Computing the roots of a univariate polynomial is a fundamental and long-studied problem of computational algebra with numerous 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 drawback of Pan’s method is that it is quite involved\footnote{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.} 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. It has been an open question for decades whether there exists a dedicated real root solver with a comparable runtime as the splitting circle method. This paper gives a positive answer to this question.

More precisely, we introduce a variant of the Descartes method, denoted ANewDsc, that 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. Our algorithm can also be used to refine the isolating intervals to an arbitrary small size. The bit complexity of ANewDsc matches the complexity of Pan’s method and, in particular, it achieves near optimal complexity for computing arbitrary good approximations of all real roots. ANewDsc derives its speed from the combination of the Descartes method with Newton iteration and approximate arithmetic. By comparison, our algorithm is considerably simpler than Pan’s method, and it can be used to compute the roots in a given interval only.

**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 studied for thousands of years (in fact already studied by the Babylonians, \(\approx 2000\text{ BC}\)) and that numerous approaches have been developed for this fundamental problem. Furthermore, this problem has strongly influenced
the development of mathematics, in particular, in algebra, algebraic geometry, and numerical computation; see [24] for an extensive historical treatment. Root finding is still an active area of research, and numerous new approaches are proposed each year. In 1982, Schönhage [30] proposed the splitting circle method for approximately factorizing a polynomial into linear terms. From an approximate factorization, one can derive arbitrary good approximations of all complex roots as well as corresponding isolating disks. Later, the splitting circle method was considerably refined by Pan [23]. Pan’s method is nearly optimal for the approximate factorization. For isolating and approximating the complex roots, it yields the best method available. The main drawback of Pan’s algorithm is that it is quite involved (see Footnote 1) and that it needs to compute all complex roots at the same time. It has not yet been implemented. 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 numerous corresponding implementations in computer algebra systems. With respect to computational complexity, all of these methods considerably lag behind the splitting circle approach, and it has been an open question for decades now whether there exists a practical method for real root finding with a comparable runtime as Pan’s method. This paper gives a positive answer to this question. For more details on related work, we refer to the corresponding section in the introduction.

We now give some details of our algorithm. 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 upper 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 \( I \) contains no root of \( p \), and \( v_I = 1 \), if the union of the circumcircles of the two equilateral triangles with side \( I \) contains exactly one root of \( I \), see Figure 1.

The advantages of the Descartes method are its simplicity and the fact that it applies to polynomials with arbitrary 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

\(^2\)Descartes did not formulate an algorithm for isolating the real roots of a polynomial but (only) a method for upper bounding the number of positive real roots of a univariate polynomial (Descartes’ rule of signs). Collins and Akritas [6] based on ideas going back to Vincent formulated a bisection algorithm based on Descartes’ rule of signs.
overcome each one of the two weaknesses separately (see Section 1.1); however, it is not known how to overcome them simultaneously. Our main result achieves this. We present an algorithm ANEWDSC (read approximate-arithmetic-Newton-Descartes) that overcomes both shortcomings at the same time. Our algorithm applies to arbitrary real polynomials given through coefficient oracles, and our algorithm works well in the presence of clustered roots.

More specifically, we prove the following theorems:

**Theorem.** Let $P(x) = P_n x^n + \ldots + P_1 x + P_0 \in \mathbb{R}[x]$ be a real, square-free polynomial of degree $n$ with $1/4 \leq P_n \leq 1$. Algorithm ANEWDSC determines isolating intervals for all real roots of $P$ with a number of bit operations bounded by $\tilde{O}(n(n^2 + n \log \text{Mea}(P) + \log M(\text{Disc}(P)^{-1}))).$

The coefficients of $P$ have to be approximated with absolute error

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

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

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

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

For general real polynomials, the bit complexity of algorithm ANEWDSC matches the bit complexity of the best algorithm known ([20]). For polynomials with integer coefficients, the bit complexity of the best algorithm known ([11, Theorem 3.1]) is $\tilde{O}(n^2 \tau)$, however, for the price of using $\Omega(n^2 \tau)$ bit operations for every input. Both algorithms mentioned are based on Pan’s approximate factorization algorithm [23], which is quite complex and always computes all complex roots.

Our algorithm is much simpler, and it 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. The best-case complexity is much lower. There is also evidence that, in the worst case (and assuming that $\tau = O(n)$), the complexity result is near optimal. We give an informal argument: There exist polynomials $P$ of degree $n$ with integer coefficients of bit size $\tau$ that have distinct real roots at pairwise distance $2^{-\Omega(n \tau)}$ (e.g., Mignotte polynomials). For any given isolating intervals for these roots, the absolute value of $P$ at the endpoints of the intervals is upper bounded by $2^{-\Omega(n \tau)}$. Now, proving that an interval is isolating for a root implies that we can prove that the signs of $P$ at the endpoints differ. However, for determining the sign of $P$ at a point $x_0$ with $|P(x_0)| = 2^{-\Omega(n \tau)}$, we need to evaluate $P$ with an absolute precision of at least $\Omega(n \tau)$. In general, polynomial evaluation needs $\Omega(n)$ arithmetic operations, and thus the total number of bit operations needed to compute isolating intervals should be lower bounded by $\tilde{O}(n^2 \tau)$.

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

---

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

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

5The $\tilde{O}$-notation suppresses polylogarithmic factors, i.e., $\tilde{O}(T) = O(T \log T)^k$, where $k$ is an arbitrary, but fixed integer.
Theorem. 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 \( \kappa \) be an arbitrary positive integer. Computing isolating intervals of size less than \( 2^{-\kappa} \) for all real roots needs a number of bit operations bounded by
\[
\tilde{O}(n \cdot (\kappa + n^2 + n \log \text{Mea}(P) + \log M(\text{Disc}(P)^{-1}))).
\]
The coefficients of \( P \) have to be approximated to
\[
\tilde{O}(\kappa + n + \tau_P + \max_i (n \log M(z_i) + \log M(P'(z_i)^{-1})))
\]bits after the binary point. For a square-free polynomial \( P \) with integer coefficients of size less than \( 2^\tau \), computing isolating intervals of size less than \( 2^{-\kappa} \) for all real roots needs \( \tilde{O}(n(n^2 + n\tau + \kappa)) \) bit operations.

The complexity of the root refinement algorithm is \( \tilde{O}(n\kappa) \) for large \( \kappa \). This is optimal up to logarithmic factors, as the size of the output is \( \Omega(n\kappa) \). The complexity matches the complexity shown in [20], and when considered as a function in \( \kappa \) only, it also matches the complexity as shown in the full version [17] of [16] and as announced in [32].

1.1 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.

Many algorithms for approximate factorization and root isolation are known, see [11] for a survey. 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 [6, 8, 9, 25], the Bolzano method [5, 28], the Sturm method [7], the continued fraction method [2, 31, 33], and the splitting circle method [30, 23].

Among the subdivision methods, the splitting circle method is asymptotically the best. It was introduced by Schönhage [30] and later considerably refined by Pan [23]. Pan’s algorithm computes an approximate factorization and can also be used to isolate all roots of a polynomial. For integer polynomials, it isolates all roots with \( \tilde{O}(n^2\tau) \) bit operations. Pan’s factorization algorithm is also a key subroutine in a recent algorithm [20] for isolating all roots of a complex polynomial within the time bound of our main theorem. Unfortunately, Pan’s algorithm is quite complex, and it needs to compute all complex roots at the same time. It has not yet been implemented. A “proof of concept” implementation of the splitting circle method in the computer algebra system Pari/GP is available [12].

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 [13, 25, 15] 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 \( 2^\tau \) in absolute value, see [10]. 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 $\Omega(n^3\tau)$ and the numbers to be handled grow to integers of length $\Omega(n^2\tau)$ bits.

Johnson and Krandick [14] and Rouillier and Zimmermann [25] 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. Note that the correctness of 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. Eigenwillig et al. [9] were the first to describe a Descartes method that has no need for exact arithmetic. They describe a method for polynomials with real coefficients given through oracles. The oracle may be asked to give arbitrarily good approximations of the coefficients. We remarked above that the Descartes method uses the basic arithmetic operations and the sign-test. How can one accomplish a sign-test if coefficients are only approximately known? 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 $P$. They propose to choose the split point randomly or deterministically among sufficiently many candidates and to accept only split points where the polynomial does not vanish. This fact can be established relatively cheaply with approximate arithmetic. The choice of a split point where the polynomial has a large absolute value has a nice consequence for the sign change computation. Namely, it can also be carried out with approximate arithmetic. We give more details in Section 2.4. The algorithm by Eigenwillig et al. isolates the real roots of a square-free real polynomial $P(x) = P_n x^n + \ldots + P_0$ with root separation $6$, coefficients $|P_n| \geq 1$, and $|P_i| \leq 2^7$, 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 [26] gave a variant of the Descartes method for integer polynomials that uses approximate arithmetic with a working precision of only $O(n\tau)$ bits. This leads to a bit complexity of $O(n^3\tau^2)$; the recursion tree has size $O(n(\tau + \log n))$, there are $O(n)$ arithmetic operations per node, and arithmetic on numbers of length $O(n\tau)$ bits is required. We borrow from [9] 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 [18, 19]. We describe the details in Section 2.2. We also borrow from [9] how $v_I = 0$ or $v_I = 1$ can be checked efficiently using approximate arithmetic. We combine it with the efficient approximate Taylor shift computation from [19], which is based on fast approximate multipoint evaluation.

The recursion tree of the Descartes method may have size $\Omega(n\tau)$, for instance, when $P$ is a Mignotte polynomial with two distinct roots of distance $2^{-\Omega(n\tau)}$. However, all but $O(n)$ nodes of the tree have the property that one child is immediately discarded by Descartes’ rule of signs. In other words, large subdivision trees must have long chains of nodes, where the interval that is split off is immediately discarded. There are only $O(n)$ nodes where both children are subdivided further. The effect of long chains are an indication of clustered roots. Assume the existence of a cluster of roots with small diameter. Once an interval containing the cluster and no further roots is determined, only trivial splits happen until the width of the interval is essentially equal to the diameter of the cluster. Sagraloff [27] showed how to traverse such chains more efficiently by combining Descartes’ rule of signs, bisection, and Newton iteration. More precisely, his algorithm always attempts to refine intervals by a Newton step, and only if the Newton step fails, does the algorithm fall back on bisection. As a consequence, quadratic convergence towards the real roots is achieved in most iterations. His method reduces the size of the recursion tree to $O(n \log(1/\tau^n))$, which is optimal up to logarithmic factors.\footnote{The root separation of a polynomial is the minimal distance between two roots} The method only applies to polynomials with \footnote{As there might be $n$ real roots, $n$ is a trivial lower bound on the worst-case tree size.}
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 $O(n)$, there are $O(n)$ arithmetic operations per node, and arithmetic is on numbers of amortized length $\tilde{O}(n\tau)$ bits.

We borrow from this paper the idea of combining Newton and bisection steps. Several new ideas are necessary, e.g., the algorithm in [27] 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 $v_I$ and hence have to estimate the size of a cluster differently.

The bit complexity of our new algorithm is $O(n^3 + n^2\tau)$ for integer polynomials. The size of the recursion tree is $\tilde{O}(n)$ 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 due to the use of approximate multipoint evaluation and approximate Taylor shift.

Root refinement is the process of computing better approximations once the roots are isolated. In [18, 20, 17, 32], algorithms have been proposed which scale like $\tilde{O}(n^\kappa)$ for large $\kappa$. 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, no details are given in [32] when using multipoint evaluation.

1.2 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

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

We fix the following notations.

**Definition 1.**

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 $\sigma_P := \min_i \sigma_i$.
6. $\Gamma_P := M(\log \max_i |z_i|)$ denotes the logarithmic root bound of $P$, and
7. $\text{Mea}(P) := |P_n| \cdot \prod_{i=1}^n M(|z_i|)$ denotes the Mahler Measure of $P$.\hspace{1cm} (8)
We introduce the notions multipoint (Definition 6) and admissible point (Definition 4). A point \( x^* \) in a set \( X \) is admissible if \( |P(x^*)| \geq \frac{1}{4} |P(x)| \) for all \( x \in X \). We show how to efficiently compute an admissible point in a multipoint (Corollary 8) and derive a lower bound on the value of \( P \) at such a point. Corollary 8 is our main tool for choosing subdivision points.

**Theorem 2.** Let \( P \) be a polynomial as defined in (1), \( x_0 \) be an arbitrary real point, and \( L \) be an arbitrary positive integer.

(a) Computing an approximation \( \tilde{y}_0 \) of \( y_0 := P(x_0) \) with \( |y_0 - \tilde{y}_0| \leq 2^{-L} \) needs a number of bit operations bounded by

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

For this, the coefficients of \( P \) as well as the point \( x_0 \) have to be approximated to \( O(\tau_P + n \log M(|x|) + L + \log n) \) bits after the binary point.

(b) Suppose \( y_0 \neq 0 \). Then computing an integer \( t \) with \( 2^{t-1} \leq |y_0| \leq 2^{t+1} \) needs

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

bit operations. The computation can be carried out with fixed-precision arithmetic with a precision of \( O(\tau_P + n \log M(x) + L + \log n) \) bits.

**Proof.** Part (a) follows directly from [16, 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 \( O(\tau_P + n \log M(x) + L + \log n) \) bits.

\( \tilde{O} \) 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 \( \ell \in \mathbb{Z}_{\geq 0} \).

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 \( I \).\(^8\)

We assume the existence of an oracle that provides arbitrary good approximations of the polynomial \( P \). Let \( L \geq 1 \) be an integer. We call a polynomial \( \tilde{P} = \tilde{P}_n x^n + \ldots + \tilde{P}_0 \), with \( \tilde{P}_i = s_i \cdot 2^{-(L+1)} \) and \( s_i \in \mathbb{Z} \), an absolute \( L \)-approximation of \( P \) if \( |\tilde{P}_i - P_i| \leq 2^{-L} \) for all \( i \). 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 frequently use the phrase “the coefficients of \( P \) need to be approximated to \( L \) bits after the binary points” instead of “the algorithm requires an absolute \( L \)-approximation of \( P \)”.

We have \( \tau_P \leq M(\log(2^n \cdot \text{Mea}(P))) \leq M(n + n\Gamma_p) = n(1 + \Gamma_P) \leq 2n\Gamma_P \). According to [20, Theorem 1] (or [26, Section 6.1]), we can compute an integer approximation \( \tilde{\Gamma}_P \) of \( \Gamma_P \) with

\[
\Gamma_P + 1 \leq \tilde{\Gamma}_P \leq \Gamma_P + 8 \log n + 1
\]

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

\[
\Gamma_P + 1 \leq \tilde{\Gamma}_P \leq \Gamma \leq 2 \cdot \tilde{\Gamma}_P \leq 2 \cdot (\Gamma_P + 8 \log n + 1).
\]

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 6) and admissible point (Definition 4). A point \( x^* \) in a set \( X \) is admissible if \( |P(x^*)| \geq \frac{1}{4} |P(x)| \) for all \( x \in X \). We show how to efficiently compute an admissible point in a multipoint (Corollary 8) and derive a lower bound on the value of \( P \) at such a point. Corollary 8 is our main tool for choosing subdivision points.

**Theorem 2.** Let \( P \) be a polynomial as defined in (1), \( x_0 \) be an arbitrary real point, and \( L \) be an arbitrary positive integer.

(a) Computing an approximation \( \tilde{y}_0 \) of \( y_0 := P(x_0) \) with \( |y_0 - \tilde{y}_0| \leq 2^{-L} \) needs a number of bit operations bounded by

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

For this, the coefficients of \( P \) as well as the point \( x_0 \) have to be approximated to \( O(\tau_P + n \log M(|x|) + L + \log n) \) bits after the binary point.

(b) Suppose \( y_0 \neq 0 \). Then computing an integer \( t \) with \( 2^{t-1} \leq |y_0| \leq 2^{t+1} \) needs

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

bit operations. The computation can be carried out with fixed-precision arithmetic with a precision of \( O(\tau_P + n \log M(x) + L + \log n) \) bits.

**Proof.** Part (a) follows directly from [16, 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 \( O(\tau_P + n \log M(x) + L + \log n) \) bits.

---

\( \dagger \) 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, $2^{t-1} \leq (3/4)2^{-1/2}t^2 \leq (3/4)|\tilde{y}_0| \leq |y_0| \leq (5/4)|\tilde{y}_0| \leq (5/4)2^{1/2}2^t \leq 2^{t+1}$. 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 [18, 19] that the cost for approximating evaluating a polynomial of degree $n$ at $N = O(n)$ points is comparable to the cost of approximating evaluating it at a single point.

**Theorem 3 ([18, 19]).** Let $P$ be a polynomial as in (1), let $x_1, \ldots, x_N$ be arbitrary real points with $N = O(n)$, and let $L$ be an arbitrary positive integer. Then, computing approximations $\tilde{y}_i$ of $y_i := P(x_i)$ with $|y_i - \tilde{y}_i| \leq 2^{-L}$, $i = 1, \ldots, N$, needs a number of bit operations bounded by

$$\tilde{O}(n + \tau_P + n \log M(\max_i |x_i|)) + L + n \log n \text{ bits after the binary point.}$$

For this computation, the coefficients of $P$ as well as the points $x_i$ have to be approximated to $O(\tau_P + n \log M(\max_i |x_i|) + L + n \log n)$ bits after the binary point.

Fast approximate multipoint evaluation provides an efficient method for selecting a point $x_i$ from a given set $X = \{x_1, \ldots, x_N\}$ of points where $|P(x_i)|$ is close to maximal: We consider $L = 1, 2, 4, 8, \ldots$ and approximate all values $|P(x_i)|$ to a precision of $L$ bits after the binary point until, for at least one $i$, we obtain an approximation $2^{t_i}$ with $t_i \in \mathbb{Z}$ and $2^{t_i-1} \leq |P(x_i)| \leq 2^{t_i+1}$. Now, let $i_0$ be such that $t_{i_0}$ is maximal; then, it follows that $2^{t_{i_0}-1} \leq \lambda := \max_i |P(x_i)| \leq 2^{t_{i_0}+1}$. An argument similar to the one as in the proof of Part (b) in Lemma 2 now yields the following result:

**Definition 4.** Let $X := \{x_1, \ldots, x_N\}$ be a set of $N = O(n)$ arbitrary real points. We call a point $x^* \in X$ admissible with respect to $X$ (or just admissible if there is no ambiguity) if $|P(x^*)| \geq \frac{1}{2} \cdot \max_i |P(x_i)|$.

**Lemma 5.** Let $X := \{x_1, \ldots, x_N\}$ be a set of $N = O(n)$ arbitrary real points. We can determine 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 $\tilde{O}(n + \tau_P + n \log M(\max_i |x_i|) + \log M(\lambda^{-1}))$ bit operations. The coefficients of $P$ and the points $x_i$ have to be approximated to $O(n + \tau_P + n \log M(\max_i |x_i|) + \log M(\lambda^{-1}))$ bits after the binary point.

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 lower bound $\lambda$ 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 6.** For an arbitrary real point $m$ and an arbitrary real positive value $\epsilon$, the $(m, \epsilon)$-multipoint $m|\epsilon|$ is defined as

$$m|\epsilon| := \{m_i := m + (i - \lceil n/2 \rceil) \cdot \epsilon \mid i = 0, \ldots, 2 \cdot \lceil n/2 \rceil\}. \quad (4)$$
Lemma 7. Let \( m \) be an arbitrary real point, let \( \epsilon \) be an arbitrary real positive value, and let \( K \) be a positive real with \( K \geq 2 \cdot \lceil n/2 \rceil \). If the disk \( \Delta := \Delta_{K,\epsilon}(m) \) with radius \( K \cdot \epsilon \) and center \( m \) contains at least two roots of \( P \), then each admissible point \( m^\ast \in m[\epsilon] \) satisfies

\[
|P(m^\ast)| > 2^{-4n-1} \cdot K^{-\mu(\Delta)} \cdot \sigma_i \cdot |P'(z_i)| \quad \text{for all roots } z_i \in \Delta,
\]

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

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 \( \epsilon \), 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 us now consider an arbitrary but fixed root \( z_i \in \Delta \). For any different root \( z_j \in \Delta \), we have \( |z_i - z_j| > |m_{i_0} - z_j| \leq 2(K \epsilon/\epsilon) = 4K, \) and, for any root \( z_j \not\in \Delta \), we have \( |z_i - z_j| > |m_{i_0} - z_j| \leq 2(K - [n/2]) \not\leq 4 \). Hence, it follows that

\[
|P(m_{i_0})| = |P_n| \cdot |m_{i_0} - z_i| \cdot \prod_{j \neq i} |m_{i_0} - z_j| > |P_n| \cdot \frac{\epsilon}{2} \cdot (4K)^{-\mu(\Delta)} \cdot 4^{-n+\mu(\Delta)} \cdot \prod_{j \neq i} |z_i - z_j|
\]

\[
= |P'(z_i)| \cdot \frac{\epsilon}{2n} \cdot 4^{-n} \cdot K^{-\mu(\Delta)} = 2^{-(\log n + 1) - 2n} \cdot \epsilon \cdot K^{-\mu(\Delta)} \cdot |P'(z_i)|
\]

\[
> 2^{-2n-\log n - 2} \cdot K^{-\mu(\Delta)} \cdot \sigma_i \cdot |P'(z_i)|,
\]

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

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

Corollary 8. Let \( m \) be an arbitrary real point, let \( \epsilon \) be an arbitrary real positive value, and let \( K \) be a positive real with \( K \geq 2 \cdot \lceil n/2 \rceil \), and assume that the disk \( \Delta := \Delta_{K,\epsilon}(m) \) contains at least two roots of \( P \). Then, for each admissible point \( m^\ast \in m[\epsilon] \),

\[
|P(m^\ast)| > 2^{-4n-1} \cdot K^{-\mu(\Delta)} \cdot \sigma_i \cdot |P'(z_i)| \quad \text{for all roots } z_i \in \Delta.
\] (5)

An admissible point can be computed with a number of bit operations bounded by

\[
\tilde{O}(n(\mu(\Delta) \cdot \log K + n + \tau_P + n \log M(|m| + n \epsilon) + \log M(\max_{z_i \in \Delta} |\sigma_i \cdot |P'(z_i)|)))).
\] (6)

The coefficients of \( P \) and the points \( m_i \) have to be approximated to a precision of

\[
O(\mu(\Delta) \cdot \log K + n + \tau_P + n \log M(|m| + n \epsilon) + \log M(\max_{z_i \in \Delta} |\sigma_i \cdot |P'(z_i)|))
\]

bits after the binary point.

Corollary 8 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 \( m^\ast \in m[\epsilon] \) and are guaranteed that \( |P(m^\ast)| \) has at least the value stated in (5). The fact that \( |P| \) is reasonably large at \( m^\ast \) will play a crucial role in the analysis of our algorithm, cf. Theorem 26.

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 [8, 9, 27]);
Figure 1: For any $k$, $0 \leq k \leq n$, the Obreshkoff disks $\overline{C}_{k}$ and $\overline{C}_{k}$ 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 $\overline{C}_{k} \cap \overline{C}_{k}$, and the Obreshkoff area $A_{k}$ is the interior of $\overline{C}_{k} \cup \overline{C}_{k}$. 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 $L_{0} \subseteq \ldots \subseteq L_{k} \subseteq L_{0}$ 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 $\overline{C}_{0}$ coincide. They have their centers at the midpoint of $I$. The circles $\overline{C}_{1}$ and $\overline{C}_{1}$ are the circumcircles of the two equilateral triangles having $I$ as one of their edges. We call $\overline{A}_{0} = \Delta(I)$ and $A_{1}$ the one-circle and the two-circle regions for $I$, respectively.

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 $I = (a, b) \subseteq \mathbb{I} = (-2^{q}, 2^{q})$, we use Descartes’ rule of signs: For an arbitrary 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$^9$ 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

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

and let $v_{I} := \text{var}(P, I) := \text{var}(p_{I,0}, \ldots, p_{I,n})$ be defined as the number of sign variations in the 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 9 ([21, 22]).** Let $I = (a, b)$ be an open interval and $v_{I} = \text{var}(P, I)$. 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 at most $k$ roots (counted with multiplicity) of $P$, then $v_{I} \leq k$. In particular,

$$\# \text{ of roots of } P \text{ in } L_{n} \leq v_{I} = \text{var}(P, I) \leq \# \text{ of roots of } P \text{ in } A_{n}.$$

---

$^9$Zero entries are not considered. For instance, $\text{var}(-1,0,0,2,0,-1) = \text{var}(-1,2,-1) = 2$. 

10
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, 8]). Theorem 9 implies that if the one-circle region \( A_0 = \Delta(I) \) of \( I \) contains a root \( z_i \) with separation \( \sigma_i > 2w(I) = 2(b - a) \), then this root must be real and \( v_I = 1 \). Namely, the condition on \( \sigma_i \) 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 \( \text{var}(P, I) \); e.g., see [8, Corollary 2.27] for a self-contained proof:

**Theorem 10** ([29]). Let \( I \) be an interval and \( I_1 \) and \( I_2 \) be two disjoint subintervals of \( I \). Then,

\[
\text{var}(P, I_1) + \text{var}(P, I_2) \leq \text{var}(P, I).
\]

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

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

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 (7) and the entries of \( I \) and the entries of \( B \). For a self-contained proof, we refer to [8].

**Lemma 11.** 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 (7). It holds that

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

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 (7). This will turn out to be useful in the next section, where we review results from [9] 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 \( \text{var}(P, I) = 0 \) (\( \text{var}(P, I) = 1 \)), 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 15 and 18.

#### 2.4.1 The case \( \text{var}(P, I) = 0 \)

Consider the following Lemma, which follows directly from [9, Lemma 5] and its proof:

**Lemma 12.** 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 an arbitrary 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 \( P(x) = \sum_{i=0}^{n} b_i B^n_i(a,m)(x) \) and \( P(x) = \sum_{i=0}^{n} b_i B^n_i[m,b](x), \) respectively. Suppose that \( \text{var}(P, I) = 0. \) Then, \( \text{var}(P, I') = \text{var}(P, I'') = 0, \) 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 11 now yields:

**Corollary 13.** Let \( I, I', \) and \( I'' \) be intervals as in Lemma 12. Furthermore, let

\[
L_{1,0} := \log M(\min(|P(a)|, |P(b)|)^{-1}) + 2(n + 1) + 1,
\]

and let \( \sum_{i=0}^{n} \tilde{p}_{t,i} \cdot x^i \) and \( \sum_{i=0}^{n} \tilde{p}_{t'',i} \cdot x^i \) be absolute \( L_{1,0} \)-bit approximations of \( P_t \) and \( P_{t''} \), respectively. If \( \text{var}(P, I) = 0, \) then \( \text{var}(\tilde{p}_{t,0}, \ldots, \tilde{p}_{t,n}) = 0, \) and \( |\tilde{p}_{t,i}, \tilde{p}_{t'',i}| > 2^{-L_{1,0}} \) for all \( i = 0, \ldots, n. \)

**Proof.** Suppose that \( \text{var}(P, I) = 0, \) then Lemma 12 yields \( |b'_i|, |b''_i| > \min(|P(a)|, |P(b)|) \cdot 4^{-(n+1)} \geq 2 \cdot 2^{-L_{1,0}} \) for all \( i = 0, \ldots, n, \) and, in addition, all coefficients \( b'_i \) and \( b''_i \) have the same sign. Since \( \tilde{p}_{t,i} = (0)^i \cdot b_{\alpha-i} \) and \( \tilde{p}_{t'',i} = (0)^i \cdot b''_{\alpha-i}, \) it follows that the coefficients \( p_{t,i} \) and \( p_{t'',i} \) also have absolute value larger than \( 2 \cdot 2^{-L_{1,0}}. \) Thus, \( |\tilde{p}_{t,i}, |p_{t,i}| > 2^{-L_{1,0}} \) since \( |p_{t,i} - \tilde{p}_{t,i}| \leq 2^{-L_{1,0}} \) and \( |p_{t'',i} - \tilde{p}_{t'',i}| \leq 2^{-L_{1,0}} \) for all \( i. \) In addition, all coefficients \( p_{t,i} \) and \( p_{t'',i} \) have the same sign because this holds for their exact counterparts. \( \square \)

The above corollary allows one to discard an interval \( I \) by using approximate arithmetic with a precision that is directly related to the absolute values of \( P \) at the endpoints of \( I. \) More precisely, we consider the following exclusion test which applies to intervals \( I = (a, b) \) with \( P \neq 0 \) and \( P(b) \neq 0: \)

**0-Test:** Compute approximations \( 2^t_a \) and \( 2^t_b \) for \( |P(a)| \) and \( |P(b)| \) with \( t_a, t_b \in \mathbb{Z}, \) and \( 2^t_a - 1 \leq |P(a)| \leq 2^{t_a+1}, \) and \( 2^t_b - 1 \leq |P(b)| \leq 2^{t_b+1}. \) If follows that \( L_{1,0} \leq L := M(\min(t_a - 1, t_b - 1)) + 2(n + 1) + 1 \leq L_{1,0} + 2. \) Now, for \( I' = (a, m(I)) \) and \( I'' = (m(I), b), \) compute absolute \( L \)-bit approximations \( \tilde{P}_I = \sum_{i=0}^{n} \tilde{p}_{t,i} \cdot x^i \) and \( \tilde{P}_I = \sum_{i=0}^{n} \tilde{p}_{t,i} \cdot x^i \) of the polynomials \( P_I \) and \( P_{I''}, \) respectively. If all approximate coefficients \( \tilde{p}_{t',i} \) and \( \tilde{p}_{t'',i} \) have the same sign (i.e. \( \text{var}(\tilde{p}_{t',0}, \ldots, \tilde{p}_{t',n}) = 0 \) and if all of them have absolute value larger than \( 2^{-L}. \) then \( \text{var}(P, I') = \text{var}(P, I'') = 0 \) and, thus, \( I \) contains no root of \( P. \) \( 11 \) In this case, we say that the 0-Test succeeds.

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 14.** Let \( I = (a, b) \) be an arbitrary interval, and let \( L \) be an arbitrary positive integer. Then, we can compute an absolute \( L \)-bit approximation \( \tilde{P}(x) = \sum_{i=0}^{n} \tilde{p}_{l,i} \cdot x^i \) of \( P(x) = \sum_{i=0}^{n} p_{l,i} x^i \) with a number of bit operations bounded by

\[
\tilde{O}(n(n + \tau_P + n \log M(a) + n \log M(b) + L)).
\]

For this computation, the coefficients of \( P \) and the endpoints of \( I \) have to be approximated to \( O(n + \tau_P + n \log M(a) + n \log M(b) + L) \) bits after the binary point.

---

\(^{10}\) According to Lemma 2, we can compute \( t_a \) and \( t_b \) with a number of bit operations bounded by \( \tilde{O}(n(\tau_P + n \log M(a) + n \log M(b) + M(P(a)^{-1}) + \log M(P(b)^{-1}))). \)

\(^{11}\) Notice that \( P(m) \neq 0 \) because \( |P(m)| = |\tilde{p}_{l,m}| > 0. \)
Proof. The computation of $P_1$ 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 $P_2(x) := P(a + w(I) \cdot x)$. 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 $P_I$, we proceed as follows: Let $L_1$ be a positive integer, which will be specified later. According to [19, Theorem 14] (or [30, Theorem 8.4]), we can compute an absolute $L_1$-bit approximations $P_I$ of $P_1$ with $O(n(n + \tau_P + n \log M(a) + L_1))$ bit operations, where we used that the coefficients of $P$ have absolute value of size $2^{2^r}$ or less. For this step, the coefficients of $P$ as well as the endpoint $a$ have to be approximated to $O(n + \tau_P + n \log M(a) + L_1)$ bits after the binary point. The coefficients of $P_1$ have absolute value less than $2^{n + \tau_P} M(a)^n$, and thus, the coefficients of $P_I$ have absolute value less than $2^{n + \tau_P} M(a)^n + 1 < 2^{n + \tau_P} M(a)^n$. Computing $w(I)^i$ for all $i = 0, \ldots, n$ to an absolute error of $L_1$ bits after the binary point takes $O(n(n \log M(w(I)) + L_1)) = O(n(n \log M(a) + n \log M(b) + L_1))$ bit operations. This yields an approximation $P_2$ of $P_3$ to $L_2 := L_1 - n - \tau_P - n \log M(a)$ bits after the binary point.

The coefficients of $P_2$ have absolute value less than $2^{n + \tau_P} M(a)^n M(w(I))^n$. Reversing the coefficients of $P_3$ trivially yields an absolute $L_2$-bit approximation $P_4$ of $P_3$. For the last step, we again apply [19, 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 $O(L + n + \tau_P + n \log M(a) + n \log M(w(I)))$. The cost for this computation is bounded by $O(n L_4)$ bit operations. Hence, it suffices to start with an integer $L_1$ of size $O(L + n + \tau_P + n \log M(a) + n \log M(w(I)))$. 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 $O(n(n + \tau_P + n \log M(a) + n \log M(b) + L_1))$ 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 15.** For an interval $I = (a, b)$, the 0-Test needs no more than

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

(11)

bit operations. The coefficients of $P$ and the endpoints of $I$ have to be approximated to $O(n + \tau_P + n \log M(a) + n \log M(b) + \log M(\min(|P(a)|, |P(b)|)^{-1})))$ bits after the binary point.

### 2.4.2 The case $\var(P, I) = 1$

We need the following result, which follows directly from [9, Lemma 6] and its proof.

**Lemma 16.** With the same definitions as in Lemma 12, suppose that $\var(P, I) = 1$ and $P(m) \neq 0$. Then,

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

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

Again, combining the latter result with Lemma 11 yields the following result, whose proof is completely analogous to the proof of Corollary 17.

**Corollary 17.** With the same definitions as in Lemma 12 and Lemma 16, let

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

(12)
and let $\sum_{i=0}^{n} \tilde{p}_{r,i} \cdot x^i$ and $\sum_{i=0}^{n} \tilde{p}_{\nu,i} \cdot x^i$ be absolute $L_{1,1}$-bit approximations of $P_r$ and $P_{\nu}$, respectively. Suppose that $\text{var}(P, I) = 1$ and $P(m) \neq 0$. Then, it follows that $|\tilde{p}_{r,i}|, |\tilde{p}_{\nu,i}| > 2^{-L_{1,1}}$ for all $i = 0, \ldots, n$, and, in addition, $\text{var}(\tilde{p}_{r,0}, \ldots, \tilde{p}_{r,n}) = 1$ (and $\text{var}(\tilde{p}_{\nu,0}, \ldots, \tilde{p}_{\nu,n}) = 0$) or $\text{var}(\tilde{p}_{r,0}, \ldots, \tilde{p}_{r,n}) = 1$ (and $\text{var}(\tilde{p}_{\nu,0}, \ldots, \tilde{p}_{\nu,n}) = 0$).

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 $P(b) \neq 0$:

**1-Test:** Compute approximations $2^t a$ and $2^t b$ for $|P(a)|$ and $|P(b)|$ with $t_a, t_b \in \mathbb{Z}$ and $2^{t_a-1} \leq |P(a)| \leq 2^{t_a+1}$, $2^{t_b-1} \leq |P(b)| \leq 2^{t_b+1}$. For $\epsilon := w(I) \cdot 2^{-[\log n+2]} \leq \frac{w(I)}{n}$, compute (using the method from Lemma 5) an admissible $m^* \in m(I)[x]$ and an integer $t$ with

$$2^{t+1} \geq \max_i |P(m_i)| \geq |P(m^*)| \geq 2^{-t-1}.$$  

With $m := m^*$,\(^{12}\) it follows that $L_{1,1} \leq L := \mathcal{M}(-\min(t_a - 1, t_b - 1, t-1)) + 4n + 2 \leq L_{1,1} + 2$. Now, compute absolute $L$-bit approximations for the polynomials $P_r$ and $P_{\nu}$, where $I' = (a, m^*)$ and $I'' = (m^*, b)$. If all approximate coefficients $\tilde{p}_{r,i}$ and $\tilde{p}_{\nu,i}$ have absolute value larger than $2^{-L}$, and if $\text{var}(\tilde{p}_{r,0}, \ldots, \tilde{p}_{r,n}) = 1$ and $\text{var}(\tilde{p}_{\nu,0}, \ldots, \tilde{p}_{\nu,n}) = 0$, then $I'$ isolates a root of $p$, whereas $I''$ contains no root of $I'$. If $|\tilde{p}_{r,i}| > 2^{-L}$ and $|\tilde{p}_{\nu,i}| > 2^{-L}$ for all $i$, and if $\text{var}(\tilde{p}_{r,0}, \ldots, \tilde{p}_{r,n}) = 0$ and $\text{var}(\tilde{p}_{\nu,0}, \ldots, \tilde{p}_{\nu,n}) = 1$, then $I''$ isolates a root of $p$, whereas $I'$ contains no root of $I'$. In each of the latter two cases, we say that the 1-Test succeeds.

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

**Corollary 18.** 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 $I$ needs a number of bit operations bounded by

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

The coefficients of $P$ and the endpoints of $I$ have to be approximated to $O(n + \tau_R + n \log M(a) + n \log M(b) + \log M(\min(|P(a)|, |P(b)|, \lambda)^{-1})$ bits after the binary point.

### 2.5 Useful Inequalities

**Lemma 19.** 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,

$$\text{Mea}(p) \leq \|p\|_2 \leq \|p\|_1 \leq (n + 1)2^{\tau_p}$$  
$$\sigma_p \geq \sqrt{\text{Disc}(p)} \|p\|_2^{-n+1} n^{-(n+2)/2}$$  
$$|\text{Disc}(p)| \leq n^n (\text{Mea}(p))^{2n-2} \leq n^n \|p\|_2^{2n-2}$$  
$$\log |p'(z_i)| = O(\log n + \tau_p + n \log M(z_i))$$  
$$\sum_i \log M(p'(z_i)^{-1}) = O(n \tau_p + n^2 + n \log \text{Mea}(p) + |\log \text{Disc}(p)|^{-1})$$  
$$\text{Mea}(p(x - z_i)) \leq 2^{\tau_p} 2^{n+1} M(z_i)^n$$  
$$\tau_p = O(n + \log \text{Mea}(p)).$$  

\(^{12}\)Notice that each point $m_i$ is contained in $[m(I) - w(I), m(I) + w(I)]$ since $[n/2] \cdot 2^{-[\log n + 2]} < 1/4$. Thus, we can use Lemma 16 with $m = m^*$.  

14
Proof. [34, Lemma 4.14] establishes (14). [34, Corollary 6.29] establishes (15) and (16). For (17), observe

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

(18) follows from

\[ \sum_i \log M(p'(z_i))^{-1} = \log \prod_i M(p'(z_i)) = \log \prod_i \frac{M(p'(z_i))}{|p'(z_i)|} = O(\log n + n^2 + n \log Mea(p) + \log |\text{Disc}(p)|^{-1}). \]

For (19), we use Mea(p) ≤ ∥p∥₁ and p(x − z_i) = \(\sum_{0 \leq k \leq n} (x^k \sum_{k \leq j \leq n} p_j \binom{j}{k} (-z_i)^{j-k})\), and hence,

\[ \|p(x - z_i)\|_1 \leq \sum_{0 \leq k \leq n} \sum_{k \leq j \leq n} p_j \binom{j}{k} M(z_i)^{j-k} \leq 2^r M(z_i)^n \sum_{0 \leq j \leq n} \sum_{k \leq j} \binom{j}{k} \]

\[ \leq 2^r M(z_i)^n \sum_{0 \leq j \leq n} 2^j \leq 2^r M(z_i)^n 2^{n+1}. \]

For (20), we first recall that \(\tau_p = log M(\|p\|_\infty)\). The coefficient \(p_i\) is given by

\[ p_i = \frac{p_n}{\prod_{I \subseteq \{1, \ldots, n\}, |I| = n-i} z_i}. \]

Thus

\[ |p_i| \leq |p_n| \left(\frac{n}{n-i}\right) \frac{\text{Mea}(p)}{|p_n|} \leq 2^n \text{Mea}(p). \]

\[ \square \]

### 3 The Algorithm

We are now ready for our algorithm ANewDsc for isolating the real roots of \(P\). We maintain a list \(A\) of active intervals,¹⁴ a list \(O\) of isolating intervals, and the invariant that the intervals in \(O\) are isolating and that each real root of \(P\) is contained in either an active or an isolating interval. We initialize \(O\) to the empty set and \(A\) to the interval \(I = (-2^7, 2^7)\), where \(\Gamma = 2^7\) is defined as in (3). 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 \(I\).

¹³Our algorithm is an approximate arithmetic variant of the algorithm NewDsc presented in [27]. 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”.

¹⁴In fact, \(A\) 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 \(A\) as a list of intervals only.
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.

In the presence of a cluster \( C \) of roots (i.e., a set of \( k := |C| \geq 2 \) 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 \( C \) by using Newton iteration. More precisely, if we consider a point \( \xi \), whose distance \( d \) to the cluster \( C \) 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

\[
\xi' := \xi - k \cdot \frac{P(\xi)}{P'(\xi)}
\]

to the cluster \( C \) is much smaller than the distance from \( \xi \) to \( C \).\(^{15}\) The distance \( d' \) of \( \xi' \) to the cluster is approximately \( d^2 \) if \( d < 1 \). Thus, we can achieve quadratic convergence to the cluster \( C \) by iteratively applying (21). Unfortunately, when running the subdivision algorithm, we neither know whether there actually exists a cluster \( C \) 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 \( \xi \), say \( \xi_1 \) and \( \xi_2 \). Let \( \xi'_i \), \( i = 1, 2 \), be the Newton iterates. For the correct value of \( k \), we should have \( \xi'_1 = \xi'_2 \). 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 \( N_I = 2^{2+n} \), with \( n_I \geq 1 \), which is the level of this interval. 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 list 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 \( N_I/N_I \)).

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:

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

\[
x_{n+1} - \alpha = x_n - \alpha - \frac{(x_n - \alpha)^kg(x_n)}{k(x_n - \alpha)^{k-1}g(x_n) + (x_n - \alpha)^k g'(x_n)}
\]

\[
= (x_n - \alpha)(1 - \frac{kg(x_n) + (x_n - \alpha)g'(x_n)}{kg(x_n) + (x_n - \alpha)g'(x_n)},
\]

and hence, we have quadratic convergence in an interval around \( \alpha \).
(T0) We apply the 0-Test to \( I \). In case of success, we know that \( I \) contains no root and discard it.

(T1) Otherwise, we apply the 1-Test to \( I \). In case of success, it returns an isolating interval\(^{16} \) \( I' \) with \( \frac{w(I)}{4} \leq w(I') \leq \frac{3w(I)}{4} \). We add \( I' \) to \( \mathcal{O} \).

(Q) If both of the former tests fail, we apply the Boundary-Test as well as the Newton-Test to \( I \).

If one of these tests succeed, we obtain an interval \( I' \subseteq I \), with \( \frac{w(I)}{5N_I} \leq w(I') \leq \frac{w(I)}{N_I} \), which contains all roots contained in \( I \). We add \((I', N_{I'})\) to \( \mathcal{A} \), where \( N_{I'} := N_I^2 \) (quadratic step).

(L) If all of the steps (T0), (T1), and (Q) fail, we compute an admissible point\(^{17} \) \( m^* \in m(I)[\frac{w(I)}{2(1 + \log n)}] \) and add \((I', N_{I'})\) and \((I'', N_{I''})\) to \( \mathcal{A} \), where \( I' = (a, m^*) \) and \( I'' = (m^*, b) \), and \( N_{I''} := \max(4, \sqrt{N_{I'}}) \) (linear step).

We continue until the list \( \mathcal{A} \) becomes empty. Then, we return the list \( \mathcal{O} \) of isolating intervals.

If we succeed in Step (Q), 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 \( w(I) < \frac{w(I)}{2} \) satisfies \( \text{var}(P, I) = 0 \) or \( \text{var}(P, I) = 1 \), either the 0-Test or the 1-Test succeeds for \( I \). This proves termination (i.e., Step (T0) or Step (T1) 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 \( I = (-2^7, 2^7) \). 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 \( I \) into disjoint intervals \( I_k = (s_k, s_{k+1}) \), with \( k = 0, \ldots, 2 \cdot \gamma + 1 \) 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 \( k \):

\[
\begin{align*}
\min_{x \in I_k} |P(s_k^*)|, |P(s_{k+1}^*)| &> 2^{-8n \log n}, \quad \text{and} \\
\max_{x \in I_k} \log M(x) &\leq 2 \cdot (1 + \min_{x \in I_k} \log M(x)).
\end{align*}
\tag{22}
\]

The intervals \((-2^{2^7}, -2^{2^7 - 1}), (-2^{2^7 - 1}, -2^{2^7 - 2}), \ldots, (-2^{2^7}, 2^{2^7}), \ldots, (-2^{2^{\gamma - 1}}, -2^{2^{\gamma}})\) satisfy the second condition. In order to also satisfy the first, consider the points

\[
s_k := \begin{cases} -2^{2^7 - k} & \text{for } k = 0, \ldots, \gamma \\ +2^{k+1-\gamma} & \text{for } k = \gamma + 1, 2\gamma + 1 \end{cases}
\]

and corresponding multipoints \( M_k := s_k[2^{-2\log n}] \).

We compute admissible points \( s_k^* \in M_k \). For each index \( k \), there exists at least one point in \( M_k \) with distance at least \( \frac{1}{5n} \) or larger to all roots of \( P \). Thus, \(|P(s_k^*)| \geq |P_n| \cdot \left(\frac{1}{5n}\right)^n \geq 2^{-4n-n \log n} \geq 2^{-8n \log n} \). We define:

\[
I_k := (s_k^*, s_{k+1}^*) \quad \text{for } k = 0, \ldots, 2\gamma + 1.
\tag{23}
\]

It is easy to check that the second condition in (22) is also fulfilled for these intervals.

\(^{16}\)In fact, from the definition of the 1-Test, it even holds that \( \text{var}(P, I') = 1 \).

\(^{17}\)Notice that such a point has already been computed in the 1-Test in (T0), and thus, there is no need to repeat this computation.
3.2 The Newton-Test and the Boundary-Test

The Newton-Test and the Boundary-Test are the key to quadratic convergence. As input, it receives an arbitrary interval \( I = (a, b) \subseteq \mathbb{I} \) and an integer \( N_I = 2^{2^i} \), where \( n_I \geq 1 \) is an integer. In case of success, the test returns an interval \( I' \) with \( \frac{w(I)}{2N_I} \leq w(I') \leq \frac{w(I)}{N_I} \) 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 \( 2^{\log n_I + 10} \cdot N_I \cdot w(I) \) and center \( m(I) \) contains no further root of \( P \), see Lemma 20 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 have inserted footnotes that explain the rationale behind our choices. For this rationale, we assume the existence of a cluster \( C \) of \( k \) roots centered at some point \( \xi \in I \) with diameter \( d(C) \ll w(I) \) 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 20.

Newton-Test: Consider the points\(^{18} \xi_1 := a + \frac{1}{4} \cdot w(I), \xi_2 := a + \frac{1}{2} \cdot w(I), \xi_3 := a + \frac{3}{4} \cdot w(I), \) and let \( \epsilon := 2^{-5 \log n} \). For \( j = 1, 2, 3 \), compute admissible points

\[
\xi_j^* \in \xi_j[\epsilon \cdot w(I)]
\]

using the method from Lemma 5. These points define values \( v_j := \frac{P(\xi_j^*)}{P'(\xi_j^*)} \) as they appear in the Newton iteration (21) with \( \xi = \xi_j^* \).

For the three distinct pairs of indices \( j_1, j_2 \in \{1, 2, 3\} \) with \( j_1 < j_2 \), we perform the following computations in parallel: For \( L = 1, 2, 4, \ldots \), we compute approximations of \( P(\xi_{j_1}^*), P(\xi_{j_2}^*), P'(\xi_{j_1}^*), \) and \( P'(\xi_{j_2}^*) \) to \( L \) bits after the binary point. We stop doubling \( L \) for a particular pair \((j_1, j_2)\) if we can either verify that\(^{19} \)

\[
|v_{j_1}|, |v_{j_2}| > w(I) \quad \text{or} \quad |v_{j_1} - v_{j_2}| < \frac{w(I)}{4n}
\]

or that

\[
|v_{j_1}|, |v_{j_2}| < 2 \cdot w(I) \quad \text{and} \quad |v_{j_1} - v_{j_2}| > \frac{w(I)}{8n}.
\]

If (25) holds, we discard the pair \((j_1, j_2)\). Otherwise (i.e., (26) holds), we compute sufficiently good approximations of \( P(\xi_{j_1}^*), P(\xi_{j_2}^*), P'(\xi_{j_1}^*), \) and \( P'(\xi_{j_2}^*) \), such that we can derive an approximation \( \lambda_{j_1,j_2} \) of\(^{20} \)

\[
\lambda_{j_1,j_2} := \xi_{j_1}^* - \frac{\xi_{j_2}^* - \xi_{j_1}^*}{v_{j_1} - v_{j_2}} v_{j_1}
\]

\(^{18}\)At least two of the three points \( \xi_j \) (say \( \xi_1 \) and \( \xi_2 \)) have a distance from \( C \) that is large compared to the diameter of \( C \). In addition, their distances to all remaining roots are also large, and thus, the points \( \xi_i^* := \xi_i - k \cdot v_i \) and \( \xi_j^* := \xi_j - k \cdot v_j \) obtained from considering one Newton step have much smaller distances to the cluster \( C \) than the points \( \xi_1 \) and \( \xi_2 \). Note that \( k \) is not known to the algorithm at this point.

\(^{19}\)The meanings of the conditions (25) and (26) will become clear in the proof of Lemma 20.

\(^{20}\)The Newton iteration (21) with \( \xi = \xi_j^* \) for a \( k \)-fold root produces \( \xi_j' = \xi_j^* - kv_j \). Equating \( \xi_{j_1}' = \xi_{j_2}' \) yields 

\[-k = \frac{\xi_{j_2}^* - \xi_{j_1}^*}{v_{j_1} - v_{j_2}} \text{. Then, } \xi_{j_1}' \text{ and } \xi_{j_2}' \text{ are given by } 27.\]
with \( |\tilde{\lambda}_{j_1,j_2} - \lambda_{j_1,j_2}| \leq \frac{1}{12N_f}. \) If \( \tilde{\lambda}_{j_1,j_2} \not\in [a, b] \), we discard the pair \((j_1, j_2)\). Otherwise, let \( \ell_{j_1,j_2} := \frac{1}{w(4N_f)} \) \( |\tilde{\lambda}_{j_1,j_2}| - a \). Then \( \ell_{j_1,j_2} \in \{0, \ldots, 4N_f\} \). We further define

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

If \( a_{j_1,j_2} = a \), we set \( a^*_{j_1,j_2} := a \), and if \( b_{j_1,j_2} = b \), we set \( b_{j_1,j_2} := b \). For all other values for \( a_{j_1,j_2} \) and \( b_{j_1,j_2} \), we use the method from Lemma 5 to compute admissible points

\[
a^*_{j_1,j_2} \in a_{j_1,j_2}[e \cdot \frac{w(I)}{N_f}] \quad \text{and} \quad b^*_{j_1,j_2} \in b_{j_1,j_2}[e \cdot \frac{w(I)}{N_f}].
\]

We define \( I' := I^*_{j_1,j_2} := (a^*_{j_1,j_2}, b^*_{j_1,j_2}) \). Notice that \( I' \) is contained in \( I \) with width \( \frac{w(I)}{8N_f} \leq w(I') \leq \frac{w(I)}{N_f} \) and that its endpoints are dyadic numbers (assuming that \( a \) and \( b \) are dyadic).

In the final step, we apply the 0-Test to the intervals \( I'_1 := (a, a^*_{j_1,j_2}) \) and \( I'_2 := (b^*_{j_1,j_2}, b) \). If both tests succeed, and hence, neither interval contains a root of \( P \), we return \( I' \). If one of the 0-Tests fails, we discard the pair \((j_1, j_2)\).

We say that the Newton-Test succeeds if it returns an interval \( I' = I^*_{j_1,j_2} \) for at least one of the three pairs \( j_1, j_2 \). If we obtain an interval for more than one pair, we can output either one of them. Otherwise, the test fails.

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

**Lemma 20.** Let \( I = (a, b) \) be an interval, \( N_f = 2^{2^j} \) with \( n_f \in \mathbb{Z}_{\geq 1} \), and \( J = (c, d) \subseteq I \) be a subinterval of width \( w(J) \leq 2^{-13} \cdot \frac{w(I)}{N_f} \). 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 \( 2^{\log n+10} \cdot N_f \cdot w(I) \) 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 \( \xi^*_j, j = 1, 2, 3 \), the inequality

\[
\left| m(J) - (\xi^*_j - k \cdot \frac{P(\xi^*_j)}{P'(\xi^*_j)}) \right| < \frac{w(I)}{128N_f},
\]

holds: There exist at least two points (say \( \xi := \xi^*_j \) and \( \bar{\xi} := \xi^*_j \) with \( j_1 < j_2 \)) whose distances to any root from \( z_1, \ldots, z_k \) are larger than \( \frac{|\xi - \bar{\xi}| - w(J)}{2} > \frac{3}{2} w(I) - \frac{1}{2} w(J) \geq 512N_f w(J) \). In addition, the distances to any of the remaining roots \( z_{k+1}, \ldots, z_n \) are larger than \( 2^{10}n_f w(I) - w(I) \geq 512 \cdot n_f w(I) \). Hence, with \( m := m(I) \), it follows that

\[
\left| \frac{1}{k} \sum_{i=1}^{k} \frac{\xi - m}{\xi - z_i} \right| = \frac{1}{k} \sum_{i=1}^{k} \frac{\xi - m}{\xi - z_i} + \frac{1}{k} \sum_{i=1}^{k} \frac{\xi - m}{\xi - z_i} - 1 = \frac{1}{k} \sum_{i=1}^{k} \frac{z_i - m}{\xi - z_i} + \frac{1}{k} \sum_{i=1}^{k} \frac{\xi - m}{\xi - z_i} \leq \frac{1}{k} \sum_{i=1}^{k} \frac{|z_i - m|}{|\xi - z_i|} + \frac{1}{k} \sum_{i=1}^{k} \frac{|\xi - m|}{|\xi - z_i|} < \frac{w(J)}{512n_f w(J)} + \frac{(n - k) \cdot w(I)}{512kn_f w(I)}
\]

where we used that \( \frac{P'(\xi^*_j)}{P(\xi^*_j)} = \sum_{i=1}^{n_f} (\xi - z_i)^{-1} \). This yields the existence of an \( \epsilon \in \mathbb{R} \) with \( |\epsilon| < \frac{1}{128N_f} \) and \( \frac{1}{k} \cdot \frac{1}{|\xi - m|} \cdot \frac{P'(\xi^*_j)}{P(\xi^*_j)} = 1 + \epsilon \). We can now derive the following bound on the

---

21 Notice that we can carry out all computations by approximate evaluation of the polynomials \( P \) and \( P' \) at the points \( \xi^*_j \) and \( \xi^*_j \) to an absolute precision of \( O(\log n + \log N_f + \log M \cdot |P(\xi^*_j)|^{-1} + \log M \cdot |P'(\xi^*_j)|^{-1}) \) bits after the binary point.

22 For intervals \( I'_1 \) or \( I'_2 \), which are empty (i.e., \( I'_1 = (a, a) \) or \( I'_2 = (b, b) \)), nothing needs to be done.
distance between the approximation \( \xi' = \xi - k \cdot \frac{P(\xi)}{P'(\xi)} \) obtained by the Newton iteration and \( m \):

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

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

Let \( v_{j_1} = \frac{P(\xi)}{P'(\xi)} \) and \( v_{j_2} = \frac{P(\xi)}{P'(\xi)} \) be defined as in the Newton-Test. Then, from the above considerations, it follows that \( |(\xi - k \cdot v_{j_1}) - (\xi - k \cdot v_{j_2})| < \frac{w(I)}{64N_I} \). 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 \( |\xi - k \cdot v_{j_1} - m| < \frac{w(I)}{128N_I} \) and \( m \in I \). An analogous argument yields that \( |k \cdot v_{j_2}| < w(I) \).

Hence, the conditions in (26) are fulfilled, whereas the conditions in (25) cannot hold. In the next step, we show that \( \lambda := \lambda_{j_1,j_2} \) as defined in (27) is actually a good approximation of \( \xi - k \cdot v_{j_1} \):

There exist \( \epsilon \) and \( \bar{\epsilon} \), both of magnitude less than \( \frac{w(I)}{128N_I} \), such that \( \xi - k \cdot v_{j_1} = m + \epsilon \) and \( \xi - k \cdot v_{j_2} = m + \bar{\epsilon} \). This yields

\[
\lambda = \xi + \frac{\xi - \xi}{v_{j_1} - v_{j_2}} \cdot v_{j_1} = \xi + \frac{(\epsilon - \bar{\epsilon} + k \cdot (v_{j_2} - v_{j_1}))}{v_{j_1} - v_{j_2}}.
\]

The absolute value of the fraction on the right side is smaller than \( \frac{w(I)}{64N_I} \cdot \frac{w(I)}{8k} \leq \frac{w(I)}{8N_I} \), and thus \( |\xi - k \cdot v_{j_1} - \lambda| < \frac{w(I)}{32N_I} \). It follows that

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

Hence, 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 \( \frac{w(I)}{16N_I} > \frac{w(J)}{32N_I} > \frac{w(I)}{16N_I} \). This shows that the interval \( I' = I_{j_1,j_2}' \) contains \( J \). Hence, the Newton-Test succeeds since the one-circle regions of \( I_{j_1}' \) and \( I_{j_2}' \) 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 \( C \subseteq \Delta(I) \) of roots whose center is close to one of the endpoints of \( I \). If, in addition, \( C \) 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 \( \frac{w(I)}{16N_I} \leq \frac{w(I')}{8N_I} \), which contains all real roots that are contained in \( I \), or it proves that \( I \) contains no root.

**Boundary-Test:** Let \( m_{\ell} := a + \frac{w(I)}{2N_I} \) and \( m_{r} := b - \frac{w(I)}{2N_I} \), and let \( \epsilon := 2^{-[2+\log n]} \). Compute admissible points

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

and run the 0-Test for the intervals \( I_{\ell} := (m_{\ell}, b) \) and \( I_{r} := (a, m_{r}) \). If both 0-Tests succeed, \( I \) contains no root, and thus, we return this result. If it succeeds only for \( I_{\ell} \), then \( I' := (a, m_{\ell}) \) contains all roots that are contained in \( I \), and we return \( I' \). If it succeeds only for \( I_{r} \), we return \( I' = (m_{r}, b) \). Notice that from our definition of \( m_{\ell}^* \) and \( m_{r}^* \), it follows that both intervals \( I_{\ell} \) and \( I_{r} \) have width in between \( \frac{w(I)}{4N_I} \) and \( \frac{w(I)}{8N_I} \). If neither 0-Test succeeds, the Boundary-Test fails.
We use $T$ to denote the subdivision forest which is induced by our algorithm ANewDsc. More precisely, in this forest, we have one tree for each interval $I_k$, with $k = 0, \ldots, 2 \log \Gamma + 1$, as defined in (23). Furthermore, an interval $I’$ is a child of some $I \in T$ if and only if it has been created by our algorithm when processing $I$. We have $\frac{w(I)}{\max N} \leq w(I’) \leq \frac{w(I)}{\min N}$ in a quadratic step and $\frac{1}{4} \cdot w(I) \leq w(I’) \leq \frac{1}{4} \cdot w(I)$ in a linear step. An interval in $T$ 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 $I \notin I$ with $\var(P,I) \leq 1$ 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 $T$, we estimate for each $I_k$ the size of the tree $T_k$ rooted at it. If $I_k$ is terminal, $T_k$ consists only of the root. So, assume that $I_k$ is non-terminal. Call a non-terminal $I \in T_k$ splitting if either $I$ is the root of $T_k$, 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 $T_k$ 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 $s_{\text{max}}$ 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 $T_k$ is bounded by $1 + s_{\text{max}} \cdot (2|\mathcal{M}_k| - 1)$, and the total number of non-terminal nodes in the subdivision forest is $O(\log \Gamma + n \cdot s_{\text{max}})$. Hence, the same bound also applies to the number of all nodes in $T$.

The remainder of this section is concerned with proving that $s_{\text{max}} = O(\log n + \log(\Gamma + \log M(\sigma_P^{-1})))$.

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 $I \in T$ and the corresponding numbers $N_I$ (Lemma 21).

2. We then study an abstract version of how interval sizes and interval levels develop in quadratic interval refinement (Lemma 22).

3. In a third step, we then derive the bound on $s_{\text{max}}$ (Lemma 23).

Lemma 21. For each interval $I \in T \setminus \mathcal{I}$, we have

$$2^\Gamma \geq w(I) \geq 2^{-4\Gamma - 6} \sigma_P^5 \quad \text{and} \quad 4 \leq N_I \leq 2^{4(\Gamma + 1)} \cdot \sigma_P^{-4}.$$
Claim 1: Suppose that the first \( n \) indices are strong. Then, \( x_{k+1} \leq 2^{-2^{m+i+1}} \), and \( n_{k+1} \geq 2 \). Thus, \( x_{k+1}/N_{k+1} < w'/w \), and \( k+1 \) is weak.

Let us next consider the subsequence \( S = k', k'+1, \ldots, i_0 - 6 \). This sequence starts with a weak index. We will split \( S \) 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 \( n_1 \). The details follow:

Claim 2: \( S \) contains no subsequence of type \( SS \) or \( SWSWS \).

Consider any weak index \( i \) followed by a strong index \( i+1 \). Then, \( N_i \geq N_j \) and \( x_{i+2} \leq x_i \), and hence, \( x_{i+2}/N_{i+2} \leq x_i/N_i < w' \). Thus, \( i+2 \) is weak. Since \( S \) 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 \( i+3 \) weak. We come to the evolution of interval sizes and levels in quadratic interval refinement. The following Lemma has been introduced in [27, Lemma 4] in a slightly weaker form:

**Lemma 22.** 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 \( s_1 = (x_1, x_1) := (w, m) \), and

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

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

**Proof.** The proof is similar to the proof given in [27]. 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 \( k \in \mathbb{N}_{\geq 1} \) be the unique integer with

\[
2^{-2^{k+1}} < w'/w < 2^{-2^k}.
\]

Then, \( k \leq \log \log \frac{w}{w'} \). 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 next show that the length of the prefix of strong indices is bounded by \( k \). 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. More precisely:

**Claim 1:** \( k' \leq k + 1 \).

Suppose that the first \( k \) indices are strong. Then, \( x_{k+1} \leq 2^{-2^{k+i}} \), for \( i = 1, \ldots, k \), and hence,

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

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

Let us next consider the subsequence \( S = k', k'+1, \ldots, i_0 - 6 \). This sequence starts with a weak index. We will split \( S \) 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 \( n_1 \). The details follow:
are weak. Then, \( N_i = N_{i+2} = N_{i+4}, N_{i+1} = N_i^2, \) \( x_{i+1} \leq x_i/N_i, \) \( x_{i+3} \leq x_{i+2}/N_{i+2}, \) \( x_{i+4} < x_{i+3}, \) \( x_{i+3} < x_{i+1}, \) and hence,

\[
\frac{x_{i+4}}{N_{i+4}} < \frac{x_{i+2}}{N_{i+2}N_{i+4}} \leq \frac{x_{i+1}}{N_i^2} = \frac{x_{i+1}}{N_{i+1}} < w'.
\]

Thus, \( i + 4 \) is weak.

**Claim 3:** If \( i \) is weak and \( i \leq i_0 - 6, \) then \( n_i \geq 2. \)

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

We now partition the sequence \( S \) into maximal subsequences \( S_1, S_2, \ldots, S_r, \) such that each \( S_j, j = 1, \ldots, r, \) contains no two consecutive weak elements. Then, according to our above results, each \( S_j, \) with \( j < r, \) is of type \( W, WS, WSW, \) or \( WSWS. \) The last subsequence \( S_r \) is of type \( WSWSW, \) or \( WSWSW. \) Since \( n_i \geq 2 \) for all weak \( i \) with \( i \leq i_0 - 6, \) the number \( n_i \) decreases by one after each \( S_j, \) with \( j < r. \) Thus, we must have \( r \leq n_1 + k' - 2 \) since \( n_{k'} = n_1 + k' - 1, n_{r-1} = n_{k'} - (r - 1), \) and \( n_r \geq 2. \) Since the length of each \( S_j \) is bounded by 5, it follows that

\[
i_0 = i_0 - 6 + 6 \leq k' + 5r + 6 \leq k' + 5(n_1 + k' - 2) + 6 \leq 5(n_1 + k) + k < 8(n_1 + k).
\]

\[\square\]

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

**Lemma 23.** The maximal length \( s_{\text{max}} \) 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 \( a_{s_1} \neq a_1 \) and \( b_{s_1} \neq b_1. \) 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 \( s_2 \geq s_1 \) be minimal such that either \( s_2 = s \) or \( w(I_{s_2}) \leq 2^{-13 - \log n} w(I_{s_1})/N_{s_2}. \) 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 \( s_2 < s, \) we have \( w(I_j) \leq 2^{-13 - \log n} w(I_{s_1})/N_{j} \) for all \( j \geq s_2. \) If \( I_{j+1} \) comes
   from \( I_j \) by a linear step, this is obvious because \( w(I_{j+1}) \leq w(I_j) \) and \( N_{j+1} \leq N_{j}. \) If it is
   generated in a quadratic step, we have \( w(I_{j+1}) \leq w(I_j)/N_{j}, \) and \( N_{j+1} = N_{j}^2. \)

In order to derive a bound on \( s_1, \) we appeal to Lemma 22. If \( w(I_j)/N_{j} \geq 4 \cdot w(I_{s_1+1}) \) 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, \( w(I_j)/N_{j} < 4 \cdot w(I_{s_1+1}). \) 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 \( I_{j_0+1} \) is quadratic, and hence, \( w(I_{j_0+1}) \leq w(I_{j_0})/N_{j_0} < 4 \cdot w(I_{s_1+1}) \),
and hence, a constant number of steps suffices to reduce the width of \( I_{j_0+1} \) to the width of \( I_{s_1+1}. \) For
\( j = 1, \ldots, j_0 - 1, \) the sequence \( (w(I_j), n_{j}) \) coincides with a sequence \((x_j, y_j)\) as defined in
**Lemma 22.** where \( w := w(I_j), w' := 4w(I_{s_1+1}), \) and \( n_1 := n := n_{j_0}. \) Namely, if \( w(I_j)/N_{j} \geq w', \)
we have \( w(I_{j+1}) \leq w(I_j)/N_{j} \) and \( n_{j+1} = 1 + n_{j}, \) and otherwise, we have \( w(I_{j+1}) \leq \frac{3}{4} \cdot w(I_j)\)

23
and $n_{I_j+1} = \max(1, n_{I_j} - 1)$. Hence, according to Lemma 22, 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 21.

We come to the bound on $s_2$. Observe first that $\min(|a_1 - a_s|, |b_1 - b_s|) \geq \frac{1}{2}w(I_s)$. 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, $N_{I_j} \leq N_{\text{max}} := 2O(\Gamma + \log M(\sigma_P^{-1}))$ for all $j$ according to Lemma 21. Thus, if the sequence $I_{s_1'}, I_{s_1'}, \ldots$ starts with $m_{\text{max}} := \max(5, \log \log N_{\text{max}} + 1)$ 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 $j' \leq s_1' + m_{\text{max}}$. Otherwise, there exists a $j'$ with $s_1' \leq j' \leq s_1' + m_{\text{max}}$, 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 $m_{\text{max}}$, there must exist a $j''$ with $j' + 1 \leq j'' \leq j' + m_{\text{max}} + 1$ 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
\[ w(I_{j''+1}) \leq \frac{3}{4}w(I_{j''}) \leq \frac{3w(I_{j'-1})}{4N_{I_{j'-1}}} \leq 2^{-13} \log n, \quad \frac{w(I_{s_1})}{N_{I_{j'-1}}}. \]
Hence, in any case, there exists an $s_2 \leq s_1 + 2m_{\text{max}} + 1$ 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
\[ |x - z_i| > 2^{\log n + 10} N_{I_j} w(I_j) \text{ for all } z_i \notin M(I_j) \text{ and all } x \in I_j. \]  

From (30) and Lemma 20, we conclude that the step from $I_j$ to $I_{j+1}$ is quadratic if $j \geq s_2$ and $w(I_j) \leq 2^{-13} \cdot \frac{w(I_j)}{N_{I_j}}$. 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 22. The sequence $(w(I_{s_2+1}), n_{I_{s_2+1}}) = (s_{3+s_2})$ coincides with a sequence $(x_i, n_i)_{i=1, \ldots, s_3+s_2}$ as defined in Lemma 22, where $n_1 = m = n_{I_{s_2+1}}$ and $w' := 2^{13} \cdot w(I_s)$. Namely, if $w(I_{s_2+1}) \cdot N_{I_{s_2+1}}^{-1} \geq w'$, then $w(I_{s_2+1}) \leq w(I_{s_2+1}) \cdot N_{I_{s_2+1}}^{-1}$ and $n_{I_{s_2+1}} = 1 + n_{I_{s_2+1}}$, whereas we have $w(I_{s_2+1}) \leq 2^w w(I_{s_2+1})$ and $n_{I_{s_2+1}} = \max(n_{I_{s_2+1}} - 1, 1)$ for $w(I_{s_2+1}) \cdot N_{I_{s_2+1}}^{-1} < w'$. It follows that $s_3 - s_2$ is bounded by $8(n_1 + \log \log \max(4, w(I_{s_2+1})/w')) = O(\log(\Gamma + \log M(\sigma_P^{-1})))$.

**Theorem 24.** Let $K = \log n + \log(\log \Gamma + M(\sigma_P^{-1}))$. The size $|\mathcal{T}|$ of the subdivision forest is
\[ O \left( \sum_{k=0}^{2^{\gamma+1}} (1 + |\mathcal{M}(I_k)| \cdot K) \right) = O(nK), \]
where $I_k$ are the intervals as defined in (23), and $\mathcal{M}(I_k)$ denotes the set of all roots contained in the one-circle region $\Delta(I_k)$ of $I_k$.

**4.2 Bit Complexity**

In order to bound the bit complexity of our algorithm, we associate a root of $P$ with every interval $I$ in the subdivision forest and argue that the cost (in number of bit operations) of processing $I$ is
\[ \tilde{O}(n(n + \tau_P + n \log M(z_i) + \log M(P'(z_i)^{-1}))). \]
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 upper bounded by summing the bound in (31) over all roots of $P$ and multiplying by $s_{\max} \log n + \log \Gamma$. Theorem 26 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 $|\mathcal{M}(\cdot)|$. 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 $I_k$, we associate the same root as with the parent interval. With terminal intervals $I_k$, 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 $I$.

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 $|\mathcal{M}(\cdot)|$ 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 $I_k$ with any root can be associated, and each root associated with an interval $I \not\subseteq I_k$ cannot be associated with any interval $I' \not\subseteq I_k$ 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 $\xi$ special with respect to $I$ (or just special if there is no ambiguity) if $\xi$ is

(P1) an endpoint of $I$, that is, $\xi = a$ or $\xi = b$.
(P2) an admissible point $m_{s} \in m(I)\{w(I) \cdot 2^{−[\log n + 2]}\}$ as computed in the 1-Test.
(P3) an admissible point $\xi_{s} \in \xi_{s}(I)\{w(I) \cdot 2^{−[5+\log n]}\}$ as computed in (24) in the Newton-Test.
(P4) an admissible point $a_{s} \in a_{s}(I)\{2^{−[5+\log n]}\}$ as computed in (28) in the Newton-Test.
(P5) an admissible point $m_{s} \in m(I)\{2^{−[5+\log n]}\}$ as computed in (29) in the Boundary-Test.

For intervals $I$ with $\var(P, I) = 0$, we have only special points of type (P1), and for intervals with $\var(P, I) = 1$, 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 25.** Let $I \in \mathcal{T}$ be an arbitrary interval, and let $\xi$ be a special point with respect to $I$. If $\Delta(I)$ contains a root of $P$, then

$$|P(\xi)| > 2^{−40n \log n − 2\Gamma} \cdot M(z_{i})^{−5n} \cdot \min(1, \sigma_{z_{i}})^{5} \cdot \min(1, |P'(z_{i})|)^{5}$$

(32)

for all $z_{i} \in \Delta(I)$. If $\Delta(I)$ contains no root, then $\xi$ fulfills the above inequality for all roots contained in $\Delta(I)$, where $I$ is the parent of $I$.

**Proof.** We will prove the claim via induction on the depth $k$ of an interval $I$, where the depth of the intervals $I_k$ is one. According to (22), the endpoints of $I_k$ fulfill inequality (32). Now, if $\xi$ is

---

23 If $I = I_k$ for some $k$ and $\Delta(I)$ contains no root, then $z_{i}$ can be chosen arbitrarily.
a special point (with respect to \( I_k \)) of type (P2), then

\[
|P(\xi)| \geq \frac{1}{4} \cdot \max \{|P(x)| : x \in m(I_k) | w(I_k) \cdot 2^{-[\log n+2]} \}.
\]

Since at least one of the points in \( m(I_k) | w(I_k) \cdot 2^{-[\log n+2]} \) has distance more than \( w(I_k) \cdot 2^{-[\log n+3]} > \frac{1}{8n} \) 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 \( w(I_k) \geq 1 \). An analogous argument yields \( |P(\xi)| > \frac{1}{8n \cdot 2^{4n+3}} \geq 2^{-8n \log n} \) if \( \xi \) is a special point of type (P3) to (P5), where we additionally use \( N_{I_k} = 4 \) 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 \( \xi \) 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 (32) holds for all admissible points \( \xi \) of type (P1).

The point \( \xi \) is a special point of type (P2): Since \( \varphi(P, J) \geq 2 \), the two-circle region of \( J \) contains at least two roots of \( P \), and thus the disk \( \Delta := \Delta_{2w(J)}(m(I)) \) centered at midpoint \( m(I) \) of \( I \) contains at least two roots. This shows that \( \sigma_i < 2w(J) \) for any root \( z_i \in \Delta \).

With \( \epsilon := w(I) \cdot 2^{-[\log n+2]} \) and \( K := \frac{w(J)}{w(I)} \cdot 2^{-[\log n+3]} \), we can now use Lemma 7 to show that

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

where \( z_i \) is an arbitrary root in \( \Delta \), and \( \mu(\Delta) \) denotes the number of roots contained in \( \Delta \). If the subdivision step from \( J \) to \( I \) is linear, then \( w(J)/w(I) \in [4/3, 4] \), and thus the bound in (32) is fulfilled. Otherwise, we have \( w(J)/w(I) \in [N_J, 4N_J] = [N_J^2, (2N_J)^2] \). In addition, \( w(J) \leq 4w(I_k)/N_J \), where \( k \) is the unique index with \( J \subseteq I_k \). We conclude that \( N_J \leq 4w(I_k)/w(J) \), and thus

\[
\frac{w(J)}{w(I)} \leq (2N_J)^2 \leq \left( \frac{8w(I_k)}{w(I)} \right)^2 \leq \frac{2^{12} \cdot M(z_i)^2}{w(J)^2}.
\]

since \( w(I_k) \leq 8M(x)^2 \) for all \( x \in I_k \). Furthermore, since

\[
|P'(z_i)| = n|P_n| \cdot \prod_{z_j \in \Delta, z_j \neq z_i} |z_i - z_j| \cdot \prod_{z_j \notin \Delta} |z_i - z_j| \leq n \cdot (2w(J))^\mu(\Delta)-1 \cdot \prod_{z_j \notin \Delta} |z_i - z_j|
\]

\[
\leq n \cdot 2^n \cdot w(J)^{\mu(\Delta)-1} \cdot \prod_{z_j \notin \Delta} M(z_i - z_j) \leq n \cdot 2^n \cdot w(J)^{\mu(\Delta)-1} \cdot \frac{\text{Mea}(P(z_i - x))}{|P_n|}
\]

\[
\leq n \cdot 2^n \cdot w(J)^{\mu(\Delta)-1} \cdot 2^{\tau_\phi 2^n M(z_i)^n} < 2^{4^n + \tau_\phi} \cdot M(z_i)^n \cdot w(J)^{\mu(\Delta)-1},
\]

it follows that

\[
w(J)^{-\mu(\Delta)-1} = w(J)^{-\mu(\Delta)-1} w(J)^{-2} < \frac{2^{8n + \tau_\phi} \cdot M(z_i)^n}{\sigma_i^4 \cdot |P'(z_i)|^4},
\]

where we used that \( 2w(J) > \sigma_i \) in order to bound \( w(J)^{-2} \). Hence, it follows from (34) that

\[
\left( \frac{w(J)}{w(I)} \right)^{-\mu(\Delta)-1} \geq (2^{12} \cdot M(z_i)^2 \cdot w(J)^{-2})^{-\mu(\Delta)-1}
\]

\[
\geq 2^{-12(n+1)} \cdot M(z_i)^{-2(n+1)} \cdot 2^{-16n-2\tau_\phi} \cdot M(z_i)^{-2n \sigma_i^4} \cdot |P'(z_i)|^4
\]

\[
> 2^{-32n-2\tau_\phi} \cdot M(z_i)^{-5n} \cdot \sigma_i^4 \cdot |P'(z_i)|^4.
\]
Plugging the latter inequality into (33) eventually yields

\[ |P(\xi)| > 2^{-8n \log n} \cdot \left( \frac{w(J)}{w(I)} \right)^{-\mu(\Delta)-1} \cdot \sigma_i \cdot |P'(z_i)| > 2^{-40n \log n - 2r_p} \cdot M(z_i)^{-5n} \cdot \sigma_i^5 \cdot |P'(z_i)|^5. \]

Thus, \( \xi \) fulfills the bound (32).

**The point \( \xi \) 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 \( \xi_j \) contains at least two roots, and thus we can use Lemma 7 with \( \epsilon := w(J) \cdot 2^{-[\log n + 5]} \) and \( K := \frac{w(J)}{w(I)} \cdot 2^{[\log n + 6]} \).

**The point \( \xi \) 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 \( \text{var}(P, I) \geq 2 \), and thus, each disk \( \Delta := \Delta_{2w(I)}(x_0) \) with radius \( 2w(I) \) centered at an arbitrary point \( x_0 \in I \) contains at least two roots. We use this fact \( x_0 = a_{i_1, j_2} \) and \( x_0 = b_{j_1, j_2} \) and obtain, using Lemma 7 with \( \epsilon := \frac{w(I)}{N_I} \cdot 2^{-[\log n + 5]} \) and \( K := N_I \cdot 2^{[\log n + 4]} \),

\[ |P(\xi)| > 2^{-4n - 1} \cdot N_I^{-\mu(\Delta)-1} \cdot \sigma_i \cdot |P'(z_i)| > 2^{-8n \log n} \cdot N_I^{-\mu(\Delta)-1} \cdot \sigma_i \cdot |P'(z_i)|. \]  

(35)

If the subdivision step from \( J \) to \( I \) is linear, then \( N_J \leq N_I \leq 4w(I_k)/w(J) \), where \( k \) is the unique index with \( J \subseteq I_k \). If the step from \( J \) to \( I \) is quadratic, then \( N_J = N_I^2 \leq (4w(I_k)/w(J))^2 \). Now, the same argument as in the type (P2) case (see (34) and the succeeding computation) shows that

\[ N_I^{-\mu(\Delta)-1} \geq \left( \frac{4w(I_k)}{w(J)} \right)^{-\mu(\Delta)-1} \geq \left( 2^{10} \cdot M(z_i)^2 \cdot w(J)^{-2} \right)^{-\mu(\Delta)-1} \]

\[ \geq 2^{-32n - 2r_p} \cdot M(z_i)^{-5n} \cdot \sigma_i^4 \cdot |P'(z_i)|^4, \]

and thus

\[ |P(\xi)| > 2^{-8n \log n} \cdot N_I^{-\mu(\Delta)-1} \cdot \sigma_i \cdot |P'(z_i)| > 2^{-40n \log n - 2r_p} \cdot M(z_i)^{-5n} \cdot \sigma_i^5 \cdot |P'(z_i)|^5. \]

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

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

**Theorem 26.** Let \( P = P_0 x^n + \ldots + P_n \in \mathbb{R}[x] \) 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

\[
\tilde{O}(n \cdot (n^2 + n \log \text{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1})))
\]

(36)

\[ = \tilde{O}(n(n^2 + n \log \text{Mea}(P) + \log M(\text{Disc}(P)^{-1}))). \]

(37)

The coefficients of \( P \) have to be approximated to

\[
\tilde{O}(n + \tau_p + \max_i (n \log M(z_i) + \log M(P'(z_i)^{-1})))
\]

bits after the binary point.
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 $\xi$ with respect to $I$. Since each of these points fulfills the inequality (32), we conclude from Lemma 5 that the computation of all special points $\xi$ uses

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

bit operations, where $z_i$ is an arbitrary root contained in $\Delta(I)$. We remark that when applying Lemma 5, we used (22) 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 15 and Corollary 18 yield the same complexity bound as stated in (38) 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 (38) applies to all 0-Tests and 1-Tests.

It remains to bound the cost for the computation of the values $\lambda_{j_1,j_2}$ in the Newton-Test: We have already remarked (Footnote 21) that, for the latter computation, it suffices to evaluate $P$ and $P'$ at the points $\xi_{j_1}^*$ and $\xi_{j_2}^*$ to an absolute precision of

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

Thus, according to Lemma 2 and Lemma 25, the total cost for this step is bounded by

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

bit operations, where $z_i$ is an arbitrary root contained in $\Delta(I)$. Here, we used the fact that $2w(I) > \sigma_i$ (notice that $\text{var}(P,I) \geq 2$, and thus, the two-circle region of $I$ contains at least two roots) and that $\log N_I = O(\log M(\sigma_i^{-1}) + \log M(z_i))$ as shown in the proof of Lemma 25. In summary, for any interval $I$ whose one-circle region contains at least one root, the cost for processing $I$ is bounded by (38). 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 (38), where $z_i$ is an arbitrary root in $\Delta(J)$ and $J \in \mathcal{T}$ is the parent of $I$.

Since we can choose an arbitrary root $z_i \in M(I)$ (or $z_i \in M(J)$ for the parent $J$ of $I$ if $M(I)$ 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 $O(\max_{j_k} \log n + \log \Gamma)$ many roots associated with it, the total cost for processing all intervals is bounded by

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

bit operations, where we used that $\sum_{i=1}^{n} \log M(z_i) = \log \frac{\text{Mea}(P)}{P_{n+1}}$, $\tau_P \leq n + \log \text{Mea}(P)$, and that the factor $s_{\max} \log n + \log \Gamma$ is swallowed by $\tilde{O}$. We can further discard the sum $\sum_{i=1}^{n} \log M(\sigma_i^{-1})$ in the above complexity bound. Namely, if $z_k$ denotes the root with minimal distance to $z_i$, then (we use inequality (19))

$$\sigma_i \geq \frac{|P'(z_i)|}{|P_{n+1}| \cdot \prod_{j \neq k,i} |z_j - z_i|} \geq \frac{|P'(z_i)|}{\text{Mea}(P(z - z_i))} \geq \frac{|P'(z_i)|}{2^{n^2} \cdot 2^n \cdot M(z_i)^n},$$

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

For the alternative bound, we use inequalities (18) and (20).

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 27. For a polynomial \( p(x) = p_n \cdot x^n + \cdots + p_0 \in \mathbb{Z}[x] \) with integer coefficients of absolute value \( 2^r \) or less, the algorithm ANewDsc computes isolating intervals for all real roots of \( p \) with \( \tilde{O}(n^3 + n^2\tau) \) bit operations.

Proof. We first consider 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 26, where we use that \( \tau_P \leq \tau \), \( \text{Mea}(P) \leq \text{Mea}(p) \leq (n + 1)2^t \) (inequality (14)), \( \text{Disc}(P) = 2^{-(2n-2)t} \cdot \text{Disc}(p) \), and the fact that the discriminant of an integer polynomial is integral. \( \square \)

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 arbitrary good approximations of the coefficients of \( P \), our algorithm ANewDsc returns isolating intervals \( I_1 \) to \( I_m \) with the property that \( \text{var}(P, I_k) = 1 \) 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.\(^{24}\)

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 \( \kappa \) is an arbitrary given 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 \( \kappa \), the latter bound is optimal (up to logarithmic factors) since the amortized cost per root and bit of precision is logarithmic in \( n \) and \( \kappa \).

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 \( \text{var}(P, I_k) = 1 \). According to Theorem 9, the Obreshkoff lens \( L_n \) of each interval \( I_k \) is also isolating for the root \( z_k \). Hence, from the proof of [27, Lemma 5] (see also [27, Figure 3.1]), we conclude that

\[
|x - z_j| > \frac{\min(|x - a_k|, |x - b_k|)}{4n} \quad \text{for all } x \in I_k \text{ and all } j \neq k. \tag{39}
\]

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 \( f \) at the endpoints of an interval. In ANewDsc, we used:

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

\[
m[\epsilon] := \{m[i] := m + (i - \lfloor n/2 \rfloor) \cdot \epsilon \mid i = 0, \ldots, 2 \cdot \lfloor n/2 \rfloor \}
\]

is a multipoint of size \( 2 \cdot \lfloor n/2 \rfloor + 1 \).

\(^{24}\)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 \( \alpha \) of the resultant polynomial \( R(x) := \text{res}(f, f_y; y) \in \mathbb{Z}[x] \) 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 \( \alpha \) in order to obtain good approximations of the coefficients of \( f(\alpha, y) \).
(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 \(I\).\(^{25}\)

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

\[
m[\epsilon]' = \{m_1', m_2'\} := \{m - [n/2] \epsilon, m + [n/2] \epsilon\} \subset m[\epsilon]
\]

consists of the first and the last point from \(m[\epsilon]\) only.\(^{26}\)

(B') execution of a sign-test on an interval \(I' = (a',b') \subset (a,b)\) (i.e., the computation of \(\text{sgn}(f(a') \cdot f(b'))\)), where \(a'\) and \(b'\) are admissible points in some \(m[\epsilon]'\).\(^{27}\)

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

**\textsc{Refine}:** We maintain a list \(\mathcal{A} := \{(I, N_I)\}\) of pairs, each consisting of an active interval and a corresponding positive integer \(N_I = 2^{2^k}\) with \(n_I \in \mathbb{N}_{\geq 1}\). \(\mathcal{O}\) denotes a list of isolating intervals of size less than \(2^{-\kappa}\). Initially, set \(\mathcal{A} := \{(I_k, 4)\}_{k=1, \ldots, m}\) and \(\mathcal{O} := \emptyset\). In each iteration, we remove a pair \((I, N_I)\) from \(\mathcal{A}\), with \(I = (a,b)\), and proceed as follows:

(Q') We apply the Boundary-Test as well as the Newton-Test to \(I\), where the steps in (A) and (B) are replaced by the respective modifications (A') and (B'). If one of these tests succeeds, we obtain an interval \(I' \subseteq I\), with \(\frac{w(I)}{N_I} \leq w(I') \leq \frac{w(I)}{N_I}\), which isolates the root that is isolated by \(I\). If \(w(I') < 2^{-\kappa}\), we add \(I\) to \(\mathcal{O}\). Otherwise, we add \((I', N_{I'})\) to \(\mathcal{A}\), where \(N_{I'} := N_I^2\).

(L') If (Q') fails, we compute an admissible point \(m^* \in m[\epsilon][\frac{w(I)}{2^{\frac{1}{2^{n-1}}}}]'\). Let \(I' := (a, m^*)\), \(I'' := (m^*, b)\), and \(N_{I'} := N_{I''} := \max(4, \sqrt{N_I})\).

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

(linear step)

We continue until the list \(\mathcal{A}\) becomes empty. Then, we return the list \(\mathcal{O}\).

\(^{25}\)Notice that this step also uses the computation of admissible points.

\(^{26}\)In fact, one can show that choosing two arbitrary points from \(m[\epsilon]\) does not affect any of the following results.

\(^{27}\)More precisely, we compute \(s := \text{sgn}(P(a') \cdot P(b'))\). If \(s > 0\), then \(I'\) contains no root. If \(s < 0\), then \(I'\) isolates the root \(z_k\). Since \(\text{var}(P, I_k) = 1\), it follows that \(s > 0\) if and only if \(\text{var}(P, I') = 0\), and \(s < 0\) if and only if \(\text{var}(P, I') = 1\).
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 28.

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

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

iterations, where $s_{\text{max}, k}$ 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 $O(n(\log n + \log(\Gamma + \kappa)))$.

**Proof.** Similar as in the proof of Lemma 21, 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 (22), 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 20 and of our considerations in the proof of Lemma 23. Namely, exactly the same argument as in the proof of Lemma 23 shows that the maximal length of any path between splitting nodes, denoted $s'_{\text{max}, k}$, is $O(\log n + \log(\log M(z_k) + \kappa))$, and thus the path from $I_k$ to the refined interval $J_k \subset I_k$ of size less than $2^{-\kappa}$ has length $s'_{\text{max}, k} \cdot |\mathcal{M}(I_k)|$. □

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

---

28However, we will later show how to make good use of approximate multipoint evaluation in order to improve the worst case bit complexity.

29For 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 $s'_{\text{max}, k}$.
Lemma 29. For an active interval $I \subset I_k$ of size $w(I) \geq \sigma_k/2$, the cost for processing $I$ is bounded by
\[ O\left(n(n + \tau_p + n \log M(z_1) + \log M(P'(z_1)^{-1}))\right), \]
where $z_1$ is an arbitrary root contained in the one-circle region of $I$. If $w(I) < \sigma_k/2$, the cost for processing $I$ is bounded by
\[ O\left(n(\kappa + n + \tau_p + n \log M(z_1) + \log M(P'(z_1)^{-1}))\right). \]

Proof. Suppose that $w(I) \geq \sigma_k/2$, 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 $[n/2] \cdot \epsilon \geq n \cdot \epsilon/2$. Hence, from inequality (39) 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 7 (with $x_{i_0} := m_1'$) shows that
\[ |P(\xi)| > 2^{-6n-1} \cdot K^{-\mu_1}\cdot \sigma_1 \cdot |P'(z_i)| \]
for all $z_i \in I$, where $K \geq 2 \cdot \lfloor n/2 \rfloor$ is an arbitrary positive real value, such that the disk $\Delta := \Delta_{\mu_1}(m)$ contains at least two roots of $P$. Since $w(I) \geq \sigma_k/2$, it further follows that the disk $\Delta_{w(I)}(m(I))$ contains at least two roots. Thus, we can use the same argument as in the proof of Lemma 25 (type (P2)-(P5) cases) to prove that the inequality (32) holds for $\xi$. In addition, inequality (32) 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
\[ O(n \log n + \tau_p + n \log M(z_1) + \log M(\sigma_1^{-1}) + \log M(P'(z_1)^{-1})) \]
\[ O(n \log n + \tau_p + n \log M(z_1) + \log M(P'(z_1)^{-1})), \]
for $n \geq 2$. Let $\xi \in m[\epsilon]'$ be a point that is computed 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
\[ |P(\xi)| \geq \frac{|P(m_1')|}{4} \geq \frac{1}{4} \cdot \frac{w(I)}{8N_I} \cdot 2^{-2(n-1)} \cdot |P'(z_k)| \geq \frac{w(I)^3}{2^9M(z_k)^3} \cdot 2^{-2(n-1)} \cdot |P'(z_k)| \geq 2^{-4n - 3\kappa - 2n - 7} \cdot |P'(z_k)|, \]
where we used the bounds for $w(I)$ and $N_I$ as computed in the proof of Lemma 28. Furthermore, the endpoints of $I$ fulfill the inequality (32), and thus all approximate polynomial evaluations (when processing $I$) are carried out with an absolute precision of
\[ O(\log M(w(I)^{-1}) + n \log n + \tau_p + n \cdot \log M(z_k) + \log M(\sigma_k^{-1}) + \log M(P'(z_k)^{-1})) = \]
\[ O(\kappa + n \log n + \tau_p + n \cdot \log M(z_k) + \log M(P'(z_k)^{-1})). \]
This proves the second claim. \( \square \)
Combining Lemma 28 and Lemma 29 now yields the following result:

\textbf{Theorem 30.} The cost for refining $I_k$ to an interval of size less than $2^{-\kappa}$ is bounded by

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

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

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

\textbf{Proof.} We split the total cost into those for refining the interval $I_k$ to a size less than $\sigma_k/2$ and into those for the additional refinement steps until the interval has size less than $2^{-\kappa}$. For the latter cost, we remark that $|M(I)| = 1$ if $I$ is an isolating interval for $z_k$ of width $w(I) < \sigma_k/2$. Hence, there are at most $s_{\text{max},k}$ refinement steps of $I$, each of cost $O(n(k + n + \tau_p + n \log M(z_k) + \log M(P'(z_k)^{-1})))$. It remains to bound the cost for refining $I_k$ to a width of less than $\sigma_k/2$. According to Lemma 29, the cost for processing an interval $I$ of width $w(I) \geq \sigma_k/2$ is bounded by $O(n(n + \tau_p + n \log M(z_i) + \log M(P'(z_i)^{-1})))$, where we can choose an arbitrary root $z_i \in M(I)$. If we choose the root $z_i$ that is associated with $I$, then each root in $M(I_k)$ is considered at most $s_{\text{max},k}$ many times. Thus, the first complexity bound follows. The bound \((43)\) 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.

\[\square\]

\section{5.3 Asymptotic Improvements}

In this section, we show that our complexity bound \((43)\) for refining all intervals can be further improved, that is, the term $n^2k$ can be replaced by $nk$. We achieve this result by using fast approximate multipoint evaluation. For an integer $l$, with $0 \leq l \leq \log k + 1$, 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 $l$-\textit{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 $m_k(l)''$ 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

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

where $z_i$ is an arbitrary root in the one-circle region of the interval $I_k$, and $I_k'$ is the interval for which the highest precision is needed; see \((41)\) and \((42)\), and use that $w(I'_k) \geq 2^{-2^l}$. We now distinguish the following three cases:

\[\text{Essentially, we use the the same definition as in Section 4.2. More precisely, we say that a root $z_i$ is associated with $I$ if $z_i \in M(I)$ and the number of children $I' \subset I$ with $z_i \in M(I')$ is minimal for all roots in $M(I)$. Notice that each root $z_i \in \Delta(I_k)$ is associated with at most $s_{\text{max},k}$ intervals; see Lemma 28.} \]
(1) \( l = 0 \): The cost in (44) 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 \( I_k^{n_0} \).

(2) \( l > 0 \) and there exists an interval \( I_k' \) with \( \mathcal{M}(I_k') > 1 \): Since \( 2^{-2^{l-1} - 1} > w(I_k') \geq 2^{-2^l} \), each root \( z_i \) in \( \mathcal{M}(I_k) \) has separation \( \sigma_i < 2^{-2^l-1} \), and thus, replacing the term \( 2^l \) in (44) by \( \log M(\sigma_i^{-1}) \) yields

\[ \tilde{O}(n(\log M(\sigma_i^{-1}) + n + \tau_p + n \cdot \log M(z_j) + M(P'(z_j)^{-1}))) \]

where \( z_i \) is an arbitrary root in \( \mathcal{M}(I_k') \) and \( z_j \) is an arbitrary root in \( \mathcal{M}(I_k^{n_0}) \). We allocate the cost to roots \( z_i \) that are associated to the intervals \( I_k' \) and \( I_k^{n_0} \), 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 \( I_k^{n_0} \).

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

\[ \tilde{O}(n(n^2 + n \log \text{Mea}(P) + \sum_{i=1}^{n} \log M(\sigma_i^{-1}) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1}))) = \]

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

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 \( \max_{k=1,\ldots,m} s_{\text{max},k} \) 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 \( s_{\text{max},k} \). Hence, the sum over the first term \( n \cdot 2^l \) in (44) over all \( l \) is bounded by \( \max_{k=1,\ldots,m} s_{\text{max},k} \cdot n \cdot \kappa \). The sum over the remaining term is again bounded by

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

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

**Theorem 31.** 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 \( \kappa \) be an arbitrary positive integer. Computing isolating intervals of size less than \( 2^{-\kappa} \) for all real roots needs a number of bit operations bounded by

\[ \tilde{O}(n \cdot (\kappa + n^2 + n \log \text{Mea}(P) + \sum_{i=1}^{n} \log M(P'(z_i)^{-1}))) \]  

\[ = \tilde{O}(n \cdot (\kappa + n^2 + n \log \text{Mea}(P) + \log M(\text{Disc}(P)^{-1}))) \]  

The coefficients of \( P \) have to be approximated to

\[ \tilde{O}(\kappa + n + \tau_p + \max_{i} n \log M(z_i) + \log M(P'(z_i)^{-1}))) \]

bits after the binary point. For a polynomial \( P \) with integer coefficients of size less than \( 2^7 \), 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 arbitrary 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 1982 splitting circle method. By comparison, our approach is completely different from Pan’s method and, in addition, it is much 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.

We are currently working on an implementation of ANewDsc. More precisely, we are considering a randomized version of our algorithm, where we choose admissible (subdivision) points at random and not via approximate multipoint evaluation as proposed in this paper (see Section 2.2). It seems likely that such a randomized version will yield a comparable bit complexity bound in expectation while, at the same time, showing excellent behavior in practice.

References


