) on the boundary of L_k sees I under the angle $\pi - \frac{\pi}{k+2}$. We have $L_n \subseteq \ldots \subseteq L_1 \subseteq L_0$ and $A_0 \subseteq A_1 \subseteq \ldots \subseteq A_n$. The cases $k = 0$ and $k = 1$ are of special interest: The circles $\overline{C}_0$ and $\underline{C}_0$ coincide. They have their centers at the midpoint of I. The circles $\overline{C}_1$ and $\underline{C}_1$ are the circumcircles of the two equilateral triangles having I as one of their edges. We call $A_0 = \Delta(I)$ and A_1 the *one-circle* and the *two-circle region* for I, respectively.

Corollary 11 is a key ingredient of our root isolation algorithm. We will appeal to it whenever we have to choose a subdivision point. Assume, in an ideal world with real arithmetic at unit cost, we choose a subdivision point m. The polynomial P may take a very small value at m, and this would lead to a high bit complexity. Instead of choosing m as the subdivision point, we choose a nearby admissible point $m^* \in m[\epsilon]$ and are guaranteed that $|P(m^*)|$ has at least the value stated in (6). The fact that $|P|$ is reasonably large at m^* will play a crucial role in the analysis of our algorithm, cf. Theorem 31.

2.3 Descartes' Rule of Signs in Monomial and in Bernstein Basis

This section provides a brief review of Descartes' rule of signs. We remark that most of what follows in this section has already been presented (in more detail) elsewhere (e.g. in [9, 10, 36]); however, for the sake of a self-contained representation, we have decided to reiterate the most important results which are needed for our algorithm and its analysis.

In order to estimate the number m_I of roots of P contained in an interval $I = (a, b) \subseteq \mathcal{I} = (-2^\Gamma, 2^\Gamma)$, we use Descartes' rule of signs: *For a polynomial $F(x) = \sum_{i=0}^{N} f_i x^i \in \mathbb{R}[x]$, the number m of positive real roots of F is bounded by the number v of sign variations*[12] *in its coefficient sequence $(f_0, \ldots, f_N)$ and, in addition, $v \equiv m \mod 2$.* We can apply this rule to the polynomial P and the interval I by considering a Möbius transformation $x \mapsto \frac{ax+b}{x+1}$ that maps $(0, +\infty)$ one-to-one onto I. Namely, let

$$P_I(x) := \sum_{i=0}^{n} p_{I,i} \cdot x^i := (x+1)^n \cdot P\left(\frac{ax+b}{x+1}\right), \tag{8}$$

and let $v_I := \mathrm{var}(P, I) := \mathrm{var}(p_{I,0}, \ldots, p_{I,n})$ be defined as *the number of sign variations in the*

[12] Zero entries are not considered. For instance, $\mathrm{var}(-1, 0, 0, 2, 0, -1) = \mathrm{var}(-1, 2, -1) = 2$.

coefficient sequence $(p_{I,0}, \ldots, p_{I,n})$ *of* P_I. Then, v_I is an upper bound for m_I (i.e. $v_I \geq m_I$) and v_I has the same parity as m_I (i.e. $v_I \equiv m_I \mod 2$). Notice that the latter two properties imply that $v_I = m_I$ if $v_I \leq 1$.

The following theorem states that the number v_I is closely related to the number of roots located in specific neighborhoods of the interval I.

Theorem 12 ([28, 29]). *Let* $I = (a, b)$ *be an open interval and* $v_I = \mathrm{var}(P, I)$. *If the Obreshkoff lens* L_{n-k} *(see Figure 1 for the definition of* L_{n-k}*) contains at least* k *roots (counted with multiplicity) of* P, *then* $v_I \geq k$. *If the Obreshkoff area* A_k *contains no more than* k *roots (counted with multiplicity) of* P, *then* $v_I \leq k$. *In particular,*

$$\#\text{ of roots of } P \text{ in } L_n \leq v_I = \mathrm{var}(P, I) \leq \#\text{ of roots of } P \text{ in } A_n.$$

We remark that the special cases $k = 0$ and $k = 1$ appear as the one-circle and the two-circle theorems in the literature (e.g. [3, 9]). Theorem 12 implies that if the one-circle region $A_0 = \Delta(I)$ of I contains a root z_i with separation $\sigma(z_i, P) > 2w(I) = 2(b - a)$, then this root must be real and $v_I = 1$. Namely, the condition on $\sigma(z_i, P)$ guarantees that the two-circle region A_1 contains z_i but no other root of P. If the one-circle region contains no root, then $v_I = 0$. Hence, it follows that each interval I of width $w(I) < \sigma_P/2$ yields $v_I = 0$ or $v_I = 1$. In addition, we state the variation diminishing property of the function $\mathrm{var}(P, I)$; e.g., see [9, Corollary 2.27] for a self-contained proof:

Theorem 13 ([40]). *Let* I *be an interval and* I_1 *and* I_2 *be two disjoint subintervals of* I. *Then,*

$$\mathrm{var}(P, I_1) + \mathrm{var}(P, I_2) \leq \mathrm{var}(P, I).$$

In addition to the above formulation of Descartes' rule of signs in the monomial basis, we provide corresponding results for the representation of $P(x)$ in terms of the Bernstein basis $B_0^n, \ldots, B_n^n$ with respect to $I = (a, b)$, where

$$B_i^n(x) := B_i^n[a, b](x) := \binom{n}{i} \frac{(x - a)^i (b - x)^{n-i}}{(b - a)^n}, \quad 0 \leq i \leq n. \tag{9}$$

If $P(x) = \sum_{i=0}^n b_i B_i^n[a, b](x)$, we call $B = (b_0, \ldots, b_n)$ the Bernstein representation of P with respect to I. For the first and the last coefficient, we have $b_0 = P(a)$ and $b_n = P(b)$. The following Lemma provides a direct correspondence between the coefficients of the polynomial P_I from (8) and the entries of B. For a self-contained proof, we refer to [9].

Lemma 14. *Let* $I = (a, b)$ *be an interval,* $P(x) = \sum_{i=0}^n b_i B_i^n[a, b](x)$ *be the Bernstein representation of* P *with respect to* I, *and* $P_I(x) = \sum_{i=0}^n p_{I,i} \cdot x^i$ *as in (8). It holds that*

$$p_{I,i} = b_{n-i} \cdot \binom{n}{i} \quad \text{for all } i = 0, \ldots, n. \tag{10}$$

In particular, v_I *coincides with the number of sign variations in the sequence* $(b_0, \ldots, b_n)$.

In essence, the above lemma states that, when using Descartes' Rule of Signs, it makes no difference whether we consider the Bernstein basis representation of P with respect to I or the polynomial P_I from (8). This will turn out to be useful in the next section, where we review results from [10] which allow us to treat the cases $v_I = 0$ and $v_I = 1$ by using approximate arithmetic.

2.4 Descartes' Rule of Signs with Approximate Arithmetic

We introduce the 0-Test and 1-Test for intervals I with the following properties.

(1) If $\mathrm{var}(P, I) = 0$ ($\mathrm{var}(P, I) = 1$), then the 0-Test (1-Test) for I succeeds.
(2) If the 0-Test (1-Test) for I succeeds, I contains no (exactly one) root of P.
(3) The 0-Test and the 1-Test for I can be carried out efficiently with approximate arithmetic, see Corollaries 18 and 21.

2.4.1 The case $\mathrm{var}(P, I) = 0$

Consider the following Lemma, which follows directly from [10, Lemma 5] and its proof:

Lemma 15. *Let $P(x) = \sum_{i=0}^{n} b_i B_i^n[a, b](x)$ be the Bernstein representation of P with respect to the interval $I = (a, b)$, and let m be a subdivision point contained in $[m(I) - \frac{w(I)}{4}, m(I) + \frac{w(I)}{4}]$. The Bernstein representations of P with respect to $I' = (a, m)$ and $I'' = (m, b)$ are given by $P(x) = \sum_{i=0}^{n} b'_i B_i^n[a, m](x)$ and $P(x) = \sum_{i=0}^{n} b''_i B_i^n[m, b](x)$, respectively. Suppose that* $\mathrm{var}(P, I) = 0$. *Then,* $\mathrm{var}(P, I') = \mathrm{var}(P, I'') = 0$, *and*

$$|b'_i|, |b''_i| > \min(|P(a)|, |P(b)|) \cdot 4^{-(n+1)} \quad \text{for all } i = 0, \ldots, n.$$

Combining the latter result with Lemma 14 now yields:

Corollary 16. *Let I, I', I'' be intervals as in Lemma 15, and let L be an integer with*

$$L \geq L_{I,0} := \log M(\min(|P(a)|, |P(b)|)^{-1}) + 2(n + 1) + 1. \tag{11}$$

Suppose that $\sum_{i=0}^{n} \tilde{p}_{I',i} \cdot x^i$ and $\sum_{i=0}^{n} \tilde{p}_{I'',i} \cdot x^i$ are absolute L-bit approximations of $P_{I'}$ and $P_{I''}$, respectively. If $\mathrm{var}(P, I) = 0$, *then* $\mathrm{var}(\tilde{p}_{I',0}, \ldots, \tilde{p}_{I',n}) = \mathrm{var}(\tilde{p}_{I'',0}, \ldots, \tilde{p}_{I'',n}) = 0$, *and $|\tilde{p}_{I',i}|, |\tilde{p}_{I'',i}| > 2^{-L}$ for all $i = 0, \ldots, n$.*

Proof. Suppose that $\mathrm{var}(P, I) = 0$, then Lemma 15 yields $|b'_i|, |b''_i| > \min(|P(a)|, |P(b)|) \cdot 4^{-(n+1)} \geq 2 \cdot 2^{-L_{I,0}}$ for all $i = 0, \ldots, n$, and, in addition, all coefficients b'_i and b''_i have the same sign. Since $p_{I',i} = \binom{n}{i} b'_{n-i}$ and $p_{I'',i} = \binom{n}{i} b''_{n-i}$, it follows that the coefficients $p_{I',i}$ and $p_{I'',i}$ also have absolute value larger than $2 \cdot 2^{-L_{I,0}}$. Thus, $|\tilde{p}_{I',i}|, |\tilde{p}_{I'',i}| > 2^{-L_{I,0}}$ since $|p_{I',i} - \tilde{p}_{I',i}| \leq 2^{-L_0}$ and $|p_{I'',i} - \tilde{p}_{I'',i}| \leq 2^{-L} \leq 2^{-L_{I,0}}$ for all i. In addition, all coefficients $\tilde{p}_{I',i}$ and $\tilde{p}_{I'',i}$ have the same sign because this holds for their exact counterparts. □

The above corollary allows one to discard an interval I by using approximate arithmetic with a precision directly related to the absolute values of P at the endpoints of I. More precisely, we consider the following exclusion test that applies to intervals $I = (a, b)$ with $P(a) \neq 0$ and $P(b) \neq 0$. Comments explaining the rationale behind our choices are typeset in italic and start with the symbol $/\!/$.

Algorithm: 0-Test

Input: A polynomial $P(x)$ as in (2) and an interval $I = (a, b)$ with $P(a) \neq 0$ and $P(b) \neq 0$.

Output: True or False. In case of True, I contains no root of P. In case of False, it holds that $\mathrm{var}(P, I) > 0$.

(1) Compute integers t_a and t_b with $2^{t_a - 1} \leq |P(a)| \leq 2^{t_a + 1}$ and $2^{t_b - 1} \leq |P(b)| \leq 2^{t_b + 1}$ using the algorithm **Admissible Point** with input $X = \{a\}$ and $X = \{b\}$, respectively.

$/\!/$ Notice that, according to Lemma 5, we can compute t_a and t_b with a number of bit operations bounded by $\tilde{O}(n(\tau_P + n \log M(a) + n \log M(b) + \log M(P(a)^{-1}) + \log M(P(b)^{-1})))$.

(2) $L := M(-\min(t_a - 1, t_b - 1)) + 2(n+1) + 1$

// From the definition of $L_{I,0}$ in (11), it follows that $L_{I,0} \leq L \leq L_{I,0} + 2$.

(3) $I' := (a, m(I))$ and $I'' := (m(I), b)$

(4) Compute absolute L-bit approximations $\tilde{P}_{I'} = \sum_{i=0}^{n} \tilde{p}_{I',i} \cdot x^i$ and $\tilde{P}_{I''} := \sum_{i=0}^{n} \tilde{p}_{I'',i} \cdot x^i$ of the polynomials $P_{I'}$ and $P_{I''}$, respectively.

// For an efficient solution of this step, consider the algorithm from the proof of Lemma 17.

(5) If $\mathrm{var}(\tilde{p}_{I',0}, \ldots, \tilde{p}_{I',n}) = \mathrm{var}(\tilde{p}_{I'',0}, \ldots, \tilde{p}_{I'',n}) = 0$, and $|\tilde{p}_{I',j}| > 2^{-L}$ and $|\tilde{p}_{I'',j}| > 2^{-L}$ for all $j = 0, \ldots, n$, then return True.

// We conclude that $\mathrm{var}(p_{I',0}, \ldots, p_{I',n}) = \mathrm{var}(p_{I'',0}, \ldots, p_{I'',n}) = 0$. Since $\tilde{p}_{I'',0} = P(m(I))$, we also have $P(m(I)) \neq 0$, and thus I contains no root of P.

(6) return False.

It remains to provide an efficient method to compute an absolute L-bit approximation of a polynomial P_I as required in the 0-Test:

Lemma 17. *Let $I = (a, b)$ be an interval, and let L be a positive integer. The algorithm stated in the proof computes an absolute L-bit approximation $\tilde{P}_I(x) = \sum_{i=0}^{n} \tilde{p}_{I,i}$ of $P_I(x) = \sum_{i=0}^{n} p_{I,i} x^i$ with*

$$\tilde{O}(n(n + \tau_P + n \log M(a) + n \log M(b) + L))$$

bit operations. It requires approximations of the coefficients of P and the endpoints of I of quality $O(n + \tau_P + n \log M(a) + n \log M(b) + L)$.

Proof. The computation of P_I decomposes into four steps: First, we substitute x by $a + x$, which yields the polynomial $P_1(x) := P(a + x)$. Second, we substitute x by $w(I) \cdot x$ in order to obtain $P_2(x) := P_1(a + w(I) \cdot x)$. Third, the coefficients of P_2 are reversed (i.e. the i-th coefficient is replaced by the $(n - i)$-th coefficient), which yields the polynomial $P_3(x) = x^n P_2(1/x) = x^n P(a + w(I)/x)$. In the last step, we compute the polynomial $P_4(x) := P_3(x+1) = (x+1)^n P(a + w(I)/(x+1)) = P_I(x)$.

Now, for the computation of an absolute L-bit approximation $\tilde{P}_I$, we proceed as follows: Let L_1 be a positive integer, which will be specified later. According to [21, Theorem 14] (or [41, Theorem 8.4]), we can compute an absolute L_1-bit approximations $\tilde{P}_1$ of P_1 with $\tilde{O}(n(n + \tau_P + n \log M(a) + L_1))$ bit operations, where we used that the coefficients of P have absolute value of size 2^{τ_P} or less. For this step, the coefficients of P as well as the endpoint a have to be approximated with quality $\tilde{O}(n + \tau_P + n \log M(a) + L_1)$. The coefficients of P_1 have absolute value less than $2^{n+\tau_P} M(a)^n$, and thus, the coefficients of $\tilde{P}_1$ have absolute value less than $2^{n+\tau_P} M(a)^n + 1 < 2^{n+1+\tau_P} M(a)^n$. Computing $w(I)^i$ for all $i = 0, \ldots, n$ with quality L_1 takes $\tilde{O}(n(n \log M(w(I)) + L_1)) = \tilde{O}(n(n \log M(a) + n \log M(b) + L_1))$ bit operations. This yields an approximation $\tilde{P}_2$ of P_2 with quality $L_2 := L_1 - n - 1 - \tau_P - n \log M(a)$.

The coefficients of $\tilde{P}_2$ have absolute value less than $2^{n+1+\tau_P} M(a)^n M(w(I))^n$. Reversing the coefficients of $\tilde{P}_2$ trivially yields an absolute L_2-bit approximation $\tilde{P}_3$ of P_3. For the last step, we again apply [21, Theorem 14] to show that we can compute an absolute L-bit approximation of $P_4 = P_3(x+1)$ from an L_3-bit approximation of P_3, where L_3 is an integer of size $\tilde{O}(L + n + \tau_P + n \log M(a) + n \log M(w(I)))$. The cost for this computation is bounded by $\tilde{O}(nL_3)$ bit operations. Hence, it suffices to start with an integer L_1 of size $\tilde{O}(L + n + \tau_P + n \log M(a) + n \log M(w(I)))$. This shows the claimed bound for the needed input precision, where we use that $w(I) \leq |a| + |b|$.

The bit complexity for each of the two Taylor shifts (i.e. $x \mapsto a+x$ and $x \mapsto x+1$) as well as for the approximate scaling (i.e. $x \mapsto w(I) \cdot x$) is bounded by $\tilde{O}(n(n+\tau_P+n\log M(a)+n\log M(b)+L))$ bit operations. □

The above lemma (applied to the intervals $I' = (a, m(I))$ and $I'' = (m(I), b)$) now directly yields a bound on the bit complexity for the 0-Test:

Corollary 18. *For an interval $I = (a, b)$, the 0-Test requires*

$$\tilde{O}(n(n+\tau_P+n\log M(a)+n\log M(b)+\log M(\min(|P(a)|,|P(b)|)^{-1})) \tag{12}$$

bit operations using approximations of the coefficients of P and the endpoints of I of quality $O(n+\tau_P+n\log M(a)+n\log M(b)+\log M(\min(|P(a)|,|P(b)|)^{-1}))$.

2.4.2 The case $\mathrm{var}(P, I) = 1$

We need the following result, which follows directly from [10, Lemma 6] and its proof.

Lemma 19. *With the same definitions as in Lemma 15, suppose that $\mathrm{var}(P, I) = 1$ and $P(m) \neq 0$. Then,*

$$|b_i'|, |b_i''| > \min(|P(a)|, |P(b)|, |P(m)|) \cdot 16^{-n} \quad \text{for all } i = 0, \ldots, n.$$

Furthermore, $\mathrm{var}(P, I') = 1$ (and $\mathrm{var}(P, I'') = 0$) or $\mathrm{var}(P, I'') = 1$ (and $\mathrm{var}(P, I') = 0$).

Again, combining the latter result with Lemma 14 yields the following result, whose proof is completely analogous to the proof of Corollary 20.

Corollary 20. *With the same definitions as in Lemma 15 and Lemma 19, let L be an integer with*

$$L \geq L_{I,1} := \log M(\min(|P(a)|, |P(b)|, |P(m)|)^{-1}) + 4n + 1, \tag{13}$$

and let $\sum_{i=0}^{n} \tilde{p}_{I',i} \cdot x^i$ and $\sum_{i=0}^{n} \tilde{p}_{I'',i} \cdot x^i$ be absolute L-bit approximations of $P_{I'}$ and $P_{I''}$, respectively. Suppose that $\mathrm{var}(P, I) = 1$ and $P(m) \neq 0$. Then, it follows that $|\tilde{p}_{I',i}|, |\tilde{p}_{I'',i}| > 2^{-L}$ for all $i = 0, \ldots, n$, and, in addition, $\mathrm{var}(\tilde{p}_{I',0}, \ldots, \tilde{p}_{I',n}) = 1$ (and $\mathrm{var}(\tilde{p}_{I'',0}, \ldots, \tilde{p}_{I'',n}) = 0$) or $\mathrm{var}(\tilde{p}_{I',0}, \ldots, \tilde{p}_{I',n}) = 1$ (and $\mathrm{var}(\tilde{p}_{I'',0} \ldots, \tilde{p}_{I'',n}) = 0$).

Based on the above Corollary, we can now formulate the 1-*Test*, which applies to intervals $I = (a, b)$ with $P(a) \neq 0$ and $P(b) \neq 0$:

Algorithm: 1-Test

Input: A polynomial $P(x)$ as in (2) and an interval $I = (a, b)$ with $P(a) \neq 0$ and $P(b) \neq 0$.

Output: True or False. In case of True, the algorithm also returns an interval I', with $I' \subset I$, and such that $\frac{1}{4} \cdot w(I) \leq w(I') \leq \frac{3}{4} \cdot w(I)$, $\mathrm{var}(f, I) = 1$, and $I \backslash I'$ contains no root. In case of False, it holds that $\mathrm{var}(P, I) \neq 1$.

(1) Compute integers t_a and t_b with $2^{t_a-1} \leq |P(a)| \leq 2^{t_a+1}$ and $2^{t_b-1} \leq |P(b)| \leq 2^{t_b+1}$ using the algorithm **Admissible Point**.

(2) For $\epsilon := w(I) \cdot 2^{-\lceil \log n+2 \rceil} \leq \frac{w(I)}{4n}$, compute an admissible point $m \in m(I)[\epsilon]$ and an integer t with $2^{t+1} \geq \max_i |P(m_i)| \geq |P(m^*)| \geq 2^{t-1}$ using the algorithm **Admissible Point**.

(3) $L := M(-\min(t_a - 1, t_b - 1, t - 1)) + 4n + 2$

(4) $I' := (a, m)$ and $I'' := (m, b)$

// Notice that each point from $m(I)[\epsilon]$ is contained in $[m(I) - \frac{w(I)}{4}, m(I) + \frac{w(I)}{4}]$ since $\lceil n/2 \rceil \cdot 2^{-\lceil \log n + 2 \rceil} < \frac{1}{4}$. In addition, we have $L_{I,1} \leq L \leq L_{I,1} + 2$. Thus, we can use Corollary 20.

(5) Compute absolute L-bit approximations $\tilde{P}_{I'} = \sum_{i=0}^{n} \tilde{p}_{I',i} \cdot x^i$ and $\tilde{P}_{I''} := \sum_{i=0}^{n} \tilde{p}_{I'',i} \cdot x^i$ of the polynomials $P_{I'}$ and $P_{I''}$, respectively.

(6) If $|\tilde{p}_{I',j}| \leq 2^{-L}$ or $|\tilde{p}_{I'',j}| > 2^{-L}$ for some $j = 0, \ldots, n$ then return False

// Corollary 20 implies that $\mathrm{var}(P, I) \neq 1$.

(7) If $\mathrm{var}(\tilde{p}_{I',0}, \ldots, \tilde{p}_{I',n}) = 1$ and $\mathrm{var}(\tilde{p}_{I'',0}, \ldots, \tilde{p}_{I'',n}) = 0$ then return True and I'

// Corollary 20 implies that $\mathrm{var}(P, I') = 1$ and $\mathrm{var}(P, I'') = 0$.

(8) If $\mathrm{var}(\tilde{p}_{I',0}, \ldots, \tilde{p}_{I',n}) = 0$ and $\mathrm{var}(\tilde{p}_{I'',0}, \ldots, \tilde{p}_{I'',n}) = 1$ then return True and I''

// Corollary 20 implies that $\mathrm{var}(P, I') = 0$ and $\mathrm{var}(P, I'') = 1$.

(9) return False

// Corollary 20 implies that $\mathrm{var}(P, I) \neq 1$.

In completely analogous manner as for the 0-Test, we can estimate the cost for the 1-Test:

Corollary 21. *Let $I = (a, b)$ be an interval, let $m(I)[\epsilon]$ be the multipoint defined in the 1-Test, and let $\lambda := \max\{|P(x)| \; ; \; x \in m(I)[\epsilon]\}$. Then, the 1-Test applied to I requires*

$$\tilde{O}(n(n + \tau_P + n \log M(a) + n \log M(b) + \log M(\min(|P(a)|, |P(b)|, \lambda)^{-1}))) \tag{14}$$

bit operations using approximations of the coefficients of P and the endpoints of I of quality $O(n + \tau_P + n \log M(a) + n \log M(b) + \log M(\min(|P(a)|, |P(b)|, \lambda)^{-1}))$.

2.5 Useful Inequalities

Lemma 22. *Let $p = \sum_{0 \leq i \leq n} p_i x^i = p_n \prod_{1 \leq i \leq n} (x - z_i) \in \mathbb{R}[z]$. Then,*

$$\mathrm{Mea}(p) \leq \|p\|_2 \leq \|p\|_1 \leq (n+1) 2^{\tau_p} \tag{15}$$

$$\sigma_p \geq \sqrt{|\mathrm{Disc}(p)|} \, \|p\|_2^{-n+1} \, n^{-(n+2)/2} \tag{16}$$

$$|\mathrm{Disc}(p)| \leq n^n (\mathrm{Mea}(p))^{2n-2} \leq n^n \, \|p\|_2^{2n-2} \tag{17}$$

$$\log |p'(z_i)| = O(\log n + \tau_p + n \log M(z_i)) \tag{18}$$

$$\sum_i \log M(p'(z_i)^{-1}) = O(n\tau_p + n^2 + n \log \mathrm{Mea}(p) + |\log \mathrm{Disc}(p)|^{-1}) \tag{19}$$

$$\mathrm{Mea}(p(x - z_i)) \leq 2^{\tau_p} 2^{n+1} M(z_i)^n \tag{20}$$

$$\tau_p = O(n + \log \mathrm{Mea}(p)). \tag{21}$$

Proof. [46, Lemma 4.14] establishes (15). [46, Corollary 6.29] establishes (16) and (17). For (18), observe

$$\log |p'(z_i)| = \log\Big(\sum_{1 \leq k \leq n} \big| p_k k z_i^{k-1} \big| \Big) \leq \log(n \cdot 2^{\tau_p} n M(z_i)^n) = O(\log n + \tau_p + n \log M(z_i)).$$

(19) follows from

$$\sum_i \log M(p'(z_i)^{-1}) = \log \prod_i M(p'(z_i)^{-1}) = \log \frac{\prod_i M(p'(z_i))}{\prod_i |p'(z_i)|}$$
$$= \log \frac{|p_n|^{n-2} \prod_i M(p'(z_i))}{|\mathrm{Disc}(p)|} = O(n\tau_p + n^2 + n \log \mathrm{Mea}(p) + \log |\mathrm{Disc}(p)|^{-1}).$$

For (20), we use $\mathrm{Mea}(p) \le \|p\|_1$ and $p(x - z_i) = \sum_{0 \le k \le n} (x^k \sum_{k \le j \le n} p_j \binom{j}{k} (-z_i)^{j-k})$, and hence,

$$\|p(x - z_i)\|_1 \le \sum_{0 \le k \le n} \sum_{k \le j \le n} p_j \binom{j}{k} M(z_i)^{j-k} \le 2^{\tau_p} M(z_i)^n \sum_{0 \le j \le n} \sum_{k \le j} \binom{j}{k}$$
$$\le 2^{\tau_p} M(z_i)^n \sum_{0 \le j \le n} 2^j \le 2^{\tau_p} M(z_i)^n 2^{n+1}.$$

For (21), we first recall that $\tau_P = \log M(\|p\|_\infty)$. The coefficient p_i is given by

$$p_i = p_n \cdot \sum_{I \subseteq \{1,\ldots,n\}, |I| = n-i} \prod_{i \in I} z_i.$$

Thus

$$|p_i| \le |p_n| \binom{n}{n-i} \frac{\mathrm{Mea}(p)}{|p_n|} \le 2^n \mathrm{Mea}(p).$$

□

3 The Algorithm

We are now ready for our algorithm ANewDsc[13] for isolating the real roots of P. We maintain a list $\mathcal{A}$ of active intervals,[14] a list $\mathcal{O}$ of isolating intervals, and the invariant that the intervals in $\mathcal{O}$ are isolating and that each real root of P is contained in either an active or an isolating interval. We initialize $\mathcal{O}$ to the empty set and $\mathcal{A}$ to the interval $\mathcal{I} = (-2^\Gamma, 2^\Gamma)$, where $\Gamma = 2^\gamma$ is defined as in (4). This interval contains all real roots of P. Our actual initialization procedure is more complicated, see Section 3.1, but this is irrelevant for the high level introduction to the algorithm.

In each iteration, we work on one of the active intervals, say I. We first apply the 0-Test and the 1-Test to I; see Section 2.4 for a discussion of these tests. If the 0-Test succeeds, we discard I. This is safe, as a successful 0-Test implies that I contains no real root. If the 1-Test succeeds, we add I to the set of isolating intervals. This is safe, as a successful 1-Test implies that I contains exactly one real root. If neither 0- or 1-Test succeeds, we need to subdivide I.

Classical bisection divides I into two equal or nearly equal sized subintervals. This works fine, if the roots contained in I spread out nicely, as then a small number of subdivision steps suffices to separate the roots contained in I. This works poorly if the roots contained in I form a cluster of nearby roots, as then a larger number of subdivision steps are needed until I is shrunk to an interval whose width is about the diameter of the cluster.

[13] Our algorithm is an *approximate arithmetic* variant of the algorithm NewDsc presented in [36]. NewDsc combines the classical Descartes method and Newton iteration. It uses exact rational arithmetic and only applies to polynomials with rational coefficients. Pronounce ANewDsc as either "approximate arithmetic Newton-Descartes" or "a new Descartes".

[14] In fact, $\mathcal{A}$ is a list of pairs (I, N_I), where I is an interval and $N_I \in \mathbb{N}$ a power of two. For the high level introduction, the reader may think of $\mathcal{A}$ of a list of intervals only.

In the presence of a cluster $\mathcal{C}$ of roots (i.e., a set of $k := |\mathcal{C}| \geq 2$ nearby roots that are "well separated" from all other roots), straight bisection converges only linearly, and it is much more efficient to obtain a good approximation of $\mathcal{C}$ by using Newton iteration. More precisely, if we consider a point ξ, whose distance d to the cluster $\mathcal{C}$ is considerably larger than the diameter of the cluster, and whose distance to all remaining roots is considerably larger than d, then the distance from the point

$$\xi' := \xi - k \cdot \frac{P(\xi)}{P'(\xi)} \tag{22}$$

to the cluster $\mathcal{C}$ is much smaller than the distance from ξ to $\mathcal{C}$.[15]. The distance d' of ξ' to the cluster is approximately d^2 if $d < 1$. Thus, we can achieve quadratic convergence to the cluster $\mathcal{C}$ by iteratively applying (22). Unfortunately, when running the subdivision algorithm, we neither know whether there actually exists a cluster $\mathcal{C}$ nor do we know its size or diameter. Hence, the challenge is to make the above insight applicable to a computational approach.

We overcome these difficulties as follows. First, we estimate k. For this, we consider two choices for ξ, say ξ_1 and ξ_2. Let ξ_i', $i = 1, 2$, be the Newton iterates. For the correct value of k, we should have $\xi_1' \approx \xi_2'$. Conversely, we can estimate k by solving $\xi_1' = \xi_2'$ for k. Secondly, we use quadratic interval refinement [1]. With every active interval $I = (a, b)$, we maintain a number $N_I = 2^{2^{n_I}}$, where $n_I \geq 1$ is an integer. We call n_I the *level* of interval I. We hope to refine I to an interval $I' = (a', b')$ of width $w(I)/N_I$. We compute candidates for the endpoints of I' using Newton iteration, that is, we compute a point inside I' and then obtain the endpoints of I' by rounding. We apply the 0-Test to (a, a') and to (b', b). If both 0-Tests succeed, we add (I', N_I^2) to the set of active intervals. Observe that, in a regime of quadratic convergence, the next Newton iteration should refine I' to an interval of width $w(I')/N_I^2$. If we fail to identify I', we bisect I and add both subintervals to the list of active intervals (with N_I replaced by $\max(4, \sqrt{N_I})$).

The details of the Newton step are discussed in Section 3.2, where we introduce the Newton-Test and the Boundary-Test. The Boundary-Test treats the special case that the subinterval I' containing all roots in I shares an endpoint with I, and there are roots outside I and close to I.

There is one more ingredient to the algorithm. We need to guarantee that P is large at interval endpoints. Therefore, instead of determining interval endpoints as described above, we instead take an admissible point chosen from an appropriate multipoint.

We next give the details of the algorithm ANewDsc:

Algorithm: ANewDsc
Input: A polynomial $P(x)$ as in (2).

Output: Disjoint isolating intervals $I_1, \ldots, I_m$ for all real roots of P with $\mathrm{var}(P, I_j) = 1$ for all $j = 1, \ldots, n$.

(1) $\mathcal{A} := \{(\mathcal{I}_k, 4)\}_{k=0,\ldots,2\gamma+2}$, with $\mathcal{I}_k$ as computed by Algorithm **Initialization**, and $\mathcal{O} := \emptyset$.

// Algorithm **Initialization** *is defined in Section 3.1. It computes a set of open and*

[15] The following derivation gives intuition for the behavior of the Newton iteration. Consider $P(x) = (x-\alpha)^k g(x)$, where α is not a root of g, and consider the iteration $x_{n+1} = x_n - k\frac{P(x_n)}{P'(x_n)}$. Then,

$$\begin{aligned} x_{n+1} - \alpha &= x_n - \alpha - k\frac{(x_n-\alpha)^k g(x_n)}{k(x_n-\alpha)^{k-1}g(x_n) + (x_n-\alpha)^k g'(x_n)} \\ &= (x_n-\alpha)(1 - \frac{kg(x_n)}{kg(x_n) + (x_n-\alpha)g'(x_n)} = (x_n-\alpha)^2 \frac{g'(x_n)}{kg(x_n) + (x_n-\alpha)g'(x_n)}, \end{aligned}$$

and hence, we have quadratic convergence in an interval around α.

disjoint intervals $\mathcal{I}_k$, such that all real roots of P are covered by the union of these intervals. We remark that ANewDsc *works for any set of intervals with this property, however, the choice of Section 3.1 simplifies the complexity analysis of the algorithm.*

(2) while $\mathcal{A} \neq \emptyset$ do

(2.1) Choose an arbitrary pair (I, N_I) from $\mathcal{A}$, with $I = (a, b)$, and remove (I, N_I) from $\mathcal{A}$

(2.2) If the algorithm **0-Test** (with input P and I) returns True, then go to Step (2.1)

// In this case, we have I contains no root of P, and thus I can be discarded.

(2.3) If the algorithm **1-Test** (with input P and I) returns True and an interval I', then add I' to $\mathcal{O}$ and go to Step (2.1)

// If the **1-Test** *returns an interval I', then I' is isolating and contains all roots of P that are contained in I. Hence, we can store I' as isolating and discard $I \backslash I'$.*

(2.4) If the algorithm **Boundary-Test** or the algorithm **Newton-Test** returns True and an interval I', then add $(I', N_{I'}) := (I', N_I^2)$ to $\mathcal{A}$, and go to Step (2.1)

(quadratic step)

// The **Boundary-Test** *and the* **Newton-Test** *are defined in Section 3.2. The interval I' returned by one of these tests contains all roots that are contained in I, and thus we can proceed with I'. Further notice that $\frac{w(I)}{8N_I} \leq w(I') \leq \frac{w(I)}{N_I}$.*

(2.5) Compute an admissible point $m^* \in m(I)[\frac{w(I)}{2^{\lceil 2+\log n \rceil}}]$ using the algorithm **Admissible Point** and add $(I', N_{I'})$ and $(I'', N_{I''})$ to $\mathcal{A}$, where $I' = (a, m^*)$ and $I'' = (m^*, b)$, and $N_{I'} := N_{I''} := \max(4, \sqrt{N_I})$.

(linear step)

// This step correspond to the classical bisection step, where the interval I is split into two equally sized subintervals. Here, we make sure that $|P|$ takes a reasonably large value at the splitting point. In addition, we have $\frac{w(I)}{4} \leq \min(w(I'), w(I'')) \leq \max(w(I'), w(I''))\frac{3w(I)}{4}$.

(3) return $\mathcal{O}$

If we succeed in Step (2.4), we say that the subdivision step from I to I' is *quadratic*. In a *linear* step, we just split I into two intervals of approximately the same size (i.e., of size in between $\frac{w(I)}{4}$ and $\frac{3w(I)}{4}$). From the definitions of our tests, the exactness of the algorithm follows immediately. In addition, since any interval I of width $w(I) < \frac{\sigma_P}{2}$ satisfies $\mathrm{var}(P, I) = 0$ or $\mathrm{var}(P, I) = 1$, either the 0-Test or the 1-Test succeeds for I. This proves termination (i.e., Step (2.2) or Step (2.3) succeeds) because, in each iteration, an interval I is replaced by intervals of width less than or equal to $\frac{3w(I)}{4}$.

3.1 Initialization

Certainly, the most straight-forward initialization is to start with the interval $\mathcal{I} = (-2^\Gamma, 2^\Gamma)$. In fact, this is also what we recommend doing in an actual implementation. However, in order to simplify the analysis of our algorithm, we proceed slightly differently. We first split $\mathcal{I}$ into disjoint intervals $\mathcal{I}_k = (s_k^*, s_{k+1}^*)$, with $k = 0, \ldots, 2 \cdot \gamma + 2$ and $\gamma = \log \Gamma$, such that for each interval, P is

large at the endpoints of the interval, and $\log M(x)$ is essentially constant within the interval. More precisely, the following conditions are fulfilled for all k:

$$\min(|P(s_k^*)|, |P(s_{k+1}^*)|) > 2^{-8n\log n}, \quad \text{and} \tag{23}$$
$$\max_{x\in\mathcal{I}_k} \log M(x) \le 2 \cdot (1 + \min_{x\in\mathcal{I}_k} \log M(x)).$$

The intervals $(-2^{2^\gamma}, -2^{2^{\gamma-1}})$, $(-2^{2^{\gamma-1}}, -2^{2^{\gamma-2}})$, $\ldots$, $(-2^{2^0}, 0)$, $(0, +2^{2^0})$, $\ldots$, $(-2^{2^{\gamma-1}}, -2^{2^\gamma})$ satisfy the second condition. In order to also satisfy the first, consider the points

$$s_k := \begin{cases} -2^{2^{\gamma-k}} & \text{for } k = 0, \ldots, \gamma \\ 0 & \text{for } k = \gamma+1 \\ +2^{2^{k-\gamma}} & \text{for } k = \gamma+2, \ldots, 2\gamma+2 \end{cases} \tag{24}$$

and corresponding multipoints $M_k := s_k[2^{-\lceil 2\log n\rceil}]$. In M_k, there exists at least one point with distance at least $\frac{1}{8n}$ or larger to all roots of P. Thus, $|P(s_k^*)| \ge |P_n| \cdot \left(\frac{1}{8n}\right)^n \ge 2^{-4n-n\log n} > 2^{-8n\log n}$, where s_k^* is an admissible point in M_k.

Algorithm: Initialization
Input: A polynomial $P(x)$ as in (2) and a γ as defined in (4).

Output: Disjoint open intervals $\mathcal{I}_k := (s_k^*, s_{k+1}^*)$, with $k = 0, \ldots, 2\gamma+2$, such that $\bigcup_k \mathcal{I}_k$ contains all real roots of P and the condition in (23) is fulfilled.

- For $k = 0, \ldots, 2\gamma+2$, define s_k as in (24).
- For $k = 0, \ldots, 2\gamma+2$, compute an admissible point $s_k^* \in M_k := s_k[2^{-\lceil 2\log n\rceil}]$ using the algorithm **Admissible Point**.
- Return the intervals

$$\mathcal{I}_k := (s_k^*, s_{k+1}^*) \quad \text{for } k = 0, \ldots, 2\gamma+2. \tag{25}$$

It is easy to check that the second condition in (23) is fulfilled for the intervals computed by the above algorithm.

3.2 The Newton-Test and the Boundary-Test

The Newton-Test and the Boundary-Test are the key to quadratic convergence. The Newton-test receives an interval $I = (a, b) \subseteq \mathcal{I}$ and an integer $N_I = 2^{2^{n_I}}$, where $n_I \ge 1$ is an integer. In case of success, the test returns an interval I' with $\frac{w(I)}{8N_I} \le w(I') \le \frac{w(I)}{N_I}$ that contains all roots that are contained in I. Success is guaranteed if there is a subinterval J of I of width at most $2^{-13} \cdot \frac{w(I)}{N_I}$ whose one-circle region contains all roots that are contained in the one-circle region of I and if the disk with radius $2^{\log n+10} \cdot N_I \cdot w(I)$ and center $m(I)$ contains no further root of P, see Lemma 23 for a precise statement. Informally speaking, the Newton-Test is guaranteed to succeed if the roots in I cluster in a subinterval significantly shorter than $w(I)/N_I$, and roots outside I are far away from I. In the following description of the Newton-Test, we inserted comments that explain the rationale behind our choices. For this rationale, we assume the existence of a cluster $\mathcal{C}$ of k roots centered at some point $\xi \in I$ with diameter $d(\mathcal{C}) \ll w(I)$ and that there exists no other root in a large neighborhood of the one-circle region $\Delta(I)$ of I. The formal justification for the Newton-Test will be given in Lemma 23.

Algorithm: Newton-Test

Input: An Interval $I = (a, b) \subset \mathbb{R}$, an integer $N_I = 2^{2^{n_I}}$ with $n_I \in \mathbb{N}$, and a polynomial $P \in \mathbb{R}[x]$ as defined in (2)

Output: True or False. In case of True, it also returns an interval $I' \subset I$, with $\frac{w(I)}{8N_I} \leq w(I') \leq \frac{w(I)}{N_I}$, that contains all real roots of P that are contained in I.

(1) Let $\xi_1 := a + \frac{1}{4} \cdot w(I)$, $\xi_2 := a + \frac{1}{2} \cdot w(I)$, $\xi_3 := a + \frac{3}{4} \cdot w(I)$, and $\epsilon := 2^{-\lceil 5 + \log n \rceil}$, and compute admissible points

$$\xi_j^* \in \xi_j[\epsilon \cdot w(I)], \tag{26}$$

for $j = 1, 2, 3$, using the Algorithm **Admissible Point** from Section 2.2.

// At least two of the three points ξ_j (say ξ_1 and ξ_2) have a distance from $\mathcal{C}$ that is large compared to the diameter of $\mathcal{C}$. In addition, their distances to all remaining roots are also large, and thus the points $\xi_1' := \xi_1 - k \cdot v_1$ and $\xi_2' := \xi_2 - k \cdot v_2$, with $v_{j_1} := \frac{P(\xi_{j_1}^*)}{P'(\xi_{j_1}^*)}$ and $v_{j_2} := \frac{P(\xi_{j_2}^*)}{P'(\xi_{j_2}^*)}$, obtained from considering one Newton step, with $\xi = \xi_{j_1}^*$ and $\xi = \xi_{j_2}^*$, have much smaller distances to $\mathcal{C}$ than the points ξ_1 and ξ_2. Notice that k is not known to the algorithm at this point.

(2) For each of the three distinct pairs (j_1, j_2) of indices $j_1, j_2 \in \{1, 2, 3\}$, with $j_1 < j_2$, do:

(2.1) For $L = 2, 4, 8, \ldots$, compute approximations A_{j_1}, A_{j_2}, A'_{j_1}, and A'_{j_2} of $P(\xi_{j_1}^*)$, $P(\xi_{j_2}^*)$, $P'(\xi_{j_1}^*)$, and $P'(\xi_{j_2}^*)$ of quality L, respectively, until, for some $L = L_1$, it holds that

$$\frac{|A_{j_1}| - 2^{-L}}{|A'_{j_1}| + 2^{-L}} > w(I) \quad \text{or} \quad \frac{|A_{j_2}| - 2^{-L}}{|A'_{j_2}| + 2^{-L}} > w(I) \tag{27}$$

or

$$|A_{j_1}|, |A_{j_2}| > 2^{-L+1} \text{ and } |A'_{j_1}|, |A'_{j_2}| > 2^{-L+1} \tag{28}$$

Then, if (27) holds, discard the pair (j_1, j_2). Otherwise, proceed with Step (2.2).

// The values $\tilde{v}_{j_1} := \frac{A_{j_1}}{A'_{j_1}}$ and $\tilde{v}_{j_2} := \frac{A_{j_2}}{A'_{j_2}}$ are approximations of v_{j_1} and v_{j_2}, respectively. Condition (27) implies that either $|v_{j_1}| > w(I)$ or $|v_{j_2}| > w(I)$. The proof of Lemma 23 shows that the existence of $\mathcal{C}$ implies that there is a pair (j_1, j_2), for which v_{j_1} and v_{j_2} have distance $\ll \frac{w(I)}{N_I}$ to $\mathcal{C}$ and $|v_{j_1}|, |v_{j_2}| < \frac{w(I)}{k} \leq w(I)$. Hence, such a pair cannot be discarded in step (2.1). Further notice that $L_1 \leq L_1^* := \log \max\{M(|P(\xi_{j_1}^*)|^{-1}), M(|P(\xi_{j_1}^*)|^{-1})\} + \log M(w(I)) + 3$. Namely, for $L \geq L_1^*$, either (28) holds or

$$\frac{|A_j| - 2^{-L}}{|A'_j| + 2^{-L}} > \frac{|P(\xi_j^*)| - 2^{-L+1}}{2^{-L+1} + 2^{-L}} \geq \frac{8 \cdot M(w(I)) \cdot 2^{-L} - 2^{-L+1}}{3 \cdot 2^{-L}} \geq w(I)$$

for $j = j_1$ or $j = j_2$. In addition, if (28) holds for $L = L_1$, then (28) also holds for any $L \geq 2 \cdot L_1$. This also implies that, for $j = j_1, j_2$, $|\tilde{v}_j|$ is a 4-approximation of $|v_j|$, that is, $|\tilde{v}_j|$ differs from $|v_j|$ by a factor in $(1/4, 4)$. Since $\frac{|A_j| + 2^{-L}}{|A'_j| - 2^{-L}}$ differs from $\tilde{v}_j$ by a factor in $(1, 4)$, it follows that $|\tilde{v}_j| < 64 w(I)$ for any $L \geq 2L_1$ and $j = j_1, j_2$.

(2.2) For $L = 2 \cdot L_1, 4 \cdot L_1, 8 \cdot L_1, \dots$, compute approximations A_{j_1}, A_{j_2}, A'_{j_1}, and A'_{j_2} of $P(\xi^*_{j_1})$, $P(\xi^*_{j_2})$, $P'(\xi^*_{j_1})$, and $P'(\xi^*_{j_2})$ of quality L, respectively, until, for some $L = L_2$, it holds that

$$\delta_{j_1}, \delta_{j_2} < \min\left\{\frac{w(I)}{2^5 \cdot n}, \frac{w(I)}{2^{14} \cdot N_I}\right\}, \text{ with } \delta_{j_1} := \frac{|A_{j_1}| + |A'_{j_1}|}{2^{L-2} \cdot |A'_{j_1}|^2}, \ \delta_{j_2} := \frac{|A_{j_2}| + |A'_{j_2}|}{2^{L-2} \cdot |A'_{j_2}|^2}. \tag{29}$$

If the condition

$$|\tilde{v}_{j_1} - \tilde{v}_{j_2}| + \delta_{j_1} + \delta_{j_2} \geq \frac{w(I)}{n}, \quad \text{with} \quad \tilde{v}_{j_1} := \frac{A_{j_1}}{A'_{j_1}} \quad \text{and} \quad \tilde{v}_{j_2} := \frac{A_{j_2}}{A'_{j_2}}, \tag{30}$$

is fulfilled, then proceed with Step (2.3). Otherwise, discard the pair (j_1, j_2).

// A straight-forward computation shows that $|\tilde{v}_{j_1} - v_{j_1}| < \delta_{j_1}$ and $|\tilde{v}_{j_2} - v_{j_2}| < \delta_{j_2}$, where we use that $|A_j|$ and $|A'_j|$ are relative 2-approximations of $|P(\xi^*_j)|$ and $|P'(\xi^*_j)|$, for $j = j_1, j_2$, respectively. Hence, if condition (30) does not hold, then $|v_{j_1} - v_{j_2}| < \frac{w(I)}{n}$. Due to the proof of Lemma 23, the existence of $\mathcal{C}$ yields that $|v_{j_1} - v_{j_2}| > \frac{w(I)}{k}$ for some pair (j_1, j_2), and thus (30) must be fulfilled for such a pair. Notice that the inequality in (29) holds for $L = L_2$, with an L_2 of size

$$\begin{aligned} L_2 &= O(L_1 + \log \max(M((32n)/w(I)), N_I)) \\ &= O(\log \max\{n, M(|P(\xi^*_{j_1})|^{-1}), M(|P(\xi^*_{j_2})|^{-1}), M(w(I)), M(w(I)^{-1}), N_I\}), \end{aligned}$$

where we use that $\tilde{v}_j < 64w(I)$ and $2^L \cdot |A'_j| \geq 2^L \cdot \frac{1}{4} \cdot 2^{-L_1+1} = 2^{L-L_1-1}$ for all $L \geq 2 \cdot L_1$ and $j = j_1, j_2$.

(2.3) Compute

$$\tilde{\lambda}_{j_1,j_2} := \xi^*_{j_1} + \frac{\xi^*_{j_2} - \xi^*_{j_1}}{\tilde{v}_{j_1} - \tilde{v}_{j_2}} \cdot \tilde{v}_{j_1} \tag{31}$$

If $\tilde{\lambda}_{j_1,j_2} \notin \bar{I} = [a, b]$, discard the pair (j_1, j_2). Otherwise, compute $\ell_{j_1,j_2} := \lfloor \frac{\tilde{\lambda}_{j_1,j_2} - a}{w(I)/(4N_I)} \rfloor$, which is an integer contained in $\{0, \dots, 4N_I\}$. Further define

$$I_{j_1,j_2} := (a_{j_1,j_2}, b_{j_1,j_2}) := (a + \max(0, \ell_{j_1,j_2} - 1) \cdot \frac{w(I)}{4N_I}, a + \min(4N_I, \ell_{j_1,j_2} + 2) \cdot \frac{w(I)}{4N_I}).$$

If $a_{j_1,j_2} = a$, set $a^*_{j_1,j_2} := a$, and if $b_{j_1,j_2} = b$, set $b_{j_1,j_2} := b$. For all other values for a_{j_1,j_2} and b_{j_1,j_2}, use Algorithm **Admissible Point** from Section 2.2 to compute admissible points

$$a^*_{j_1,j_2} \in a_{j_1,j_2}[\epsilon \cdot \frac{w(I)}{N_I}] \quad \text{and} \quad b^*_{j_1,j_2} \in b_{j_1,j_2}[\epsilon \cdot \frac{w(I)}{N_I}]. \tag{32}$$

Define $I^*_{j_1,j_2} := (a^*_{j_1,j_2}, b^*_{j_1,j_2})$.

// The Newton iteration (22) with $\xi = \xi^*_j$ for a k-fold root produces $\xi'_j = \xi^*_j - kv_j$. Equating $\xi'_{j_1} = \xi'_{j_2}$ yields $-k = \frac{\xi^*_{j_2} - \xi^*_{j_1}}{v_{j_1} - v_{j_2}}$. Then, ξ'_{j_1} and ξ'_{j_2} are given by

$$\lambda_{j_1,j_2} := \xi^*_{j_1} + \frac{\xi^*_{j_2} - \xi^*_{j_1}}{v_{j_1} - v_{j_2}} \cdot v_{j_1}. \tag{33}$$

A straight-forward computation shows that $|\tilde{\lambda}_{j_1,j_2} - \lambda_{j_1,j_2}| < \frac{w(I)}{32N_I}$*, where we use inequality (29) and the fact that* $|\tilde{v}_{j_1}|, |\tilde{v}_{j_2}| < 64w(I)$*, and* $|\tilde{v}_{j_1}|, |\tilde{v}_{j_2}| > \frac{w(I)}{2n}$ *due to (30). If* $\tilde{\lambda}_{j_1,j_2}$ *is contained in* I*, we (conceptually) subdivide* I *into* $4N_I$ *subintervals and determine the subinterval that contains* $\tilde{\lambda}_{j_1,j_2}$*. Extending the interval on both sides by* $\frac{w(I)}{4N_I}$ *yields an interval* I_{j_1,j_2}*, which contains* λ_{j_1,j_2}*. Finally, replacing the endpoints* a_{j_1,j_2} *and* b_{j_1,j_2} *by nearby admissible points yields an interval* $I^*_{j_1,j_2}$ *with* $\frac{w(I)}{8n} \leq w(I^*_{j_1,j_2}) \leq \frac{w(I)}{n}$*. Lemma 23 then shows that the existence of* $\mathcal{C}$ *guarantees that* $I^*_{j_1,j_2}$ *contains all roots of* P *that are contained in* I*.*

(2.4) Run the **0-Test** from Section 2.4.1 with input $I'_\ell := (a, a^*_{j_1,j_2})$ and $I'_r := (b^*_{j_1,j_2}, b)$. If it succeeds on both intervals, return **True** and the interval $I' := I^*_{j_1,j_2}$. Otherwise, discard the pair (j_1, j_2).

// For intervals I'_ℓ *or* I'_r*, which are empty (i.e.,* $I'_\ell = (a, a)$ *or* $I'_r = (b, b)$*), nothing needs to be done. If the* **0-Test** *succeeds on* I'_ℓ *as well as on* I'_r*, then neither interval contains a root of* P*. Hence,* $I^*_{j_1,j_2}$ *contains all roots of* P *that are contained in* I*.*

(3) If each of the three pairs (j_1, j_2) is discarded in one of the above steps, return **False**.

We next derive a sufficient condition for the success of the Newton-Test.

Lemma 23. *Let* $I = (a, b)$ *be an interval,* $N_I = 2^{2^{n_I}}$ *with* $n_I \in \mathbb{Z}_{\geq 1}$*, and* $J = (c, d) \subseteq I$ *be a subinterval of width* $w(J) \leq 2^{-13} \cdot \frac{w(I)}{N_I}$*. Suppose that the one-circle region of* $\Delta(J)$ *contains* k *roots* $z_1, \ldots, z_k$ *of* P*, with* $k \geq 1$*, and that the disk with radius* $2^{\log n+10} \cdot N_I \cdot w(I)$ *and center* $m(I)$ *contains no further root of* P*. Then, the Newton-Test succeeds.*

Proof. We first show that, for at least two of the three points ξ^*_j, $j = 1, 2, 3$, the inequality $\left| m(J) - (\xi^*_j - k \cdot \frac{P(\xi^*_j)}{P'(\xi^*_j)}) \right| < \frac{w(I)}{128N_I}$ holds: There exist at least two points (say $\xi := \xi^*_{j_1}$ and $\bar{\xi} := \xi^*_{j_2}$ with $j_1 < j_2$) whose distances to any root from $z_1, \ldots, z_k$ are larger than $\frac{|\xi - \bar{\xi}|}{2} - \frac{w(J)}{2} > \frac{3}{32}w(I) - \frac{1}{2}w(J) \geq 512N_I w(J)$. In addition, the distances to any of the remaining roots $z_{k+1}, \ldots, z_n$ are larger than $2^{10}nN_I w(I) - w(I) \geq 512 \cdot nN_I w(I)$. Hence, with $m := m(J)$, it follows that

$$\begin{aligned}
\left| \frac{1}{k} \cdot \frac{(\xi - m)P'(\xi)}{P(\xi)} - 1 \right| &= \left| \frac{1}{k} \sum_{i=1}^{k} \frac{\xi - m}{\xi - z_i} + \frac{1}{k} \sum_{i>k} \frac{\xi - m}{\xi - z_i} - 1 \right| = \frac{1}{k} \left| \sum_{i=1}^{k} \frac{z_i - m}{\xi - z_i} + \sum_{i>k} \frac{\xi - m}{\xi - z_i} \right| \\
&\leq \frac{1}{k} \sum_{i=1}^{k} \frac{|z_i - m|}{|\xi - z_i|} + \frac{1}{k} \sum_{i>k} \frac{|\xi - m|}{|\xi - z_i|} < \frac{w(J)}{512nN_I w(J)} + \frac{(n-k) \cdot w(I)}{512knN_I w(I)} \\
&\leq \frac{1}{256N_I},
\end{aligned}$$

where we used that $\frac{P'(\xi)}{P(\xi)} = \sum_{i=1}^{n} (\xi - z_i)^{-1}$. This yields the existence of an $\epsilon \in \mathbb{R}$ with $|\epsilon| < \frac{1}{256N_I} \leq \frac{1}{1024}$ and $\frac{1}{k} \cdot \frac{(\xi - m)P'(\xi)}{P(\xi)} = 1 + \epsilon$. We can now derive the following bound on the

distance between the approximation $\xi' = \xi - k \cdot \frac{P(\xi)}{P'(\xi)}$ obtained by the Newton iteration and m:

$$|m - \xi'| = |m - \xi| \cdot \left| 1 - \frac{1}{\frac{1}{k} \cdot \frac{(\xi - m)P'(\xi)}{P(\xi)}} \right| = |m - \xi| \cdot \left| 1 - \frac{1}{1+\epsilon} \right| = \left| \frac{\epsilon \cdot (m - \xi)}{1+\epsilon} \right| < \frac{w(I)}{128 N_I}.$$

In a completely analogous manner, we show that $\left| \bar{\xi} - k \cdot \frac{P(\bar{\xi})}{P'(\bar{\xi})} - m \right| < \frac{w(I)}{128 N_I}$.

Let $v_{j_1} = \frac{P(\xi)}{P'(\xi)}$ and $v_{j_2} = \frac{P(\bar{\xi})}{P'(\bar{\xi})}$ be defined as in the Newton-Test. Then, from the above considerations, it follows that $|(\xi - k \cdot v_{j_1}) - (\bar{\xi} - k \cdot v_{j_2})| < \frac{w(I)}{64 N_I}$. Hence, since $|\xi - \bar{\xi}| > \frac{3w(I)}{16}$ and $1 \leq k \leq n$, we must have $|v_{j_1} - v_{j_2}| > \frac{w(I)}{8k}$. Furthermore, it holds that $|k \cdot v_{j_1}| < w(I)$ since, otherwise, the point $\xi - k \cdot v_{j_1}$ is not contained in $(\xi - w(I), \xi + w(I))$, which contradicts the fact that $|\xi - k \cdot v_{j_1} - m| < \frac{w(I)}{128 N_I)}$ and $m \in I$. An analogous argument yields that $|k \cdot v_{j_2}| < w(I)$. Hence, none of the two inequalities in (27) are fulfilled, whereas the inequality in (30) must hold. In the next step, we show that $\lambda := \lambda_{j_1, j_2}$ as defined in (33) is actually a good approximation of $\xi - k \cdot v_{j_1}$: There exist ϵ and $\bar{\epsilon}$, both of magnitude less than $\frac{w(I)}{128 N_I}$, such that $\xi - k \cdot v_{j_1} = m + \epsilon$ and $\bar{\xi} - k \cdot v_{j_2} = m + \bar{\epsilon}$. This yields

$$\lambda = \xi + \frac{\bar{\xi} - \xi}{v_{j_1} - v_{j_2}} \cdot v_{j_1} = \xi + \left(\frac{\bar{\epsilon} - \epsilon + k \cdot (v_{j_2} - v_{j_1})}{v_{j_1} - v_{j_2}} \right) \cdot v_{j_1} = \xi + k \cdot v_{j_1} + \frac{(\bar{\epsilon} - \epsilon) v_{j_1}}{v_{j_1} - v_{j_2}}.$$

The absolute value of the fraction on the right side is smaller than $\frac{w(I) \cdot |v_{j_1}|}{64 N_I} \cdot \frac{8k}{w(I)} \leq \frac{w(I)}{8 N_I}$, and thus $|\xi - k \cdot v_{j_1} - \lambda| < \frac{w(I)}{8 N_I}$. Hence, with $\tilde{\lambda}_{j_1, j_2}$ as defined in (31), we have

$$|m - \tilde{\lambda}_{j_1, j_2}| \leq |m - (\xi - k \cdot v_{j_1})| + |(\xi - k \cdot v_{j_1}) - \lambda| + |\lambda - \tilde{\lambda}_{j_1, j_2}| < \frac{w(I)}{128 N_I} + \frac{w(I)}{8 N_I} + \frac{w(I)}{32 N_I} < \frac{3w(I)}{16 N_I}.$$

From the definition of the interval I_{j_1, j_2}, we conclude that $J \subseteq I_{j_1, j_2}$. Furthermore, each endpoint of I_{j_1, j_2} is either an endpoint of I, or its distance to both endpoints of J is larger than $\frac{w(I)}{16 N_I} - \frac{w(J)}{2} > \frac{w(I)}{32 N_I} > \frac{w(J)}{2}$. This shows that the interval $I' = I^*_{j_1, j_2}$ contains J. Hence, the Newton-Test succeeds since the one-circle regions of I'_ℓ and I'_r contain no roots of P. □

The Newton-Test is our main tool to speed up convergence to clusters of roots without actually knowing that there exists a cluster. However, there is one special case that has to be considered separately: Suppose that there exists a cluster $\mathcal{C} \subseteq \Delta(I)$ of roots whose center is close to one of the endpoints of I. If, in addition, $\mathcal{C}$ is not well separated from other roots that are located outside of $\Delta(I)$, then the above lemma does not apply. For this reason, we introduce the *Boundary-Test*, which checks for clusters near the endpoints of an interval I. Its input is the same as for the Newton-Test. In case of success, it either returns an interval $I' \subseteq I$, with $\frac{w(I)}{4 N_I} \leq w(I') \leq \frac{w(I)}{N_I}$, which contains all real roots that are contained in I, or it proves that I contains no root.

Algorithm: Boundary-Test

Input: An Interval $I = (a, b) \subset \mathbb{R}$, an integer $N_I = 2^{2^{n_I}}$ with $n_I \in \mathbb{N}$, and a polynomial $P \in \mathbb{R}[x]$ as defined in (2)

Output: True or False. In case of True, it also returns an interval $I' \subset I$, with $\frac{w(I)}{8 N_I} \leq w(I') \leq \frac{w(I)}{N_I}$, that contains all real roots of P that are contained in I.

(1) Let $m_\ell := a + \frac{w(I)}{2N_I}$, $m_r := b - \frac{w(I)}{2N_I}$, and $\epsilon := 2^{-\lceil 2+\log n \rceil}$. Use algorithm **Admissible Point** to compute admissible points

$$m_\ell^* \in m_\ell[\epsilon \cdot \frac{w(I)}{N_I}] \quad \text{and} \quad m_r^* \in m_r[\epsilon \cdot \frac{w(I)}{N_I}], \tag{34}$$

(2) If the 0-Test returns true for the interval $I_\ell := (m_\ell^*, b)$, then return (a, m_ℓ^*).

(3) If the 0-Test returns true for the interval $I_r := (a, m_r^*)$, then return (m_r^*, b).

(4) return False

Clearly, if all roots contained in $\Delta(I)$ have distance less than $\frac{w(I)}{4N_I}$ to one of the two endpoints of I, the Boundary-Test for I is successful, as the one-circle region of either I_ℓ or I_r contains no root of P.

4 Complexity Analysis

We bound the size of the subdivision tree in Section 4.1 and the bit complexity in Section 4.2.

4.1 Size of the Subdivision Tree

We use $\mathcal{T}$ to denote the subdivision forest which is induced by our algorithm ANewDsc. More precisely, in this forest, we have one tree for each interval $\mathcal{I}_k$, with $k = 0, \ldots, 2\log\Gamma + 2$, as defined in (25). Furthermore, an interval I' is a child of some $I \in \mathcal{T}$ if and only if it has been created by our algorithm when processing I. We have $\frac{w(I)}{8N_I} \le w(I') \le \frac{w(I)}{N_I}$ in a quadratic step and $\frac{1}{4} \cdot w(I) \le w(I') \le \frac{3}{4} \cdot w(I)$ in a linear step. An interval in $\mathcal{T}$ has two, one, or zero children. Intervals with zero children are called *terminal*. Those are precisely the intervals for which either the 0-Test or the 1-Test is successful. Since each interval $I \neq \mathcal{I}$ with $\mathrm{var}(P, I) \le 1$ is terminal, it follows that, for each non-terminal interval I, the one circle region $\Delta(I)$ contains at least one root and the two-circle region of I contains at least two roots of P. Thus, all non-terminal nodes have width larger than or equal to $\sigma_P/2$.

In order to estimate the size of $\mathcal{T}$, we estimate for each $\mathcal{I}_k$ the size of the tree $\mathcal{T}_k$ rooted at it. If $\mathcal{I}_k$ is terminal, $\mathcal{T}_k$ consists only of the root. So, assume that $\mathcal{I}_k$ is non-terminal. Call a non-terminal $I \in \mathcal{T}_k$ *splitting* if either I is the root of $\mathcal{T}_k$, or $\mathcal{M}(I') \neq \mathcal{M}(I)$ for all children I' of I (recall that $\mathcal{M}(I)$ denotes the set of roots of P contained in the one circle region $\Delta(I)$ of I), or if all children of I are terminal. By the argument in the preceding paragraph, $\mathcal{M}(I) \neq \emptyset$ for all splitting nodes. A splitting node I is called *strongly splitting* if there exists a root $z \in \mathcal{M}(I)$ that is not contained in any of the one-circle regions of its children. The number of splitting nodes in $\mathcal{T}_k$ is bounded by $2\,|\mathcal{M}_k|$ since there are at most $|\mathcal{M}_k|$ splitting nodes all of whose children are terminal, since at most $|\mathcal{M}_k| - 1$ splitting nodes all of whose children have a smaller set of roots in the one-circle region of the associated interval, and since there is one root. For any splitting node, consider the path of non-splitting nodes ending in it, and let $s_{\max}$ be the maximal length of such a path (including the splitting node at which the path ends and excluding the splitting node at which the path starts). Then, the number of non-terminal nodes in $\mathcal{T}_k$ is bounded by $1 + s_{\max} \cdot (2\,|\mathcal{M}_k| - 1)$, and the total number of non-terminal nodes in the subdivision forest is $O(\log\Gamma + n \cdot s_{\max})$. Hence, the same bound also applies to the number of all nodes in $\mathcal{T}$.

The remainder of this section is concerned with proving that

$$s_{\max} = O(\log n + \log(\Gamma + \log M(\sigma_P^{-1}))).$$

The proof consists of three parts.
(1) We first establish lower and upper bounds for the width of *all* (i.e., also for terminal) intervals $I \in \mathcal{T}$ and the corresponding numbers N_I (Lemma 24).
(2) We then study an abstract version of how interval sizes and interval levels develop in quadratic interval refinement (Lemma 25).
(3) In a third step, we then derive the bound on $s_{\max}$ (Lemma 26).

Lemma 24. *For each interval $I \in \mathcal{T} \backslash \mathcal{I}$, we have*

$$2^\Gamma \geq w(I) \geq 2^{-4\Gamma-6}\sigma_P^5 \quad \textit{and} \quad 4 \leq N_I \leq 2^{4(\Gamma+1)} \cdot \sigma_P^{-4}.$$

Proof. The inequalities $2^\Gamma \geq w(I)$ and $N_I \geq 4$ are trivial. For $N_I > 16$, there exist two ancestors (not necessarily parent and grandparent) J' and J of I, with $I \subseteq J' \subseteq J$, such that $w(I) \leq w(J')/N_{J'}$ and $w(J') \leq w(J)/N_J$, and $N_I = N_{J'}^2 = N_J^4$. Hence, it follows that J' is a non-terminal interval of width less than or equal to $2^\Gamma/N_J = 2^\Gamma N_I^{-1/4}$. Since each non-terminal interval has width $\sigma_P/2$ or more, the upper bound on N_I follows. For the claim on the width of I, we remark that the parent interval K of I has width $\sigma_P/2$ or more and that $\frac{w(K)}{8N_J} \leq w(I) \leq \frac{3}{4}w(K)$. □

We come to the evolution of interval sizes and levels in quadratic interval refinement. The following Lemma has been introduced in [36, Lemma 4] in a slightly weaker form:

Lemma 25. *Let w, $w' \in \mathbb{R}^+$ be two positive reals with $w > w'$, and let $m \in \mathbb{N}_{\geq 1}$ be a positive integer. We recursively define the sequence $(s_i)_{i\in\mathbb{N}_{\geq 1}} := ((x_i, n_i))_{i\in\mathbb{N}_{\geq 1}}$ as follows: Let $s_1 = (x_1, n_1) := (w, m)$, and*

$$s_{i+1} = (x_{i+1}, n_{i+1}) := \begin{cases} (\epsilon_i \cdot x_i, n_i + 1) \ \textit{with an } \epsilon_i \in [0, \frac{1}{N_i}], & \textit{if } \frac{x_i}{N_i} \geq w' \\ (\delta_i \cdot x_i, \max(1, n_i - 1)) \ \textit{with a } \delta_i \in [0, \frac{3}{4}], & \textit{if } \frac{x_i}{N_i} < w', \end{cases}$$

where $N_i := 2^{2^{n_i}}$ and $i \geq 1$. Then, the smallest index i_0 with $x_{i_0} \leq w'$ is bounded by $8(n_1 + \log\log\max(4, \frac{w}{w'}))$.

Proof. The proof is similar to the proof given in [36]. However, there are subtle differences, and hence, we give the full proof. We call an index i *strong* (**S**) if $x_i/N_i \geq w'$ and *weak* (**W**), otherwise. If $w/4 < w'$, then each $i \geq 1$ is weak, and thus, $i_0 \leq 6$ because of $(3/4)^5 < 1/4$.

So assume $w/4 \geq w'$ and let k' be the smallest weak index. We split the sequence $1, 2, \ldots, i_0$ into three parts, namely (1) the prefix $1, \ldots, k' - 1$ of strong indices, (2) the subsequence $k', \ldots, i_0 - 6$ starting with the first weak index and containing all indices but the last 6, and (3) the tail $i_0 - 5, \ldots, i_0$. The length of the tail is 6.

We will show that the length of the prefix of strong indices is bounded by $k \in \mathbb{N}_{\geq 1}$ where k is the unique integer with

$$2^{-2^{k+1}} < w'/w \leq 2^{-2^k}.$$

Then, $k \leq \log\log\frac{w}{w'}$. Intuitively, this holds since we square N_i in each strong step, and hence, after $O(\log\log w/w')$ strong steps we reach a situation where a single strong step guarantees that the next index is weak. In order to bound the second subsequence, we split it into subsubsequences of maximal length containing no two consecutive weak indices. We will show that the subsubsequences have length at most five and that each such subsubsequence (except for the last) has one more weak index than strong indices. Thus, the value of n at the end of a subsubsequence is one smaller than at the beginning of the subsubsequence, and hence, the number of subsubsequences is bounded by n_1. We turn to the bound on the length of the prefix of strong indices.

Claim 1: $k' \le k+1$.
Suppose that the first k indices are strong. Then, $x_{i+1} \le 2^{-2^{m+i}} x_i$ for $i = 1, \ldots, k$, and hence,

$$x_{k+1} \le w \cdot 2^{-(2^m + 2^{m+1} + \cdots + 2^{m+k-1})} = w \cdot 2^{-2^m(2^k-1)} \le 4w \cdot 2^{-2^{k+1}} < 4w',$$

and $n_{k+1} \ge 2$. Thus, $x_{k+1}/N_{k+1} < w'$, and $k+1$ is weak.

Let us next consider the subsequence $\mathcal{S} = k', k'+1, \ldots, i_0 - 6$.
Claim 2: $\mathcal{S}$ contains no subsequence of type **SS** or **SWSWS**.
Consider any weak index i followed by a strong index $i+1$. Then, $N_{i+2} \ge N_i$ and $x_{i+2} \le x_i$, and hence, $x_{i+2}/N_{i+2} \le x_i/N_i < w'$. Thus, $i+2$ is weak. Since $\mathcal{S}$ starts with a weak index, the first part of our claim follows. For the second part, assume that i, $i+2$ are strong, and $i+1$ and $i+3$ are weak. Then, $N_i = N_{i+2} = N_{i+4}$, $N_{i+1} = N_i^2$, $x_{i+1} \le x_i/N_i$, $x_{i+3} \le x_{i+2}/N_{i+2}$, $x_{i+4} < x_{i+3}$, $x_{i+3} < x_{i+1}$, and hence,

$$\frac{x_{i+4}}{N_{i+4}} < \frac{x_{i+2}}{N_{i+2}N_{i+4}} \le \frac{x_{i+1}}{N_{i+2}^2} = \frac{x_{i+1}}{N_{i+1}} < w'.$$

Thus, $i+4$ is weak.

Claim 3: If i is weak and $i \le i_0 - 6$, then $n_i \ge 2$.
Namely, if i is weak and $n_i = 1$, then $x_i/4 = x_i/N_i < w'$, and thus, $x_{i_0-1} < w'$ because $(3/4)^5 < 1/4$. This contradicts the definition of i_0.

We now partition the sequence $\mathcal{S}$ into maximal subsequences $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r$, such that each $\mathcal{S}_j$, $j = 1, \ldots, r$, contains no two consecutive weak elements. Then, according to our above results, each $\mathcal{S}_j$, with $j < r$, is of type **W**, **WSW**, or **WSWSW**. The last subsequence $\mathcal{S}_r$ is of type **W**, **WS**, **WSW**, **WSWS**, or **WSWSW**. Since $n_i \ge 2$ for all weak i with $i \le i_0 - 6$, the number n_i decreases by one after each S_j, with $j < r$. Thus, we must have $r \le n_1 + k' - 2$ since $n_{k'} = n_1 + k' - 1$, $n_{r-1} = n_{k'} - (r-1)$, and $n_{r-1} \ge 2$. Since the length of each $\mathcal{S}_j$ is bounded by 5, it follows that

$$i_0 = i_0 - 6 + 6 \le k' + 5r + 6 \le k' + 5(n_1 + k' - 2) + 6 \le 5(n_1 + k) + k + 2 < 8(n_1 + k).$$

□

We are now ready to derive an upper bound on $s_{\max}$.

Lemma 26. *The maximal length $s_{\max}$ of any path between splitting nodes is bounded by $O((\log n + \log(\Gamma + \log M(\sigma_P^{-1}))))$.*

Proof. Consider any path in the subdivision forest ending in a splitting node and otherwise containing only non-splitting nodes. Let $I_1 := (a_1, b_1) := I$ to $I_s = (a_s, b_s)$ be the corresponding sequence of intervals. Then, the one-circle regions $\Delta(I_j)$ of all intervals in the sequence contain exactly the same set of roots of P, and this set is non-empty. We show $s = O(\log n + \log(\Gamma + \log M(\sigma_P^{-1})))$. We split the sequence into three parts:

(1) Let $s_1 \in \{1, \ldots, s\}$ be the smallest index with $a_{s_1} \ne a_1$ and $b_{s_1} \ne b_1$. The first part consists of intervals I_1 to I_{s_1-1}. We may assume $a = a_1 = a_2 = \ldots = a_{s_1-1}$. We will show $s_1 = O(\log(\Gamma + \log M(\sigma_P^{-1}))$.
(2) Let $s_2 \ge s_1$ be minimal such that either $s_2 = s$ or $w(I_{s_2}) \le 2^{-13-\log n} w(I_{s_1})/N_{I_{s_2}}$. We will show $s_2 - s_1 = O(\log n + \log(\Gamma + \log M(\sigma_P^{-1}))$. The second part consists of intervals I_{s_1} to intervals I_{s_2-1}.
(3) The third part consists of the remaining intervals I_{s_2} to I_s. If $s_2 = s$, this part consists of a single interval. If $s_s < s$, we have $w(I_j) \le 2^{-13-\log n} w(I_{s_1})/N_{I_j}$ for all $j \ge s_2$. If I_{j+1} comes from I_j by a linear step, this is obvious because $w(I_{j+1}) \le w(I_j)$ and $N_{I_{j+1}} \le N_{I_j}$. If it is generated in a quadratic step, we have $w(I_{j+1}) \le w(I_j)/N_{I_j}$ and $N_{I_{j+1}} = N_{I_j}^2$.

In order to derive a bound on s_1, we appeal to Lemma 25. If $w(I_j)/N_{I_j} \geq 4 \cdot w(I_{s+1})$ for some j, then according to the remark following the definition of the Boundary-Test, the subdivision step from I_j to I_{j+1} is quadratic. However, it might also happen that the step from I_j to I_{j+1} is quadratic, and yet, $w(I_j)/N_{I_j} < 4 \cdot w(I_{s+1})$. If such a j exists, then let j_0 be the minimal such j; otherwise, we define $j_0 = s$. In either case, $s = j_0 + O(1)$. This is clear if $s = j_0$. If $j_0 < s$, the step from I_{j_0} to I_{j_o+1} is quadratic, and hence, $w(I_{j_0+1}) \leq w(I_{j_0})/N_{I_{j_0}} < 4 \cdot w(I_{s+1})$, and hence, a constant number of steps suffices to reduce the width of I_{j_0+1} to the width of I_{s+1}. For $j = 1, \ldots, j_0 - 1$, the sequence $(w(I_j), n_{I_j})$ coincides with a sequence (x_j, n_j) as defined in Lemma 25, where $w := w(I_1)$, $w' := 4w(I_{s+1})$, and $n_1 = m := n_{I_1}$. Namely, if $w(I_j)/N_{I_j} \geq w'$, we have $w(I_{j+1}) \leq w(I_j)/N_{I_j}$ and $n_{I_{j+1}} = 1 + n_{I_j}$, and otherwise, we have $w(I_{j+1}) \leq \frac{3}{4} \cdot w(I_j)$ and $n_{I_{j+1}} = \max(1, n_{I_j} - 1)$. Hence, according to Lemma 25, it follows that j_0 (and thus also s) is bounded by

$$8(n_{I_1} + \log\log\max(4, w(I_1)/w(I_s))) = O(\log(\Gamma + \log M(\sigma_P^{-1})),$$

where we used the bounds for N_{I_1}, $w(I_1)$, and $w(I_s)$ from Lemma 24.

We come to the bound on s_2. Observe first that $\min(|a_1 - a_{s_1}|, |b_1 - b_{s_1}|) \geq \frac{1}{8} w(I_{s_1})$. Obviously, there exists an $s_1' = s_1 + O(\log n)$, such that $w(I_j) \leq 2^{-13-\log n} w(I_{s_1})$ for all $j \geq s_1'$. Furthermore, $N_{I_j} \leq N_{\max} := 2^{O(\Gamma + \log M(\sigma_P^{-1}))}$ for all j according to Lemma 24. Thus, if the sequence $I_{s_1'}$, $I_{s_1'+1}, \ldots$ starts with $m_{\max} := \max(5, \log\log N_{\max} + 1)$ or more consecutive linear subdivision steps, then $N_{I_{j'}} = 4$ and $w(I_{j'}) \leq \frac{w(I_{s_1'})}{4} \leq 2^{-13-\log n} \cdot \frac{w(I_{s_1})}{4} = 2^{-13-\log n} \cdot w(I_{s_1}) \cdot N_{I_{j'}}^{-1}$ for some $j' \leq s_1' + m_{\max}$. Otherwise, there exists a j' with $s_1' \leq j' \leq s_1' + m_{\max}$, such that the step from $I_{j'}$ to $I_{j'+1}$ is quadratic. Since the length of a sequence of consecutive quadratic subdivision steps is also bounded by $m_{\max}$, there must exist a j'' with $j' + 1 \leq j'' \leq j' + m_{\max} + 1$ such that the step from $I_{j''-1}$ to $I_{j''}$ is quadratic, whereas the step from $I_{j''}$ to $I_{j''+1}$ is linear. Then, $N_{I_{j''+1}} = \sqrt{N_{I_{j''}}} = N_{I_{j''-1}}$, and

$$w(I_{j''+1}) \leq \frac{3}{4} w(I_{j''}) \leq \frac{3w(I_{j''-1})}{4N_{I_{j''-1}}} \leq 2^{-13-\log n} \cdot \frac{w(I_{s_1})}{N_{I_{j''+1}}}.$$

Hence, in any case, there exists an $s_2 \leq s_1 + 2m_{\max} + 1$ with $w(I_{s_2}) \leq 2^{-13-\log n} \cdot w(I_{s_1})/N_{I_{s_2}}$.

We next bound $s - s_2$. We only need to deal with the case that $s_2 < s$, and hence, $w(I_j) \leq 2^{-13-\log n} N_{I_j}^{-1} w(I_{s_1})$ for all $j \geq s_2$. For $j \geq s_2$, we also have

$$|x - z_i| > 2^{\log n + 10} N_{I_j} w(I_j) \text{ for all } z_i \notin \mathcal{M}(I_j) \text{ and all } x \in I_j. \tag{35}$$

From (35) and Lemma 23, we conclude that the step from I_j to I_{j+1} is quadratic if $j \geq s_2$ and $w(I_s) \leq 2^{-13} \cdot \frac{w(I_j)}{N_{I_j}}$. Again, it might also happen that there exists a $j \geq s_2$ such that the step from I_j to I_{j+1} is quadratic, and yet, $w(I_s) > 2^{-13} \cdot \frac{w(I_j)}{N_{I_j}}$. If this is the case, then we define s_3 to be the minimal such index; otherwise, we set $s_3 := s$. Clearly, $s = s_3 + O(1)$. We can now again apply Lemma 25. The sequence $(w(I_{s_2+i}), n_{I_{s_2+i}})_{i=1,\ldots,s_3-s_2}$ coincides with a sequence $(x_i, n_i)_{1 \leq i \leq s_3 - s_2}$ as defined in Lemma 25, where $n_1 = m = n_{I_{s_2+1}}$ and $w' := 2^{13} \cdot w(I_s)$. Namely, if $w(I_{s_2+i}) \cdot N_{I_{s_2+i}}^{-1} \geq w'$, then $w(I_{s_2+i+1}) \leq w(I_{s_2+i}) \cdot N_{s_2+i}^{-1}$ and $n_{I_{s_2+i+1}} = 1 + n_{I_{s_2+i}}$, whereas we have $w(I_{s_2+i+1}) \leq \frac{3}{4} w(I_{s_2+i})$ and $n_{I_{s_2+i+1}} = \max(n_{I_{s_2+i}} - 1, 1)$ for $w(I_{s_2+i}) \cdot N_{I_{s_2+i}}^{-1} < w'$. It follows that $s_3 - s_2$ is bounded by $8(n_1 + \log\log\max(4, w(I_{s_2+1})/w')) = O(\log(\Gamma + \log M(\sigma_P^{-1})))$. □

The following theorem now follows immediately from Lemma 26 and our considerations from the beginning of Section 4.1. For the bound on the size of the subdivision forest when running the algorithm on a square-free polynomial P of degree n and with integer coefficients of bit-size

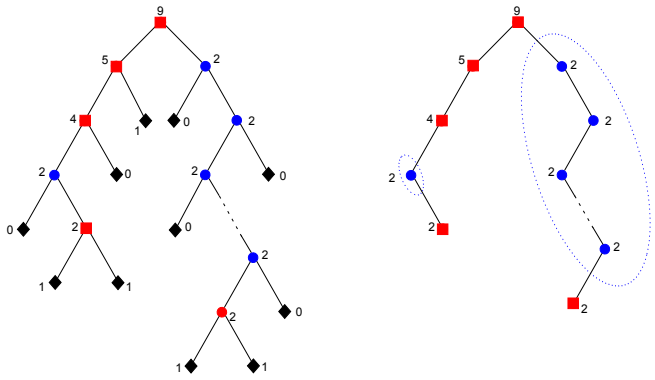

Figure 2: The left figure shows the subdivision tree $\mathcal{T}_k$ rooted at the interval $\mathcal{I}_k$, where, for each node I, the number $\mathrm{var}(f, I)$ of sign variations is given (e.g. $\mathrm{var}(P, \mathcal{I}_k) = 9$). Special nodes and terminal nodes are indicated by red squares and black diamonds, respectively. Blue dots indicate *ordinary* nodes, which are neither special nor terminal. The right figure shows the subtree $\mathcal{T}_k^*$ obtained by removing all terminal nodes. $\mathcal{T}_k^*$ partitions into special nodes and chains of ordinary nodes connecting two consecutive special nodes.

less than τ, we use that $\gamma = O(\log \Gamma)$ and $\log n + \log(\Gamma + \log M(\sigma_P^{-1})) = O(\log(n\tau))$, which is due to Lemma 22.

Theorem 27. *Let $K = \log n + \log(\Gamma + \log M(\sigma_P^{-1}))$. The size $|\mathcal{T}|$ of the subdivision forest is*

$$O\left(\sum_{k=0}^{2\gamma+2} (1 + |\mathcal{M}(\mathcal{I}_k)| \cdot K)\right) = O(nK),$$

where $\mathcal{I}_k$ are the intervals as defined in (25), and $\mathcal{M}(I_k)$ denotes the set of all roots contained in the one-circle region $\Delta(\mathcal{I}_k)$ of $\mathcal{I}_k$. In the case, where the input polynomial has integer coefficients of bit-size less than τ, the above bound simplifies to

$$O\left(\sum_{k=0}^{2\gamma+2} (1 + |\mathcal{M}(\mathcal{I}_k)| \cdot \log(n\tau))\right) = O(n \log(n\tau)).$$

We further provide an alternative bound on the number of iterations needed by the algorithm ANewDSC, which is stated in terms of the number of sign variations $v_k := \mathrm{var}(f, \mathcal{I}_k)$ of f on the initial intervals $\mathcal{I}_k$ instead of the values $|\mathcal{M}(I_k)|$:

Theorem 28. *Let $K = \log n + \log(\Gamma + \log M(\sigma_P^{-1}))$. The size $|\mathcal{T}_k|$ of the subdivision tree $\mathcal{T}_k$ rooted at the interval $\mathcal{I}_k$ is $O(\mathrm{var}(P, \mathcal{I}_k) \cdot K)$, and the size of the subdivision forest $\mathcal{T}$ is*

$$O\left(\sum\nolimits_{k=0}^{2\gamma+2} (1 + \mathrm{var}(P, \mathcal{I}_k) \cdot K)\right).$$

Proof. Let $I_1, \ldots, I_s$ be a sequence of intervals produced by the algorithm such that $I_s \subset I_{s-1} \subset \cdots \subset I_1 \subset \mathcal{I}_k$, and $v = \mathrm{var}(P, I_1) = \cdots = \mathrm{var}(P, I_s)$ for some $v \in \mathbb{N}$. Notice that $v \leq \mathrm{var}(P, \mathcal{I}_k)$ according to Theorem 13. We first show that $s = O(K)$. For $j = 1, \ldots, s$, let $L_n(I_j)$ and $A_n(I_j)$ be the Obreshkoff lens and the Obreshkoff area of I_j, respectively, as defined in Figure 1. Then,

from Theorem 12, we conclude that each $L_n(I_j)$ contains at most v roots of P, whereas each $A_n(I_j)$ contains at least v roots of P. Using the same argument in as in the proof of Lemma 26 shows that either $s = O(K)$ or there exists an $s_1 = O(\log(\Gamma + \log M(\sigma_P^{-1})))$ such that I_1 does not share any endpoint with I_{s_1}, and the distance between any two endpoints of I_1 and I_{s_1} is at least as large as the width $w(I_{s_1})$ of I_{s_1}. Hence, after $\ell = O(\log n)$, further subdivision steps, the one-circle region $\Delta(I_{s_1+\ell})$ of $I_{s_1+\ell}$ is completely contained in the lens $L_n(I_1)$, and thus the one-circle region $\Delta(I_j)$ of any interval I_j, with $j \geq s_2 := s_1 + \ell = O(K)$, is contained in $L_n(I_1)$. Since $L_n(I_1)$ contains at most v roots, we conclude that $|\mathcal{M}(I_j)| \leq v$ for all $j \geq s_2$. The same argument also shows that either $s = O(K)$ or that there exists an s_3, with $s_3 \leq s$ and $s - s_3 = O(K)$ such that the one-circle region $\Delta(I_j)$ of each interval I_j, with $j \leq s_3$, contains the Obreshkoff area $A_n(I_s)$. Thus, we have $|\mathcal{M}(I_j)| \geq v$ for all $j \leq s_3$ as $A_n(I_s)$ contains at least v roots. We can assume that $s_2 \leq s_3$ as, otherwise, $s \leq s_2 + (s - s_3) = O(K)$. Since $|\mathcal{M}(I_j)| \leq v$ for all $j \geq s_2$ and $|\mathcal{M}(I_j)| \geq v$ for all $j \leq s_3$, we have $|\mathcal{M}(I_j)| = v$ for $j = s_2, \ldots, s_3$. Then, from Lemma 26, we conclude that $s_3 - s_2 = O(K)$, and thus also $s = O(K)$. Now, consider the sub-tree $\mathcal{T}_k^*$ of $\mathcal{T}_k$ obtained from $\mathcal{T}_k$ after removing all intervals $I \in \mathcal{T}_k$ with $\mathrm{var}(P, I) \leq 1$; see also Figure 2. Then, $|\mathcal{T}_k| \leq 2 \cdot |\mathcal{T}_k^*| + 1$, and thus it suffices to bound the size of $\mathcal{T}_k^*$. We call an interval $I \in \mathcal{T}_k^*$ *special* if it is either the root $\mathcal{I}_k$ of $\mathcal{T}_k^*$ or if none of its children yields the same number of sign variations as I. Then, the following argument shows that the number of special intervals is at most $\mathrm{var}(P, \mathcal{I}_k)$. We can assume that $\mathrm{var}(P, \mathcal{I}_k) > 1$ as otherwise $\mathcal{T}_k$ has size 1. Now, let $\mathcal{A}_k \subset \mathcal{A}$ be the set of all active intervals in $\mathcal{T}_k$ produced by the algorithm in a certain iteration, and define

$$\mu := \sum_{I \in \mathcal{A} : I \subseteq \mathcal{I}_k \text{ and } \mathrm{var}(P,I) \geq 1} (\mathrm{var}(P, I) - 1),$$

then μ decreases by at least one at each special interval, whereas it stays invariant at all other intervals. Since, $\mu = \mathrm{var}(P, \mathcal{I}_k) - 1$ at the beginning, and $\mu \geq 0$ in each iteration, it follows that there can be at most $\mathrm{var}(f, \mathcal{I}_k)$ special intervals in $\mathcal{T}_k^*$. Notice that $\mathcal{T}_k^*$ splits into special intervals and chains of intervals $I_1, \ldots, I_s$ connecting two consecutive special intervals I and J, that is, $I \subset I_s \subset \cdots \subset I_1 \subset J$, and there exists no special interval I' with $I \subset I' \subset J$. Since $\mathrm{var}(P, I_j)$ is invariant for all $j = 1, \ldots, s$, our above considerations show that each such chain has length $O(K)$. Hence, the claim follows. □

Remark. For polynomials $P = P_0 + \cdots + P_n x^n \in \mathbb{Z}[x]$ with integer coefficients, we remark that the above bound can also be stated in terms of the number $v^+(P) := \mathrm{var}(P_0, \ldots, P_n)$ of sign variations of the coefficient sequence of P, and the number $v^-(P) := \mathrm{var}(P_0, -P_1, P_2, \ldots, (-1)^n \cdot P_n)$ of sign variations of the coefficient sequence of $P(-x)$: After removing a suitable factor x^i, we are left with a polynomial $\bar{P} = P \cdot x^{-i}$, which fulfills $|\bar{P}(0)| \geq 1$, $v^-(\bar{P}) = v^-(P)$, and $v^+(\bar{P}) = v^+(P)$. Let us now estimate the size of the subdivision forest induced by our algorithm when applied to the polynomial $\bar{P}$. In the definition (25) of the intervals $\mathcal{I}_k = (s_k^*, s_{k+1}^*)$, we may choose $s_{\gamma+1}^* = s_\gamma = 0$ as this does not harm the requirements from (23) posed on the intervals $\mathcal{I}_k$. Now, because of the sub-additivity of the function $\mathrm{var}(\bar{P}, .)$, it follows that $\sum_{k=0}^{\gamma+1} \mathrm{var}(\bar{P}, \mathcal{I}_k) \leq \mathrm{var}(\bar{P}, (s_0^*, s_{\gamma+1}^*))$ and $\sum_{k=\gamma+1}^{2\gamma+2} \mathrm{var}(\bar{P}, \mathcal{I}_k) \leq \mathrm{var}(\bar{P}, (s_{\gamma+1}^*, s_{2\gamma+2}))$. The intervals $I^- := (s_0^*, s_{\gamma+1}^*)$ and $I^+ := (s_{\gamma+1}^*, s_{2\gamma+2})$ are contained in $\mathbb{R}_{<0}$ and in $\mathbb{R}_{>0}$, respectively. Hence, it holds that $\mathrm{var}(\bar{P}, I^-) \leq v^-(\bar{P})$, and that $\mathrm{var}(\bar{P}, I^+) \leq v^+(\bar{P})$. Theorem 28 then implies that the subdivision forest has size $O(\gamma + K \cdot (v^-(P) + v^+(P))) = O((v^-(P) + v^+(P) + 1) \log(n\tau))$, where τ bounds the bit size of the coefficients of P. In particular, we obtain:

Theorem 29. *Let $P \in \mathbb{Z}[x]$ be a square-free polynomial of degree n and with integer coefficients of bit size less than τ. Let k be the number of non-zero coefficients of P. Then, for isolating all real roots of P,* ANewDsc *generates a tree of size $O(k \log(n\tau))$.*

Notice that, in the special case, where P is a sparse integer polynomial with only $(\log(n\tau))^{O(1)}$ non-vanishing coefficients, our algorithm generates a tree of size $(\log(n\tau))^{O(1)}$. An illustrative example of the latter kind are Mignotte polynomials of the form $P = x^n - (a \cdot x - 1)^2$, with a an integer of bit size less than τ. In order to isolate the real roots of P, our algorithm generates a tree of size $(\log(n\tau))^{O(1)}$, whereas bisection methods, such as the classical Descartes method, generate a tree of size $\Omega(n\tau)$.

4.2 Bit Complexity

In order to bound the bit complexity of our algorithm, we associate a root z_i of P with every interval I in the subdivision forest and argue that the cost (in number of bit operations) of processing I is

$$\tilde{O}(n(n + \tau_P + n \log M(z_i) + \log M(P'(z_i)^{-1}))). \tag{36}$$

The association is such that each root of P is associated with at most $O(s_{\max} \log n + \log \Gamma)$ intervals, and hence, the total bit complexity can be bounded by summing the bound in (36) over all roots of P and multiplying by $s_{\max} \log n + \log \Gamma$. Theorem 31 results.

We next define the mapping from $\mathcal{T}$ to the set of roots of P. Let I be any non-terminal interval in $\mathcal{T}$. We define a path of intervals starting in I. Assume we have extended the path to an interval I'. The path ends in I' if I' is strongly splitting or if I' is terminal; see the introduction of Section 4.1 for the definitions. If I' has a child I'' with $\mathcal{M}(I') = \mathcal{M}(I'')$, the path continues to this child. If I' has two children J_1 and J_2 with $\mathcal{M}(J_1) \cup \mathcal{M}(J_2) = \mathcal{M}(I')$, and both $\mathcal{M}(J_1)$ and $\mathcal{M}(J_2)$, are nonempty (and hence $\max(|\mathcal{M}(J_1)|, |\mathcal{M}(J_1)|) < |\mathcal{M}(I')|$), the path continues to the child with smaller value of $|\mathcal{M}(*)|$. Ties are broken arbitrarily, but consistently, i.e., all paths passing through I' make the same decision. Let J be the last interval of the path starting in I. Then, the one-circle region of J contains at least one root that is not contained in the one-circle region of any child of J. We call any such root $z \in \mathcal{M}(J) \subset \mathcal{M}(I)$ *associated with* I. With terminal intervals I that are different from any $\mathcal{I}_k$, we associate the same root as with the parent interval. With terminal intervals $\mathcal{I}_k$, we associate an arbitrary root. More informally, with an interval I, we associate a root $z_i \in \mathcal{M}(I)$, which is either "discarded" or isolated when processing the last interval of the path starting in J.

The path starting in an interval has length at most $s_{\max} \cdot \log n$ as there are at most $s_{\max}$ intervals I with the same set $\mathcal{M}(I)$, and $|\mathcal{M}(*)|$ shrinks by a factor of at least $1/2$ whenever the path goes through a splitting node. There are at most $2 \log \Gamma + 1$ intervals $\mathcal{I}_k$ with which any root can be associated, and each root associated with an interval $I \subsetneq \mathcal{I}_k$ cannot be associated with any interval $I' \subsetneq \mathcal{I}_{k'}$ with $k \neq k'$ as the corresponding one-circle regions are disjoint. As a consequence, any root of P is associated with at most $s_{\max} \cdot \log n + 2 \log \Gamma + 1 = O(s_{\max} \log n + \log \Gamma)$ intervals.

We next study the complexity of processing an interval I. We first derive a lower bound for $|P|$ at the subdivision points that are considered when processing I. We introduce the following notation: For an interval $I = (a, b) \in \mathcal{T}$, we call a point ξ *special with respect to* I (or just special if there is no ambiguity) if ξ is

(P1) an endpoint of I, that is, $\xi = a$ or $\xi = b$.

(P2) an admissible point $m^* \in m(I)[w(I) \cdot 2^{-\lceil \log n + 2 \rceil}]$ as computed in the 1-Test.

(P3) an admissible point $\xi_j^* \in \xi_j[w(I) \cdot 2^{-\lceil 5 + \log n \rceil}]$ as computed in (26) in the Newton-Test.

(P4) an admissible point $a_{j_1,j_2}^* \in a_{j_1,j_2}[2^{-\lceil 5+\log n \rceil} \cdot \frac{w(I)}{N_I}]$ or an admissible point $b_{j_1,j_2}^* \in b_{j_1,j_2}[2^{-\lceil 5+\log n \rceil} \cdot \frac{w(I)}{N_I}]$ as computed in (32) in the Newton-Test.

(P5) an admissible point $m_\ell^* \in m_\ell[2^{-\lceil 2+\log n \rceil} \cdot \frac{w(I)}{N_I}]$ or an admissible point $m_r^* \in m_r[2^{-\lceil 2+\log n \rceil} \cdot \frac{w(I)}{N_I}]$ as computed in (34) in the Boundary-Test.

For intervals I with $\mathrm{var}(P, I) = 0$, we have only special points of type (P1), and for intervals with $\mathrm{var}(P, I) = 1$, we have only special points of type (P1) and type (P2). For other intervals, we

consider all types. The following lemma provides a lower bound for the absolute value of P at special points.

Lemma 30. *Let $I \in \mathcal{T}$ be an arbitrary interval, and let ξ be a special point with respect to I. If $\Delta(I)$ contains a root of P, then*

$$|P(\xi)| > 2^{-40n\log n - 2\tau_P} \cdot M(z_i)^{-5n} \cdot \min(1, \sigma(z_i, P))^5 \cdot \min(1, |P'(z_i)|)^5 \tag{37}$$

for all $z_i \in \Delta(I)$. If $\Delta(I)$ contains no root, then ξ fulfills the above inequality for all roots contained in $\Delta(J)$, where J is the parent of I.[16]

Proof. We will prove the claim via induction on the depth k of an interval I, where the depth of the intervals $\mathcal{I}_k$ is one. According to (23), the endpoints of $\mathcal{I}_k$ fulfill inequality (37). Now, if ξ is a special point (with respect to $\mathcal{I}_k$) of type (P2), then

$$|P(\xi)| \geq \frac{1}{4} \cdot \max\{|P(x)| \,;\, x \in m(\mathcal{I}_k)[w(\mathcal{I}_k) \cdot 2^{-\lceil \log n + 2 \rceil}]\}.$$

Since at least one of the points in $m(\mathcal{I}_k)[w(\mathcal{I}_k) \cdot 2^{-\lceil \log n+2 \rceil}]$ has distance more than $w(\mathcal{I}_k) \cdot 2^{-\lceil \log n+3 \rceil} > \frac{1}{8n}$ to all roots of P, it follows that $|P(\xi)| > \frac{|P_n|}{(8n)^n} > 2^{-8n\log n}$, where we use that $|P_n| \geq 1/4$ and $w(\mathcal{I}_k) \geq 1$. An analogous argument yields $|P(\xi)| > \frac{1}{(64n)^{n+1}} \geq 2^{-8n\log n}$ if ξ is a special point of type (P3) to (P5), where we additionally use $N_{\mathcal{I}_k} = 4$ for all k.

For the induction step from k to $k+1$, suppose that $I = (a, b)$ is an interval of depth $k+1$ with parent interval $J = (c, d)$ of depth k. We distinguish the following cases.

The point ξ is a special point of type (P1): The endpoints of I are either subdivision points (as constructed in Steps (Q) or (L) in our algorithm) or endpoints of some interval $J' \in \mathcal{T}$ with $I \subseteq J'$. Hence, they are special points with respect to an interval J' that contains I. Thus, from our induction hypothesis (the depth of J' is smaller than or equal to k) and the fact that $\Delta(I) \subseteq \Delta(J')$, it follows that the inequality (37) holds for all admissible points ξ of type (P1).

The point ξ is a special point of type (P2): Since $\operatorname{var}(P, J) \geq 2$, the two-circle region of J contains at least two roots of P, and thus the disk $\Delta := \Delta_{2w(J)}(m(I))$ with radius $2w(J)$ centered at the midpoint $m(I)$ of I contains at least two roots. This shows that $\sigma(z_i, P) < 2w(J)$ for any root $z_i \in \Delta$. With $\epsilon := w(I) \cdot 2^{-\lceil \log n+2 \rceil}$ and $K := \frac{w(J)}{w(I)} \cdot 2^{\lceil \log n+3 \rceil}$, we can now use Lemma 10 to show that

$$|P(\xi)| > 2^{-4n-1} \cdot K^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)| > 2^{-8n\log n} \cdot \left(\frac{w(J)}{w(I)}\right)^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)|, \tag{38}$$

where z_i is any root in Δ, and $\mu(\Delta)$ denotes the number of roots contained in Δ. If the subdivision step from J to I is linear, then $w(J)/w(I) \in [4/3, 4]$, and thus the bound in (37) is fulfilled. Otherwise, we have $w(J)/w(I) \in [N_I, 4N_I] = [N_J^2, (2N_J)^2]$. In addition, $w(J) \leq 4w(\mathcal{I}_k)/N_J$, where k is the unique index with $J \subseteq \mathcal{I}_k$. We conclude that $N_J \leq 4w(\mathcal{I}_k)/w(J)$, and thus

$$\frac{w(J)}{w(I)} \leq (2N_J)^2 \leq \left(\frac{8w(\mathcal{I}_k)}{w(J)}\right)^2 \leq \frac{2^{12} \cdot M(z_i)^2}{w(J)^2} \tag{39}$$

[16] If $I = \mathcal{I}_k$ for some k and $\Delta(I)$ contains no root, then z_i can be chosen arbitrarily.

since $w(\mathcal{I}_k) \le 8M(x)^2$ for all $x \in \mathcal{I}_k$. Furthermore, since

$$\begin{aligned}|P'(z_i)| &= n|P_n| \cdot \prod_{z_j \in \Delta : z_j \neq z_i} |z_i - z_j| \cdot \prod_{z_j \notin \Delta} |z_i - z_j| \le n \cdot (2w(J))^{\mu(\Delta)-1} \cdot \prod_{z_j \notin \Delta} |z_i - z_j| \\ &\le n \cdot 2^n \cdot w(J)^{\mu(\Delta)-1} \cdot \prod_{z_j \notin \Delta} M(z_i - z_j) \le n \cdot 2^n \cdot w(J)^{\mu(\Delta)-1} \cdot \frac{\mathrm{Mea}(P(z_i - x))}{|P_n|} \\ &\le n \cdot 2^n \cdot w(J)^{\mu(\Delta)-1} \cdot 2^{\tau_P} 2^n M(z_i)^n < 2^{4n+\tau_P} \cdot M(z_i)^n \cdot w(J)^{\mu(\Delta)-1},\end{aligned}$$

it follows that

$$w(J)^{-\mu(\Delta)-1} = w(J)^{-\mu(\Delta)+1} w(J)^{-2} < \frac{2^{8n+\tau_P} \cdot M(z_i)^n}{\sigma(z_i, P)^2 \cdot |P'(z_i)|},$$

where we used that $2w(J) > \sigma(z_i, P)$ in order to bound $w(J)^{-2}$. Hence, it follows from (39) that

$$\begin{aligned}\left(\frac{w(J)}{w(I)}\right)^{-\mu(\Delta)-1} &\ge \left(2^{12} \cdot M(z_i)^2 \cdot w(J)^{-2}\right)^{-\mu(\Delta)-1} \\ &\ge 2^{-12(n+1)} \cdot M(z_i)^{-2(n+1)} \cdot 2^{-16n-2\tau_P} \cdot M(z_i)^{-2n} \sigma(z_i, P)^4 \cdot |P'(z_i)|^4 \\ &> 2^{-32n-2\tau_P} \cdot M(z_i)^{-5n} \cdot \sigma(z_i, P)^4 \cdot |P'(z_i)|^4.\end{aligned}$$

Plugging the latter inequality into (38) eventually yields

$$|P(\xi)| > 2^{-8n \log n} \cdot \left(\frac{w(J)}{w(I)}\right)^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)| > 2^{-40n \log n - 2\tau_P} \cdot M(z_i)^{-5n} \cdot \sigma(z_i, P)^5 \cdot |P'(z_i)|^5.$$

Thus, ξ fulfills the bound (37).

The point ξ is a special point of type (P3): The same argument as in the preceding case also works here. Namely, each disk $\Delta := \Delta_{2w(J)}(\xi_j)$ with radius $2w(J)$ centered at the point ξ_j contains at least two roots, and thus we can use Lemma 10 with $\epsilon := w(I) \cdot 2^{-\lceil \log n + 5 \rceil}$ and $K := \frac{w(J)}{w(I)} \cdot 2^{\lceil \log n + 6 \rceil}$.

The point ξ is a special point of type (P4): The Newton-Test is only performed if the 0-Test and the 1-Test have failed. Hence, we must have $\mathrm{var}(P, I) \ge 2$, and thus, each disk $\Delta := \Delta_{2w(I)}(x_0)$ with radius $2w(I)$ centered at any point $x_0 \in I$ contains at least two roots. We use this fact $x_0 = a_{j_1, j_2}$ and $x_0 = b_{j_1, j_2}$ and obtain, using Lemma 10 with $\epsilon := \frac{w(I)}{N_I} \cdot 2^{-\lceil \log n + 5 \rceil}$ and $K := N_I \cdot 2^{\lceil \log n + 4 \rceil}$,

$$|P(\xi)| > 2^{-4n-1} \cdot K^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)| > 2^{-8n \log n} \cdot N_I^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)|. \quad (40)$$

If the subdivision step from J to I is linear, then $N_I \le N_J \le 4w(\mathcal{I}_k)/w(J)$, where k is the unique index with $J \subseteq \mathcal{I}_k$. If the step from J to I is quadratic, then $N_I = N_J^2 \le (4w(\mathcal{I}_k)/w(J))^2$. Now, the same argument as in the type (P2) case (see (39) and the succeeding computation) shows that

$$\begin{aligned}N_I^{-\mu(\Delta)-1} \ge \left(\frac{4w(\mathcal{I}_k)}{w(J)}\right)^{-\mu(\Delta)-1} &\ge \left(2^{10} \cdot M(z_i)^2 \cdot w(J)^{-2}\right)^{-\mu(\Delta)-1} \\ &\ge 2^{-32n-2\tau_P} \cdot M(z_i)^{-5n} \cdot \sigma(z_i, P)^4 \cdot |P'(z_i)|^4,\end{aligned}$$

and thus

$$|P(\xi)| > 2^{-8n \log n} \cdot N_I^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)| > 2^{-40n \log n - 2\tau_P} \cdot M(z_i)^{-5n} \cdot \sigma(z_i, P)^5 \cdot |P'(z_i)|^5.$$

The point ξ is a special point of type (P5): The same argument as in the previous case works since the Boundary-Test is only applied if $\mathrm{var}(P, I) \ge 2$. □

We can now derive our final result on the bit complexity of ANewDsc:

Theorem 31 (Restatement of Theorem 1). *Let $P = P_n x^n + \ldots + P_0 \in \mathbb{R}[x]$ be a real polynomial with $1/4 \leq |P_n| \leq 1$. The algorithm* ANewDsc *computes isolating intervals for all real roots of P with a number of bit operations bounded by*

$$\tilde{O}(n \cdot (n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1}))) \tag{41}$$

$$= \tilde{O}(n(n^2 + n \log \mathrm{Mea}(P) + \log M(\mathrm{Disc}(P)^{-1}))). \tag{42}$$

The coefficients of P have to be approximated with quality

$$\tilde{O}(n + \tau_P + \max_i(n \log M(z_i) + \log M(P'(z_i)^{-1}))).$$

Proof. We first derive an upper bound on the cost for processing an interval $I \in \mathcal{T}$. Suppose that $\Delta(I)$ contains at least one root: When processing I, we consider a constant number of special points ξ with respect to I. Since each of these points fulfills the inequality (37), we conclude from Lemma 8 that the computation of all special points ξ uses

$$\tilde{O}(n(n + \tau_P + n \log M(z_i) + \log M(\sigma(z_i, P)^{-1}) + \log M(P'(z_i)^{-1}))) \tag{43}$$

bit operations, where z_i is any root contained in $\Delta(I)$. We remark that when applying Lemma 8, we used (23) which implies that $\log M(x) \leq 2(1 + \log M(z_i))$ for all $x \in I$ and all $z_i \in \Delta(I)$.

In addition, Corollary 18 and Corollary 21 yield the same complexity bound as stated in (43) for each of the considered 0-Tests and 1-Tests. Since we perform only a constant number of such tests for I, the bound in (43) applies to all 0-Tests and 1-Tests.

It remains to bound the cost for the computation of the values $\tilde{\lambda}_{j_1,j_2}$ in the Newton-Test: According to the remark following Step (2.2) in the description of the Newton-Test, it suffices to evaluate P and P' at the points $\xi^*_{j_1}$ and $\xi^*_{j_2}$ to an absolute precision of

$$O(\log n + \log N_I + \log M(P(\xi^*_{j_1})^{-1} + \log M(P(\xi^*_{j_2})^{-1}) + \log M(w(I)^{-1}) + \log M(w(I))).$$

Thus, according to Lemma 5 and Lemma 30, the total cost for this step is bounded by

$$\tilde{O}(n(n + \tau_P + n \log M(z_i) + \log M(\sigma(z_i, P)^{-1}) + \log M(P'(z_i)^{-1}))$$

bit operations, where z_i is any root contained in $\Delta(I)$. Here, we used the fact that $2w(I) > \sigma(z_i, P)$ (notice that $\mathrm{var}(P, I) \geq 2$, and thus, the two-circle region of I contains at least two roots) and that $\log N_I = O(\log M(\sigma(z_i, P)^{-1}) + \log M(z_i))$ as shown in the proof of Lemma 30. In summary, for any interval I whose one-circle region contains at least one root, the cost for processing I is bounded by (43). A completely analogous argument further shows that, for intervals I whose one-circle region does not contain any root, the cost for processing I is also bounded by (43), where z_i is any root in $\Delta(J)$ and $J \in \mathcal{T}$ is the parent of I.

Since we can choose an arbitrary root $z_i \in \mathcal{M}(I)$ (or $z_i \in \mathcal{M}(J)$ for the parent J of I if $\mathcal{M}(I)$ is empty) in the above bound, we can express the cost of processing an interval I in terms of the root associated with I. Since any root of P has at most $O(s_{\max} \log n + \log \Gamma)$ many roots associated with it, the total cost for processing all intervals is bounded by

$$\tilde{O}(n \cdot (n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(\sigma(z_i, P)^{-1}) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1})))$$

bit operations, where we used that $\sum_{i=1}^{n} \log M(z_i) = \log \frac{\mathrm{Mea}(P)}{|P_n|}$, $\tau_P \leq n + \log \mathrm{Mea}(P)$, and that the factor $s_{\max} \log n + \log \Gamma$ is swallowed by $\tilde{O}$. We can further discard the sum $\sum_{i=1}^{n} \log M(\sigma(z_i, P)^{-1})$

in the above complexity bound. Namely, if z_k denotes the root with minimal distance to z_i, then (we use inequality (20))

$$\sigma(z_i, P) \geq \frac{|P'(z_i)|}{|P_n| \cdot \prod_{j \neq k,i} |z_j - z_i|} \geq \frac{|P'(z_i)|}{\mathrm{Mea}(P(z - z_i))} \geq \frac{|P'(z_i)|}{2^{\tau_P} \cdot 2^n \cdot M(z_i)^n},$$

and thus, $\sum_{i=1}^n \log M(\sigma(z_i, P)^{-1}) = O(n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^n \log M(P'(z_i)^{-1})$. Since the cost for the computation of Γ is bounded by $\tilde{O}(n^2 \Gamma_P) = \tilde{O}(n^2 \cdot M(\log \mathrm{Mea}(P)))$ bit operations, the bound follows.

For the alternative bound, we use inequalities (19) and (21). □

For the special case where the input polynomial p has integer coefficients, we can specify the above complexity bound to obtain the following result:

Corollary 32 (Restatement of Theorem 2). *For a polynomial $p(x) = p_n \cdot x^n + \cdots + p_0 \in \mathbb{Z}[x]$ with integer coefficients of absolute value 2^τ or less, the algorithm* ANewDsc *computes isolating intervals for all real roots of p with $\tilde{O}(n^3 + n^2\tau)$ bit operations. If p has only k non-vanishing coefficients, the bound becomes $\tilde{O}(n^2(k + \tau))$ bit operations.*

Proof. We first compute a $t \in \mathbb{N}$ with $2^{t-1} \leq |p_n| < 2^t$. Then, we apply ANewDsc to the polynomial $P := 2^{-t} \cdot p$ whose leading coefficient has absolute value between $1/2$ and 1. The complexity bound now follows directly from Theorem 31, where we use that $\tau_P \leq \tau$, $\mathrm{Mea}(P) \leq \mathrm{Mea}(p) \leq (n+1)2^\tau$ (inequality (15)), $\mathrm{Disc}(P) = 2^{-(2n-2)t} \cdot \mathrm{Disc}(p)$, and the fact that the discriminant of an integer polynomial is integral.

For a polynomial with k non-vanishing coefficients, ANewDsc needs $O(k \cdot (\log n + \log(\Gamma + \log M(\sigma_p^{-1})))) = O(k \cdot \log(n\tau))$ iterations to isolate the real roots of p, where k is defined as the number of non-vanishing coefficients of p, by Theorem 29. The bound stated follows. □

5 Root Refinement

In the previous sections, we focused on the problem of isolating all real roots of a square-free polynomial $P \in \mathbb{R}[x]$. Given sufficiently good approximations of the coefficients of P, our algorithm ANewDsc returns isolating intervals I_1 to I_m with the property that $\mathrm{var}(P, I_k) = 1$ for all $k = 1, \ldots, m$. This is sufficient for some applications (existence of real roots, computation of the number of real roots, etc.); however, many other applications also need very good approximations of the roots. In particular, this holds for algorithms to compute a cylindrical algebraic decomposition, where we have to approximate polynomials whose coefficients are polynomial expressions in the root of some other polynomial.[17]

In this section, we show that our algorithm ANewDsc can be easily modified to further refine the intervals I_k to a width less than $2^{-\kappa}$, where κ is any positive integer. Furthermore, our analysis in Sections 5.2 and 5.3 shows that the cost for the refinement is the same as for isolating the roots plus $\tilde{O}(n \cdot \kappa)$. Hence, as a bound in κ, the latter bound is optimal (up to logarithmic factors) since the amortized cost per root and bit of precision is logarithmic in n and κ.

Throughout this section, we assume that z_1 to z_m are exactly the real roots of P and that $I_k = (a_k, b_k)$, with $k = 1, \ldots, m$, are corresponding isolating intervals as computed by ANewDsc. In particular, it holds that $\mathrm{var}(P, I_k) = 1$. According to Theorem 12, the Obreshkoff lens L_n of

[17] For instance, when computing the topology of an algebraic curve defined as the real valued zero set of a bivariate polynomial $f(x, y) \in \mathbb{Z}[x, y]$, many algorithms compute the real roots α of the resultant polynomial $R(x) := \mathrm{res}(f, f_y; y) \in \mathbb{Z}[x]$ first and then isolate the real roots of the fiber polynomials $f(\alpha, y) \in \mathbb{R}[y]$. The second step requires very good approximations of the root α in order to obtain good approximations of the coefficients of $f(\alpha, y)$.

each interval I_k is also isolating for the root z_k. Hence, from the proof of [36, Lemma 5] (see also [36, Figure 3.1]), we conclude that

$$|x - z_j| > \frac{\min(|x - a_k|, |x - b_k|)}{4n} \quad \text{for all } x \in I_k \text{ and all } j \neq k. \tag{44}$$

5.1 The Refinement Algorithm

We modify ANewDsc so as to obtain an efficient algorithm for root refinement. The modification is based on two observations, namely that we can work with a simpler notion of multipoints and that we can replace the 0-Test and the 1-Test with a simpler test based on the sign of P at the endpoints of an interval. In ANewDsc, we used:

(A) computation of an admissible point $m^* \in m[\epsilon]$, where

$$m[\epsilon] := \{m_i := m + (i - \lceil n/2 \rceil) \cdot \epsilon \; ; \; i = 0, \ldots, 2 \cdot \lceil n/2 \rceil\}$$

is a multipoint of size $2 \cdot \lceil n/2 \rceil + 1$.

(B) execution of the 0-Test/1-Test for an interval $(a', b') \subset (a, b)$, where a' and b' are admissible points of corresponding multipoints contained in I.[18]

The reason for putting more than n points into a multipoint was to guarantee, that at least one constituent point has a reasonable distance from all roots contained in the interval. Now, we are working on intervals containing only one root, and hence, can use multipoints consisting of only two points. An interval known to contain at most one root of P contains no root if the signs of the polynomial at the endpoints are equal and contains a root if the signs are distinct (this assumes that the polynomial is nonzero at the endpoints). We will, therefore, work with the following modifications when processing an interval $I \subset I_k$:

(A') computation of an admissible point $m^* \in m[\epsilon]'$, where

$$m[\epsilon]' = \{m'_1, m'_2\} := \{m - \lceil n/2 \rceil \epsilon, m + \lceil n/2 \rceil \epsilon\} \subset m[\epsilon]$$

consists of the first and the last point from $m[\epsilon]$ only.[19]

(B') execution of a *sign-test* on an interval $I' = (a', b') \subset (a, b)$ (i.e., the computation of $\operatorname{sgn}(f(a') \cdot f(b'))$), where a' and b' are admissible points in some $m[\epsilon]'$.[20]

We now give details of our refinement method which we denote Refine. As input, Refine receives isolating intervals I_1 to I_m for the real roots of P as computed by ANewDsc and a positive integer κ. It returns isolating intervals J_k, with $J_k \subset I_k$ and width $w(J_k) < 2^{-\kappa}$.

Algorithm: Refine

Input: A polynomial $P(x)$ as in (2), isolating intervals I_j for the real roots of P with $\operatorname{var}(P, I_j) = 1$ for $j = 1, \ldots, m$, and a positive integer κ.

Output: Isolating intervals I'_j for the real roots of P with $I'_j \subseteq I_j$ and $w(I'_j) < 2^{-\kappa}$ for

[18] Notice that this step also uses the computation of admissible points.

[19] In fact, one can show that choosing two arbitrary points from $m[\epsilon]$ does not affect any of the following results.

[20] More precisely, we compute $s := \operatorname{sgn}(P(a') \cdot P(b'))$. If $s > 0$, then I' contains no root. If $s < 0$, then I' isolates the root z_k. Since $\operatorname{var}(P, I_k) = 1$, it follows that $s > 0$ if and only if $\operatorname{var}(P, I') = 0$, and $s < 0$ if and only if $\operatorname{var}(P, I') = 1$.

$j = 1, \ldots, m$.

(1) $\mathcal{A} := \{(I_j, 4)\}_{j=1,\ldots,m}$ and $\mathcal{O} := \emptyset$.

(2) while $\mathcal{A} \neq \emptyset$ do

(2.1) Choose an arbitrary pair (I, N_I) from $\mathcal{A}$, with $I = (a, b)$, and remove (I, N_I) from $\mathcal{A}$

(2.2) Run the Boundary-Test and the Newton-Test with input P and I, where the steps in (A) and (B) are replaced by the respective modifications in (A') and (B').

(2.3) If one of the tests in Step (2.2) returns True and an interval $I' \subseteq I$, then add I to $\mathcal{O}$ if $w(I') < 2^{-\kappa}$, and add $(I', N_{I'}) = (I', N_I^2)$ to $\mathcal{A}$ if $w(I') \geq 2^{-\kappa}$. Then, go to Step (2.1).

(quadratic step)

(2.4) Compute an admissible point $m^* \in m(I)[\frac{w(I)}{2^{\lceil 2+\log n \rceil}}]'$ using the algorithm **Admissible Point**.

(2.4.1) $I' := (a, m^*)$, $I'' := (m^*, b)$, and $N_{I'} := N_{I''} := \max(4, \sqrt{N_I})$.

(2.4.2) If $P(a') \cdot P(m^*) < 0$ and $w(I') < 2^{-\kappa}$, add I' to $\mathcal{O}$.

(2.4.3) If $P(a') \cdot P(m^*) < 0$ and $w(I') \geq 2^{-\kappa}$, add $(I', N_{I'})$ to $\mathcal{A}$.

(2.4.4) If $P(a') \cdot P(m^*) > 0$ and $w(I'') < 2^{-\kappa}$, add I'' to $\mathcal{O}$.

(2.4.5) If $P(a') \cdot P(m^*) > 0$ and $w(I'') \geq 2^{-\kappa}$, add $(I'', N_{I''})$ to $\mathcal{A}$.

(linear step)

(3) return $\mathcal{O}$

The reader may notice that, in comparison to ANewDsc, there are only a constant number of polynomial evaluations at each node, and thus, there is no immediate need to use an algorithm for fast approximate multipoint evaluation.[21] Namely, when processing a certain interval $I \in \mathcal{A}$, we have to compute admissible points $m^* \in m[\epsilon]'$ for a constant number of $m[\epsilon]'$, and each $m[\epsilon]'$ consists of two points (i.e., $m'_1 = m - \lceil n/2 \rceil$ and $m'_2 = m + \lceil n/2 \rceil$) only. For computing an admissible point m^*, we evaluate $P(x)$ at $x = m'_1$ and $x = m'_2$ to an absolute error less than 2^{-L}, where $L = 1, 2, 4, 8 \ldots$. We stop as soon as we have computed a 4-approximation for at least one of the values $P(m'_1)$ and $P(m'_2)$. The cost for each such computation is bounded by

$$\tilde{O}(n(\tau_p + n \log \max(M(m'_1), M(m'_2)) + \log M(\max(|P(m'_1|, |P(m'_2)|)^{-1}))) \tag{45}$$

bit operations; see the proof of Lemma 5.

5.2 Analysis

We first derive bounds on the number of iterations that Refine needs to refine an isolating interval I_k to a size less than $2^{-\kappa}$.

Lemma 33. *For refining an interval I_k to a size less than $2^{-\kappa}$, Refine needs at most*

$$s_{\max,k} \cdot |\mathcal{M}(I_k)| = O((\log n + \log(\log M(z_k) + \kappa))) \cdot |\mathcal{M}(I_k)|$$

iterations, where $s_{\max,k}$ has size $O(\log n + \log(\log M(z_k) + \kappa)) = O(\log n + \log(\Gamma + \kappa))$ and $\mathcal{M}(I_k)$ is the set of roots contained in the one-circle region of I_k. The total number of iterations to refine all intervals I_k to a size less than $2^{-\kappa}$ is $\tilde{O}(n(\log n + \log(\Gamma + \kappa)))$.

[21] However, we will later show how to make good use of approximate multipoint evaluation in order to improve the worst case bit complexity.

Proof. Similar as in the proof of Lemma 24, we first derive upper and lower bounds for the values N_I and $w(I)$, respectively, where $I \subset I_k$ is an *active* interval produced by REFINE. According to property (23), we have $w(I) \leq w(I_k) \leq 4 \cdot M(z_k)^2$. Hence, it follows that either $N_I = 4$ or $w(I) \leq w(I_k)/\sqrt{N_I} \leq 4 \cdot M(z_k)^2/\sqrt{N_I}$, and thus, (notice that $w(I) \geq 2^{-\kappa}$)

$$N_I \leq 16 \cdot \frac{M(z_k)^4}{w(I)^2} \leq 2^{4(\Gamma+1)+2\kappa}.$$

Furthermore, for *each* interval $I \subset I_k$ produced by REFINE, we have

$$\min(2^\Gamma, 4M(z_k)^2) \geq w(I) \geq \frac{w(J)}{N_J} \geq 2^{-3\kappa-4(\Gamma+1)},$$

where J is the parent interval of I of size $w(J) \geq 2^{-\kappa}$. The bound for the number of iterations is then an immediate consequence of Lemma 23 and of our considerations in the proof of Lemma 26. Namely, exactly the same argument as in the proof of Lemma 26 shows that the *maximal length of any path between splitting nodes*, denoted $s'_{\max,k}$, is $O(\log n + \log(\log M(z_k) + \kappa))$,[22] and thus the path from I_k to the refined interval $J_k \subset I_k$ of size less than $2^{-\kappa}$ has length $s'_{\max,k} \cdot |\mathcal{M}(I_k)|$. □

In the next step, we estimate the cost for processing an active interval I.

Lemma 34. *For an active interval $I \subset I_k$ of size $w(I) \geq \sigma(z_k, P)/2$, the cost for processing I is bounded by*

$$\tilde{O}(n(n + \tau_P + n \log M(z_i) + \log M(P'(z_i)^{-1}))),$$

where z_i is any root contained in the one-circle region of I. If $w(I) < \sigma(z_k, P)/2$, the cost for processing I is bounded by

$$\tilde{O}(n(\kappa + n + \tau_P + n \log M(z_k) + \log M(P'(z_k)^{-1}))).$$

Proof. Suppose that $w(I) \geq \sigma(z_k, P)/2$, and let $\xi \in m[\epsilon]'$ be an admissible point that is computed when processing I. For at least one of the two points (w.l.o.g. say m'_1) in $m[\epsilon]'$, the distance to the root z_k as well as the distance to both endpoints of I is at least $\lceil n/2 \rceil \cdot \epsilon \geq n \cdot \epsilon/2$. Hence, from inequality (44) we conclude that the distance from m'_1 to any root of P is at least $\epsilon/8$. Now, exactly the same argument as in the proof of Lemma 10 (with $x_{i_0} := m'_1$) shows that

$$|P(\xi)| > 2^{-6n-1} \cdot K^{-\mu(\Delta)-1} \cdot \sigma(z_i, P) \cdot |P'(z_i)| \quad \text{for all } z_i \in \Delta,$$

where $K \geq 2 \cdot \lceil n/2 \rceil$ is any positive real value, such that the disk $\Delta := \Delta_{K \cdot \epsilon}(m)$ contains at least two roots of P. Since $w(I) \geq \sigma(z_k, P)/2$, it further follows that the disk $\Delta_{2w(I)}(m(I))$ contains at least two roots. Thus, we can use the same argument as in the proof of Lemma 30 (type (P2)-(P5) cases) to prove that the inequality (37) holds for ξ. In addition, inequality (37) also holds for the endpoints of I_k (as already proven in the analysis of the root isolation algorithm), and thus, by induction, it holds for the endpoints of any node $I \subset I_k$. Hence, when processing I, there are a constant number of approximate polynomial evaluations with a precision bounded by

$$\begin{aligned} &O(n \log n + \tau_P + n \log M(z_i) + \log M(\sigma(z_i, P)^{-1}) + \log M(P'(z_i)^{-1})) \\ &\qquad O(n \log n + \tau_P + n \log M(z_i) + \log M(P'(z_i)^{-1})), \end{aligned} \tag{46}$$

where we again used that $\log M(\sigma(z_k, P)^{-1}) = O(n \log M(z_i) + \tau_P + \log M(P'(z_i)^{-1}))$. This proves the first part; see the proof of Theorem 31 and Lemma 5.

[22] For REFINE, a node I is splitting if either I is terminal (i.e., $w(I) < 2^{-\kappa}$) or $\mathcal{M}(I) \neq \mathcal{M}(I')$ for the child I' of I. If I^*_k denotes the first node whose one-circle region isolates the root z_k (i.e. $|\mathcal{M}(I^*_k)| = 1$), then it follows that the path connecting I^*_k with J_k has length less than or equal to $s'_{\max,k}$.

For the second part, we now assume that $w(I) < \sigma(z_k, P)/2$. Let $\xi \in m[\epsilon]'$ be an admissible point that is considered when processing I. Then, the disk $\Delta_{w(I)}(m(I))$ contains the root z_k but no other root of P. Hence, for any $x \in I$, it holds that

$$|P(x)| = |P_n| \cdot |x - z_k| \cdot \prod_{i \neq k} |x - z_i| \geq |P_n| \cdot |x - z_k| \cdot \prod_{i \neq k} \frac{|z_k - z_i|}{4} = |x - z_k| \cdot \frac{|P'(z_k)|}{2^{2(n-1)}}.$$

Since the distance of at least one of the two points in $m[\epsilon]'$ (w.l.o.g. say m_1') to the root z_k is larger than or equal to $n\epsilon/2 \geq w(I)/(8N_I)$, it follows that

$$\begin{aligned} |P(\xi)| &\geq \frac{|P(m_1')|}{4} \geq \frac{1}{4} \cdot \frac{w(I)}{8N_I} \cdot 2^{-2(n-1)} \cdot |P'(z_k)| \\ &\geq \frac{w(I)^3}{2^9 M(z_k)^4} \cdot 2^{-2(n-1)} \cdot |P'(z_k)| > 2^{-4\Gamma - 3\kappa - 2n - 7} \cdot |P'(z_k)|, \end{aligned}$$

where we used the bounds for $w(I)$ and N_I as computed in the proof of Lemma 33. Furthermore, the endpoints of I fulfill the inequality (37), and thus all approximate polynomial evaluations (when processing I) are carried out with an absolute precision of

$$\begin{aligned} &O(\log M(w(I)^{-1}) + n \log n + \tau_P + n \cdot \log M(z_k) + \log M(\sigma(z_k, P)^{-1}) + \log M(P'(z_k)^{-1})) = \\ &O(\kappa + n \log n + \tau_P + n \cdot \log M(z_k) + \log M(P'(z_k)^{-1})) \end{aligned} \tag{47}$$

This proves the second claim. □

Combining Lemma 33 and Lemma 34 now yields the following result:

Theorem 35. *The cost for refining I_k to an interval of size less than $2^{-\kappa}$ is bounded by*

$$\tilde{O}(n\kappa + \sum_{i : z_i \in \mathcal{M}(I_k)} n(n + \tau_P + n \log M(z_i) + \log M(P'(z_i)^{-1}))).$$

The cost for refining all isolating intervals to a size less than $2^{-\kappa}$ is bounded by

$$\tilde{O}(n^2\kappa + n(n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1}))). \tag{48}$$

Proof. We split the total cost into those for refining the interval I_k to a size less than $\sigma(z_k, P)/2$ and into those for the additional refinement steps until the interval has size less than $2^{-\kappa}$. For the latter cost, we remark that $|\mathcal{M}(I)| = 1$ if I is an isolating interval for z_k of width $w(I) < \sigma(z_k, P)/2$. Hence, there are at most $s'_{\max,k}$ refinement steps of I, each of cost $\tilde{O}(n(\kappa + n + \tau_P + n \log M(z_k) + \log M(P'(z_k)^{-1})))$. It remains to bound the cost for refining I_k to a width of less than $\sigma(z_k, P)/2$. According to Lemma 34, the cost for processing an interval I of width $w(I) \geq \sigma(z_k, P)/2$ is bounded by $\tilde{O}(n(n + \tau_P + n \log M(z_i) + \log M(P'(z_i)^{-1})))$, where we can choose an arbitrary root $z_i \in \mathcal{M}(I)$. If we choose the root z_i that is associated[23] with I, then each root in $\mathcal{M}(I_k)$ is considered at most $s'_{\max,k}$ many times. Thus, the first complexity bound follows. The bound (48) for the total cost for refining all intervals follows immediately from the first bound and from the fact that the one-circle regions of the intervals I_k are pairwise disjoint. □

[23] Essentially, we use the the same definition as in Section 4.2. More precisely, we say that a root z_i is associated with I if $z_i \in \mathcal{M}(I)$ and the number of children $I' \subset I$ with $z_i \in \mathcal{M}(I')$ is minimal for all roots in $\mathcal{M}(I)$. Notice that each root $z_i \in \Delta(I_k)$ is associated with at most $s'_{\max,k}$ intervals; see Lemma 33.

5.3 Asymptotic Improvements

In this section, we show that our complexity bound (48) for refining all intervals can be further improved, that is, the term $n^2\kappa$ can be replaced by $n\kappa$. We achieve this result by using fast approximate multipoint evaluation. For an integer l, with $0 \leq l \leq \log \kappa + 1$, we consider all active intervals in the refinement process whose width is larger than or equal to 2^{-2^l}. We call each such interval an *l-active* interval. We start with $l = 0$ and proceed in rounds: For a fixed l, let (w.l.o.g.) $I'_1, \ldots, I'_{m(l)}$, with $m(l) \leq m$ and $I'_k \subset I_k$, be all l-active intervals. The crucial idea is now to carry out the polynomial evaluations for each of the l-active intervals "in parallel" by using fast approximate multipoint evaluation. That is, instead of computing admissible points m_k^* of $m_k[\epsilon_k]'$ for each interval I'_k independently, we aggregate these evaluations in one multipoint evaluation. We continue refining all l-active intervals in this way until all intervals have size less than 2^{-2^l}. Once this happens, we proceed in the same manner with $l := l + 1$. Notice that after a few iterations (for a fixed l) some of the l-active intervals might become smaller than 2^{-2^l}, whereas other intervals are still l-active. Intervals which become smaller than 2^{-2^l} are then not considered anymore in this round. The cost for each multipoint evaluation is comparable to the cost of the most expensive individual evaluation multiplied by a logarithmic factor. Furthermore, in each iteration, a constant number of multipoint evaluations is sufficient because for each interval I'_k, there are only constantly many evaluations. Hence, the cost for each iteration is bounded by

$$\tilde{O}(n(2^l + n + \tau_P + n \cdot \log M(z_i) + M(P'(z_i)^{-1})), \tag{49}$$

where z_i is any root in the one-circle region of the interval I'_{k_0}, and I'_{k_0} is the interval for which the highest precision is needed; see (46) and (47), and use that $w(I'_{k_0}) \geq 2^{-2^l}$. We now distinguish the following three cases:

(1) $l = 0$: The cost in (49) is bounded by

$$\tilde{O}(n(n + \tau_P + n \cdot \log M(z_i) + M(P'(z_i)^{-1})).$$

We allocate the cost to a root z_i that is associated to the interval I'_{k_0}.

(2) $l > 0$ and there exists an interval I'_k with $\mathcal{M}(I'_k) > 1$: Since $2^{-2^{l-1}} > w(I'_k) \geq 2^{-2^l}$, each root z_i in $\mathcal{M}(I_k)$ has separation $\sigma(z_i, P) < 2^{-2^{l-1}}$, and thus, replacing the term 2^l in (49) by $\log M(\sigma(z_i, P)^{-1})$ yields

$$\tilde{O}(n(\log M(\sigma(z_i, P)^{-1}) + n + \tau_P + n \cdot \log M(z_j) + M(P'(z_j)^{-1})),$$

where z_i is some root in $\mathcal{M}(I'_k)$ and z_j is some root in $\mathcal{M}(I'_{k_0})$. We allocate the cost to roots z_i and z_j that are associated to the intervals I'_k and I'_{k_0}, respectively.

(3) $l > 0$ and $\mathcal{M}(I'_k) = 1$ for all $k = 1, \ldots, m(l)$: We allocate the cost to a root z_i that is associated to I'_{k_0}.

It remains to sum up the cost over all iterations. The sum over all iterations of type (1) and (2) is bounded by

$$\tilde{O}(n(n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(\sigma(z_i, P)^{-1}) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1}))) =$$
$$\tilde{O}(n(n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1})))$$

because the cost of an iteration is allocated to a certain root z_i only a logarithmic number of times. For the sum over all iterations of type (3), we remark that, for a certain l, there can be at most $\max_{k=1,\ldots,m} s'_{\max,k}$ iterations of type (3). Namely, the number of iterations to refine a certain interval I'_k with $\mathcal{M}(I'_k) = 1$ to a size less than $2^{-\kappa}$ is bounded by $s'_{\max,k}$. Hence, the sum of the first term $n \cdot 2^l$ in (49) over all l is bounded by $\max_{k=1,\ldots,m} s_{\max,k} \cdot n \cdot \kappa$. The sum over the remaining term is again bounded by

$$\tilde{O}(n(n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1})))$$

because the cost of an iteration is allocated to a certain root z_i only a logarithmic number of times. We summarize:

Theorem 36 (Restatement of Theorem 3). *Let $P = P_n x^n + \ldots + P_0 \in \mathbb{R}[x]$ be a real polynomial with $1/4 \leq |P_n| \leq 1$, and let κ be a positive integer. Computing isolating intervals of size less than $2^{-\kappa}$ for all real roots needs a number of bit operations bounded by*

$$\tilde{O}(n \cdot (\kappa + n^2 + n \log \mathrm{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1}))) \tag{50}$$

$$= \tilde{O}(n \cdot (\kappa + n^2 + n \log \mathrm{Mea}(P) + \log M(\mathrm{Disc}(P)^{-1}))). \tag{51}$$

The coefficients of P have to be approximated with quality

$$\tilde{O}(\kappa + n + \tau_P + \max_i(n \log M(z_i) + \log M(P'(z_i)^{-1}))).$$

For a polynomial P with integer coefficients of size less than 2^τ, computing isolating intervals of size less than $2^{-\kappa}$ for all real roots needs $\tilde{O}(n(n^2 + n\tau + \kappa))$ bit operations.

6 Conclusion

We have introduced a novel subdivision algorithm, denoted ANewDsc, to compute isolating intervals for the real roots of a square-free polynomial with real coefficients. The algorithm can also be used to further refine the isolating intervals to an arbitrary small size.

In our approach, we combine the Descartes method with Newton iteration and approximate (but certified) arithmetic. As a result, ANewDsc uses an almost optimal number of iterations, and the precision demand as well as the working precision are directly related to the actual geometric locations of the roots; hence, the algorithm adapts to the actual hardness of the input. The bit complexity of our method matches that of Pan's method from 2002, which is the best algorithm known and goes back to Schönhage's splitting circle method from 1982. By comparison, our approach is completely different from Pan's method and, in addition, it is simpler. Because of its simpleness, we consider our algorithm to be well suited for an efficient implementation. Furthermore, it can be used to isolate the roots in a given interval only, whereas Pan's method has to compute all complex roots at the same time.

The first author, A. Kobel, and F. Rouillier are currently working on an implementation of ANewDsc. More precisely, they are considering a randomized version of the algorithm, where admissible (subdivision) points at chosen randomly and not via approximate multipoint evaluation as proposed in this paper (see Section 2.2). They expect the randomized version to show good practical behavior. It may even have an expected bit complexity comparable to the algorithm presented in this paper.

References

[1] J. Abbott. Quadratic Interval Refinement for Real Roots. *Communications in Computer Algebra*, 28:3–12, 2014. Poster presented at the International Symposium on Symbolic and Algebraic Computation (ISSAC), 2006.

[2] A. G. Akritas and A. Strzeboński. A comparative study of two real root isolation methods. *Nonlinear Analysis:Modelling and Control*, 10(4):297–304, 2005.

[3] A. Alesina and M. Galuzzi. A new proof of Vincent's theorem. *L'Enseignement Mathematique*, 44:219–256, 1998.

[4] D. Bini and G. Fiorentino. Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder. *Numerical Algorithms*, 23:127–173, 2000.

[5] M. Burr and F. Krahmer. Sqfreeeval: An (almost) optimal real-root isolation algorithm. *Journal of Symbolic Computation*, 47(2):153–166, 2012.

[6] G. E. Collins. Continued fraction real root isolation using the Hong bound. *Journal of Symbolic Computation*, 2014. in press.

[7] G. E. Collins and A. G. Akritas. Polynomial real root isolation using Descartes' rule of signs. In *Symposium on symbolic and algebraic computation (SYMSAC)*, pages 272–275, 1976.

[8] Z. Du, V. Sharma, and C. Yap. Amortized bounds for root isolation via Sturm sequences. In *Symbolic Numeric Computation (SNC)*, pages 113–130, 2007.

[9] A. Eigenwillig. *Real Root Isolation for Exact and Approximate Polynomials Using Descartes' Rule of Signs*. PhD thesis, Saarland University, 2008.

[10] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. A Descartes algorithm for polynomials with bit-stream coefficients. In *Computer Algebra in Scientific Computation (CASC)*, pages 138–149, 2005.

[11] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the descartes method. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 71–78, 2006.

[12] I. Z. Emiris, V. Y. Pan, and E. P. Tsigaridas. Algebraic algorithms. In *Computing Handbook, Third Edition: Computer Science and Software Engineering*, pages 10: 1–30. 2014.

[13] S. Fortune. An iterated eigenvalue algorithm for approximating roots of univariate polynomials. *Journal of Symbolic Computation*, 33(5):627–646, 2002.

[14] X. Gourdon. *Combinatoire, Algorithmique et Géométrie des Polynomes.* PhD thesis, École Polytechnique, 1996.

[15] M. Hemmer, E. P. Tsigaridas, Z. Zafeirakopoulos, I. Z. Emiris, M. I. Karavelas, and B. Mourrain. Experimental evaluation and cross-benchmarking of univariate real solvers. In *Symbolic Numeric Computation (SNC)*, pages 45–54, 2009.

[16] J. R. Johnson and W. Krandick. Polynomial real root isolation using approximate arithmetic. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 225–232, 1997.

[17] N. Kamath. Subdivision Algorithms for Complex Root Isolation: Empirical Comparisons. Master's thesis, Kellogg College, University of Oxford, 2010.

[18] M. Kerber and M. Sagraloff. Root refinement for real polynomials using quadratic interval refinement. *Journal of Computational and Applied Mathematics*, 280:377–395, 2015. a preliminary version appeared in the proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC), 2011.

[19] P. Kirrinnis. Partial fraction decomposition in (z) and simultaneous newton iteration for factorization in C[z]. *Journal of Complexity*, 14(3):378–444, 1998.

[20] A. Kobel, F. Rouillier, and M. Sagraloff. personal communication.

[21] A. Kobel and M. Sagraloff. Fast approximate polynomial multipoint evaluation and applications. *CORR*, abs/1304.8069, 2013.

[22] A. Kobel and M. Sagraloff. On the complexity of computing with planar algebraic curves. *Journal of Complexity*, 31(2):206–236, 2014.

[23] J. M. McNamee. A 2002 update of the supplementary bibliography on roots of polynomials. *Journal of Computational and Applied Mathematics*, 142(2):433–434, 2002.

[24] J. M. McNamee. *Numerical Methods for Roots of Polynomials*. Number 1 in Studies in Computational Mathematics. Elsevier Science, 2007.

[25] J. M. McNamee and V. Y. Pan. *Numerical Methods for Roots of Polynomials*. Number 2 in Studies in Computational Mathematics. Elsevier Science, 2013.

[26] K. Mehlhorn, M. Sagraloff, and P. Wang. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 66:34–69, 2015. A preliminary version appeared in the proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC), 2013.

[27] M. Mignotte. *Mathematics for Computer Algebra*. Springer, 1992.

[28] N. Obreshkoff. *Verteilung und Berechnung der Nullstellen reeller Polynome*. VEB Deutscher Verlag der Wissenschaften, 1963.

[29] N. Obreshkoff. *Zeros of Polynomials*. Marina Drinov, Sofia, 2003. Translation of the Bulgarian original.

[30] V. Pan. Univariate Polynomials: Nearly Optimal Algorithms for Numerical Factorization and Root Finding. *Journal of Symbolic Computation*, 33(5):701–733, 2002.

[31] V. Pan and E. Tsigaridas. On the boolean complexity of real root refinement. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 1–8, 2013.

[32] V. Y. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM Review*, 39(2):187–220, 1997.

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

[34] P. Ritzmann. A fast numerical algorithm for the composition of power series with complex coefficients. *Theoretical Computer Science*, 44(0):1 – 16, 1986.

[35] F. Rouillier and P. Zimmermann. Efficient isolation of [a] polynomial's real roots. *Journal of Computational and Applied Mathematics*, 162:33–50, 2004.

[36] M. Sagraloff. When Newton meets Descartes: A simple and fast algorithm to isolate the real roots of a polynomial. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 297–304, 2012.

[37] M. Sagraloff. A near-optimal algorithm for computing real roots of sparse polynomials. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 359–366, 2014.

[38] M. Sagraloff. On the complexity of the Descartes method when using approximate arithmetic. *Journal of Symbolic Computation*, 65(0):79 – 110, 2014.

[39] M. Sagraloff and C. K. Yap. A simple but exact and efficient algorithm for complex root isolation. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 353–360, 2011.

[40] I. Schoenberg. Über variationsvermindernde lineare Transformationen. *Mathematische Zeitschrift*, pages 321–328, 1930.

[41] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity, 1982; updated 2004. Manuscript, Department of Mathematics, University of Tübingen.

[42] V. Sharma. Complexity of real root isolation using continued fractions. *Theoretical Computer Science*, 409:292–310, 2008.

[43] E. P. Tsigaridas and I. Z. Emiris. On the complexity of real root isolation using continued fractions. *Theoretical Computer Science*, 392(1-3):158–173, 2008.

[44] J. van der Hoeven. Fast composition of numeric power series. Technical Report 2008-09, Université Paris-Sud, France, 2008. http://www.texmacs.org/joris/fastcomp/fastcomp.html.

[45] J.-C. Yakoubsohn. Finding a cluster of zeros of univariate polynomials. *Journal of Complexity*, 16(3):603 – 638, 2000.

[46] C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, 2000.