Root Refinement for Real Polynomials using Quadratic Interval Refinement

Michael Kerber

MPI for Informatics, Campus E 1.4, 66123 Saarbrücken, Germany

Corresponding author: mkerber@mpi-inf.mpg.de *(phone: +49683193251005)*

Michael Sagraloff

MPI for Informatics, Campus E 1.4, 66123 Saarbrücken, Germany

Abstract

We consider the problem of approximating all real roots of a square-free polynomial f with real coefficients. Given isolating intervals for the real roots and an arbitrary positive integer L, the task is to approximate each root to L bits after the binary point. Abbott has proposed the quadratic interval refinement method (QIR for short), which is a variant of Regula Falsi combining the secant method and bisection. We formulate a variant of QIR, denoted AQIR ("Approximate QIR"), that considers only approximations of the polynomial coefficients and chooses a suitable working precision adaptively. It returns a certified result for any given real polynomial, whose roots are all simple. In addition, we propose several techniques to improve the asymptotic complexity of QIR: We prove a bound on the bit complexity of our algorithm in terms of the degree of the polynomial, the size and the separation of the roots, that is, parameters exclusively related to the geometric location of the roots. For integer coefficients, our variant improves, in theory and practice, the variant with exact integer arithmetic. Combining our approach with multipoint evaluation, we obtain near-optimal bounds that essentially match the best known theoretical bounds on root approximation as obtained by very sophisticated algorithms.

Keywords: Root isolation, Root approximation, Quadratic Interval Refinement, Complexity Analysis

1. Introduction

The problem of computing the real roots of a polynomial in one variable is one of the best studied problems in mathematics. If one asks for a *certified* method that finds all roots, it is common to write the solutions as a set of disjoint *isolating* intervals, each containing exactly one root; for that reason, the term *real root isolation* is common in the literature. For integer polynomials, simple, though efficient, methods for this problem have been presented, for instance, based on Descartes' Rule of Signs [10], or on Sturm's theorem [9, 13]. The majority of these methods exclusively performs exact arithmetic over rational numbers, and thus returns a certified result. Recently, the focus of research shifted to polynomials with real coefficients, which are approximated during the algorithm. It is worth remarking that this approach does not just generalize the integer case but also leads to practical [16, 38] and theoretical [39] improvements of it.

We consider the related *real root refinement problem*: assuming that isolating intervals of a polynomial are known, *refine* them to a width of 2^{-L} or less, where $L \in \mathbb{N}$ is an additional input parameter. The combination of root isolation and root refinement, which we call *strong root isolation*, yields a certified approximation of all roots of the polynomial to an absolute precision of 2^{-L} or, in other words, to L bits after the binary point in binary representation. Abbott's *quadratic interval refinement* method [1] (QIR for short) is a hybrid of the bisection and the secant method, which eventually converges quadratically (see Section 3). A straightforward approach for refinement is to apply QIR to each isolating interval until the width becomes small enough. This approach combines the advantages of being certified (the refined interval is guaranteed to contain a root), simple (in terms of implementation) and adaptive (the refinement process might switch to quadratic convergence much earlier than predicted by theory) and has therefore been used in several implementations where accurate and certified approximations of real roots are required. Another advantage is that the polynomial is only queried through polynomial evaluations; for instance, derivatives as in Newton iteration are not needed. The QIR approach has been analyzed in [19] for integer polynomials: for a polynomial of degree d and τ-bit coefficients and m real roots, it requires $\tilde{O}(d^4\tau^2 + md^2L)$ bit operations to refine all roots to L bits.

1.1. Our Contributions

Our work generalizes [19] from integer to arbitrary real coefficients, without assuming that exact operations on real numbers are available at unit cost. Instead, our approach works only with approximations of the input and exclusively

performs approximate but certified arithmetic, that is, in each step of our algorithm, errors which result from the use of approximate arithmetic are bounded and taken into account using interval arithmetic. We assume the existence of an oracle which, for an arbitrary positive integer ρ, provides approximations of the coefficients of the input polynomial to an error of less than $2^{-\rho}$. We also quantify the size of ρ in the worst case. We obtain an algorithm, called AQIR ("Approximate QIR") that shares the traits of being certified, simple, and adaptive with its exact counterpart, given that arbitrary approximations of the coefficients are accessible. We analyze the bit complexity of AQIR and give a bound that depends on the degree of the polynomial, the size of its largest root and the separation of its roots; see Theorem 22 and Section 2 for the precise statement. We are not aware of a similar in-depth analysis of any related approach for real coefficients. Our analysis proceeds in a similar way as [19], splitting the sequence of QIR steps in the refinement process into a *linear sequence*, where the method behaves like bisection in the worst case, and into a *quadratic sequence*, where the interval is converging quadratically towards the root; for technical reasons, we introduce an initial *normalization phase* that modifies the intervals to guarantee the efficiency of our refinement strategy.

Remarkably, AQIR does not only generalize, but also improves the integer case: We obtain a bit complexity of $\tilde{O}(d^3\tau^2 + mdL)$ if all coefficients are τ-bit integers. Compared to [19], we get rid of one factor of d by a different approach for evaluating the sign of f at rational points, which is the main operation in the refinement procedure. For an interval of size $2^{-\ell}$, the evaluation of f at the endpoints of the interval has a complexity of $\tilde{O}(d^2(\tau+\ell))$ with exact rational arithmetic because the function values can consist of up to $d(\tau+\ell)$ bits. However, we show that we can still compute the sign of the function value with certified numerical methods using the substantially smaller working precision of $O(d\tau+\ell)$ if the distance to the closest root is not much smaller than $2^{-\ell}$. Thus, the crucial modification of AQIR is to ensure that the boundaries of an isolating interval are sufficiently far away from the root that it contains; the details are described in Sections 4 to 6. Moreover, we show by experimental comparisons (Section 7) that our asymptotic improvement is also reflected in practice: the ratio of running times of exact and approximate QIR is proportional to the degree. Although AQIR is tailored to an accessible complexity analysis and does not yield an efficient practical implementation without further modifications, our experiments show that using approximate arithmetic is crucial for any efficient implementation of QIR.

We can further reduce the complexity of our method using the technique of *fast approximate multipoint evaluation* [37, 22, 24]. With that approach, performing

one refinement step on all isolating intervals simultaneously has the same cost (up to logarithmic factors) as a single refinement step with classical evaluation. This yields a bit complexity of $\tilde{O}(d^3\tau^2 + dL)$ for integer polynomials; the bound for polynomials with arbitrary real coefficients also scales like $\tilde{O}(dL)$ for large L. Notice that this bound is optimal up to logarithmic factors if L is the dominating factor and $m = \Theta(d)$ because the output complexity is $\Theta(mL)$. We consider fast approximate multi-point evaluation as a purely theoretical tool to improve the bit (and also the arithmetic) complexity. Although not hard to implement, an *efficient* realization is not easy to achieve and its advantages are unlikely to show up for instances that are currently feasible. For this reason, we refrained from implementing the corresponding variant of AQIR.

1.2. Related Work

The problem of accurate root approximation is omnipresent in mathematical applications; certified methods are of particular importance in the context of computations with algebraic objects, for instance, when computing the topology of algebraic curves [8, 15] or when solving systems of multivariate equations [2, 25].

The theoretical complexity of approximating all complex roots has been investigated by Pan [33, 31, 34]. His approach combines the splitting circle method [40] with techniques from numerical analysis (i.e. Newton iteration, Graeffes method, discrete Fourier transforms) and fast algorithms for polynomial and integer multiplication. For polynomials with integer coefficients, this yields an algorithm with a bit complexity of $\tilde{O}(d^2\tau + dL)$ for approximating all roots to an accuracy of 2^{-L}, and, for square-free polynomials with arbitrary real coefficients, it still scales like $\tilde{O}(dL)$ for large L; see [29, 18] for details. Our bound for AQIR matches this complexity if L is the dominant factor, but it is still inferior by a factor of $d\tau$ in the first term. Still, we think that this does not turn our analysis obsolete because we show that almost optimal asymptotic bounds can also be achieved with an easily implementable and practically efficient method. In contrast, as Pan admits in [32], the splitting-circle method is difficult to implement and so is the complexity analysis when taking rounding errors in intermediate steps into account. We are not aware of any implementation of Pan's method. Our approach also has the practical advantage of only considering the real roots of the polynomial. Sometimes, this number is small, for instance for various types of random polynomials [17] and for sparse polynomials [11]. In such situations, dedicated real root isolation algorithms, such as the Descartes or the continued fraction method, can be considerably more efficient than state-of-the-art numerical root solvers [12, 23, 38] and can be combined with our QIR approach. Also, if the number of real roots

is small compared to the degree of the polynomial, the variant of AQIR without fast multipoint evaluation achieves a comparable theoretical runtime as the variant with multipoint evaluation.

Using fast multipoint evaluation to improve the (arithmetic) complexity of root refinement has already been discussed in [34]. The author claims that despite a saving of a factor of $d/\log^2 d$ in arithmetic operations, such an approach would suffer from numerical instability and therefore yield an inferior Boolean complexity. Our results do not confirm this claim, at least with respect to asymptotic complexity bounds.

The idea of combining bisection with a faster converging method, such as Newton iteration or the secant method, in order to compute approximations of roots of continuous functions has been first introduced in *Dekker's method* and elaborated in *Brent's method*; see [7] for a summary. While theoretical convergence rates have been established, a certified algorithm working with coefficient approximations has to take into account the effect of rounding errors, facing the same challenges as our AQIR approach. We are not aware of any variant of Brent's method, or of any other real root refinement method, that has been analyzed with the same rigorousness as we do in this paper for the QIR method. We are confident, however, that also other hybrid methods can be modified in a way to yield comparable complexity bounds. For polynomials with integer coefficients, a recently proposed method [35, 36] combines Newton iteration and bisection, achieving similar complexity bounds. In comparison to our approach, their method requires the initial isolating interval $I = (a,b)$ to be small enough such that, except for the isolated root, there is no other (real or non-real) root of f with a distance less than $(1+\varepsilon)\cdot\frac{b-a}{2}$ to the center $\mathrm{mid}(I)$ of I, where $(1+\varepsilon)^n > 2$ for some $n = O(\log d)$. For large L, the method also scales like $\tilde{O}(dL)$ when refining the real roots of f, also using fast multi-point evaluation.

For approximating the (complex) roots of a polynomial, further numerical algorithms are available, for instance, the *Jenkins-Traub algorithm* or *Durand-Kerner iteration*; although they usually approximate the roots very fast in practice [5], general worst-case bounds on their arithmetic complexity are not available. For some variants, even termination cannot be guaranteed in theory; we refer to the surveys [32, 26, 27] for extensive references on these and further methods.

We improve upon the conference version of this paper [20] in several ways: in our bit complexity result, we remove the dependence on the coefficient size and, thus, relate the hardness of root approximation to parameters that exclusively depend on the geometric location of the roots. In addition, we redefine the threshold for the interval width that guarantees quadratic convergence (Definition 12);

in this way, we get rid of the magnitude $R = \log|\mathrm{res}(f,f')|^{-1}$, which is a pure artifact of the analysis of [20]. Moreover, the improvements on the complexity result using multipoint evaluations as well as the experimental evaluations did not appear in [20].

Outline. We introduce notations and state the complexity bound for general polynomials in Section 2. We summarize the (exact) QIR method in Section 3. Our AQIR algorithm that only uses approximate coefficients is described in Section 4. Its precision demand is analyzed in Section 5. Based on that analysis of a single refinement step, the complexity bound of root refinement is derived in Section 6. Some experimental comparison between QIR with exact and approximate coefficients is presented in Section 7. Further asymptotic improvements using multipoint evaluation are described in Section 8. We conclude in Section 9.

2. Notations and Main Result

The following quantities remain fixed throughout the paper. Let

$$f(x) := \sum_{i=0}^{d} a_i x^i \in \mathbb{R}[x] \tag{1}$$

be a square-free polynomial of degree d, with $d \geq 2$, and $|a_d| \geq 1$, and $\tau := \lceil \log(\max_i |a_i|) \rceil \geq 1$; throughout the paper, log means the logarithm with base 2. We denote the (complex) roots of f by $z_1, \ldots, z_d$, and, w.l.o.g., we can assume that the roots are numbered such that the first m roots $z_1, \ldots, z_m$ are exactly the real roots of f. For each z_i with $1 \leq i \leq d$, $\sigma_i = \sigma(z_i, f) := \min_{j \neq i} |z_i - z_j|$ denotes the *separation of* z_i, $\Sigma_f := \sum_{i=1}^{d} \log \sigma_i^{-1}$ and $\Gamma_f := \max(1, \log(\max_i |z_i|))$ the *logarithmic root bound* of f. An interval $I = (a,b)$ is called *isolating* for a root z_i if I contains z_i and no other root of F. We define $\mathrm{mid}(I) := \frac{a+b}{2}$ to be the *center* and $w(I) := b - a$ to be the *width* of I. We will prove the following complexity result:

Main Result. *Given initial isolating intervals for the real roots of f, the variant of the algorithm* AQIR *which uses approximate multipoint evaluation refines all intervals to the width 2^{-L} using*

$$\tilde{O}(d(d\Gamma_f + \Sigma_f)^2 + dL) \tag{2}$$

bit operations, where $\tilde{O}$ means that we ignore logarithmic factors. To do so, our algorithm requires the coefficients of f at a precision *of at most*

$$\tilde{O}(d\Gamma_f + \Sigma_f + L)$$

bits after the binary point. For a polynomial f with integer coefficients of absolute value less than 2^τ, the bound in (2) simplifies to $\tilde{O}(d^3\tau^2 + dL)$.

3. Review on exact QIR

Abbott's QIR method [1, 19] is a hybrid of the simple (but inefficient) bisection method with a quadratically converging variant of the *secant method*. We refer to this method as EQIR, where "E" stands for "exact" in order to distinguish from the variant presented in Section 4[1]. Given an isolating interval $I = (a,b)$ for a real root ξ of f, we consider the secant through $(a, f(a))$ and $(b, f(b))$ (see also Figure 1). This secant intersects the real axis in the interval I, say at x-coordinate m. For I small enough, the secant should approximate the graph of the function above I quite well and, so, $m \approx \xi$ should hold. An EQIR step tries to exploit this fact:

The isolating interval I is (conceptually) subdivided into N subintervals of same size, using $N+1$ equidistant grid points. Each subinterval has width $\omega := \frac{w(I)}{N}$. Then m', the closest grid point to m, is computed and the sign of $f(m')$ is evaluated. If that sign equals the sign of $f(a)$, the sign of $f(m'+\omega)$ is evaluated. Otherwise, $f(m'-\omega)$ is evaluated. If the sign changes between the two computed values, the interval $(m', m'+\omega)$ or the interval $(m'-\omega, m')$, respectively, is set as new isolating interval for ξ. In this case, the EQIR step is called *successful*. Otherwise, the isolating interval remains unchanged, and the EQIR step is called *failing*. See Algorithm 1 for a description in pseudo-code.

In [19], the root refinement problem is analyzed using the just described EQIR method for the case of integer coefficients and exact arithmetic with rational numbers. For that, a sequence of EQIR steps is performed with $N = 4$ initially. After a successful EQIR step, N is squared for the next step; after a failing step, N is set to $\sqrt{N}$. If N drops to 2, a bisection step is performed, and N is set to 4 for the next step. In [19], a bound on the size of an interval is provided to guarantee success of every EQIR and, thus, quadratic convergence of the overall method.

[1] To avoid confusion, the approximate version presented later is also "exact" in the sense that the refined intervals are isolating, but the intermediate computations are only approximate.

Algorithm 1 EQIR: Exact Quadratic Interval Refinement

INPUT: $f \in \mathbb{R}[x]$ square-free, $I = (a,b)$ isolating, $N = 2^{2^i} \in \mathbb{N}$
OUTPUT: (J,N') with $J \subseteq I$ isolating for ξ and $N' \in \mathbb{N}$

1: **procedure** EQIR($f, I = (a,b), N$)
2: **if** $N = 2$, **return** (BISECTION(f,I),4).
3: $\omega \leftarrow \frac{b-a}{N}$
4: $m' \leftarrow a + \text{round}(N\frac{f(a)}{f(a)-f(b)})\omega$ $\triangleright\ m' \approx a + \frac{f(a)}{f(a)-f(b)}(b-a)$
5: $s \leftarrow \text{sign}(f(m'))$
6: **if** $s = 0$, **return** $([m',m'],\infty)$
7: **if** $s = \text{sign}(f(a))$ **and** $\text{sign}(f(m'+\omega)) = \text{sign}(f(b))$, **return** $([m',m'+\omega],N^2)$
8: **if** $s = \text{sign}(f(b))$ **and** $\text{sign}(f(m'-\omega)) = \text{sign}(f(a))$, **return** $([m'-\omega,m'],N^2)$
9: Otherwise, **return** $(I,\sqrt{N})$.
10: **end procedure**

4. Approximate QIR

The most important numerical operation in an EQIR step is the computation of $f(x_0)$ for values $x_0 \in I$. Notice that $f(x_0)$ is needed for determining the closest grid point m' to the secant (Step 4 of Algorithm 1), and its sign is required for checking for sign changes in subintervals (Steps 5-8).

What are the problems if f is a bitstream polynomial as in (1), so that $f(x_0)$ can only be evaluated up to a certain precision? First of all, $\frac{Nf(a)}{f(a)-f(b)}$ can only be computed approximately, too, which might lead to checking the wrong subinterval in the algorithm if m is close to the center of a subinterval. Even more seriously, if $f(x_0)$ is zero, then, in general, its sign can never be evaluated using any precision. Even if we exclude this case, the evaluation of $f(x_0)$ can become costly if x_0 is too close to a root of f. The challenge is to modify the QIR method such that it can cope with the uncertainties in the evaluation of f, requires as few precision as possible in a refinement step and still shows a quadratic convergence behavior eventually.

Bisection is a subroutine called in the QIR method if $N = 2$; before we discuss the general case, we first describe our variant of the bisection in the bitstream context. Note that we face the same problem: f might be equal or almost equal to zero at $\text{mid}(I)$, the center of I. We will overcome this problem by evaluating f

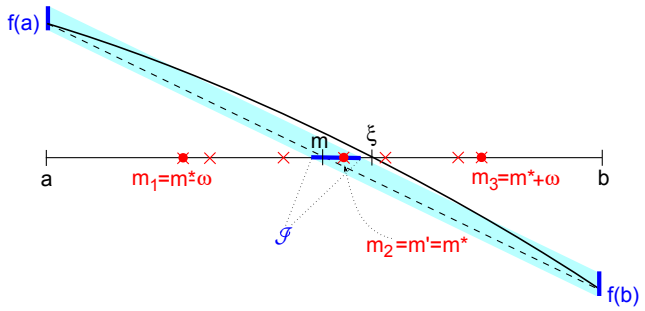

Figure 1: Illustration of an AQIR step for $N = 4$.

at several x-coordinates "in parallel". For that, we subdivide I into 4 equally wide parts using the subdivision points $m_j := a + j \cdot \frac{b-a}{4}$ for $1 \le j \le 3$. We also assume that the sign of f at a is already known. We choose a starting precision ρ and compute $f(m_1), \ldots, f(m_3)$ using interval arithmetic in precision ρ (cf. Section 5 for details). If less than 2 out of 3 signs have been determined using precision ρ, we set $\rho \leftarrow 2\rho$ and repeat the calculation with increased precision. Once the sign at at least 2 subdivision points is determined, we can determine a subinterval of at most half the size of I that contains ξ (Algorithm 2). We will refer to this algorithm as "bisection", although the resulting interval may sometimes be only a quarter of the original size. Note that f can only become zero at one of the subdivision points which guarantees termination also in the bitstream context. Moreover, at least 2 of the 3 subdivision points have a distance of at least $\frac{b-a}{8}$ to ξ. This asserts that the function value at these subdivision points is reasonably large and leads to an upper bound of the required precision (Lemma 5).

We next describe our bitstream variant of the QIR method that we call *approximate quadratic interval refinement*, or AQIR for short (see also Figure 1 for the illustration of an AQIR step for $N = 4$). Compared to the exact variant, we replace two substeps. In Step 4, we replace the computation of $\lambda := N\frac{f(a)}{f(a)-f(b)}$ as follows: For a working precision ρ, we evaluate $f(a)$ and $f(b)$ via interval arithmetic with precision ρ (blue vertical intervals in the above figure) and evaluate $N\frac{f(a)}{f(a)-f(b)}$ with interval arithmetic accordingly (cf. Section 5). Let $J = (c,d)$ denote the resulting interval (in Figure 1, $\mathscr{I} = a + J \cdot \frac{b-a}{N}$ is the intersection of the stripe defined by the interval evaluations of $f(a)$ and $f(b)$ with the real axis). If the width $w(J)$ of J is more than $\frac{1}{4}$, we set ρ to 2ρ and retry. Otherwise, let ℓ be the integer closest to $\mathrm{mid}(J)$ and set $m^* := a + \ell \cdot \frac{b-a}{N}$. For $m = a + \frac{f(a)}{f(a)-f(b)}(b-a)$

as before and $m_j := a + j \cdot \frac{b-a}{N}$ (red dots) for $j = 0, \ldots, N$, the following Lemma shows that the computed $m^* = m_\ell$ indeed approximates m on the m_j-grid:

Lemma 1. *Let m be inside the subinterval $[m_j, m_{j+1}]$. Then, $m^* = m_j$ or $m^* = m_{j+1}$. Moreover, let $m' \in \{m_j, m_{j+1}\}$ be the point that is closer to m. If $|m - m'| < \frac{b-a}{4N}$, then $m^* = m'$.*

Proof. Let $\lambda := N \frac{f(a)}{f(a)-f(b)}$ and J the interval computed by interval arithmetic as above, with width at most $\frac{1}{4}$. Since $m = f(a) + \lambda \frac{b-a}{N} \in [m_j, m_{j+1}]$, it follows that $j \leq \lambda \leq j+1$. By construction, $\lambda \in J$. Therefore, $|\lambda - \mathrm{mid}(J)| \leq \frac{1}{8}$ and, thus, it follows that $\mathrm{mid}(J)$ can only be rounded to j or $j+1$. Furthermore, for $m' = m_j$, $|m - m'| < \frac{b-a}{4N}$ implies that $|\lambda - j| < \frac{1}{4}$. It follows that $|\mathrm{mid}(J) - j| < \frac{3}{8}$ by triangle inequality, so $\mathrm{mid}(J)$ must be rounded to j. The case $m' = m_{j+1}$ is analogous. □

The second substep to replace in the QIR method is to check for sign changes in subintervals in Steps 5-8. As before, we set $\omega := w(I)/N$. Instead of comparing the signs at m' and $m' \pm \omega$, we choose seven subdivision points (red crosses in Figure 1), namely

$$m^* - \omega, m^* - \frac{7\omega}{8}, m^* - \frac{\omega}{2}, m^*, m^* + \frac{\omega}{2}, m^* - \frac{7\omega}{8}, m^* + \omega. \tag{3}$$

In case that $m^* = a$ or $m^* = b$, we only choose the 4 points of (3) that lie in I. For a working precision ρ, we evaluate the sign of f at all subdivision points using interval arithmetic. If the sign remains unknown for more than one point, we set ρ to 2ρ and retry. After the sign is determined for all except one of the points, we look for a sign change in the sequence. If such a sign change occurs, we set the corresponding interval I^* as isolating and call the AQIR step *successful*. Otherwise, we call the step *failing* and keep the old isolating interval. As in the exact case, we square up N after a successful step, and reduce it to its square root after a failing step. See Algorithm 3 for a complete description.

Note that, in case of a successful step, the new isolating interval I^* satisfies $\frac{1}{8N} w(I) \leq w(I^*) \leq \frac{1}{N} w(I)$. Also, similar to the bisection method, the function can only be zero at one of the chosen subdivision points, and the function is guaranteed to be reasonably large for all but one of them, which leads to a bound on the necessary precision (Lemma 7). The reader might wonder why we have chosen a non-equidistant grid involving the subdivision points $m^* \pm \frac{7}{8}\omega$. The reason is that these additional points allow us to give a success guarantee of the method under

Algorithm 2 Approximate Bisection

INPUT: $f \in \mathbb{R}[x]$ square-free, $I = (a,b)$ isolating, $s = \text{sign}(f(a))$
OUTPUT: $J \subseteq I$ isolating with $2 \cdot w(J) \leq w(I)$.

1: **procedure** APPROXIMATE_BISECTION($f, I = (a,b), s$)
2: $V \leftarrow [a + (i-1) \cdot \frac{b-a}{4}, i = 1, \ldots, 5]$
3: $S = [s, 0, 0, 0, -s]$
4: $\rho \leftarrow 2$
5: **while** S contains more than one zero **do**
6: **for** i=2,...,4 **do**
7: If $S[i] = 0$, set $S[i] \leftarrow \text{sign}\,\mathfrak{B}(f(V[i]), \rho)$
8: **end for**
9: $\rho \leftarrow 2\rho$
10: **end while**
11: Find v, w, such that $S[v] \cdot S[w] = -1 \wedge (v+1 = w \vee (v+2 = w \wedge S[v+1] = 0))$
12: **return** $(V[v], V[w])$
13: **end procedure**

certain assumptions in the following lemma which is the basis to prove quadratic convergence if the interval is smaller than a certain threshold (Section 6.2).

Lemma 2. *Let $I = (a,b)$ be an isolating interval for some root ξ of f, $s = \text{sign}(f(a))$ and m as before. If $|m - \xi| < \frac{b-a}{8N} = \frac{\omega}{8}$, then* AQIR*$(f, I, N, s)$ succeeds.*

Proof. Let m^* be the subdivision point selected by the AQIR method. We assume that $m^* \notin \{a,b\}$; otherwise, a similar (simplified) argument applies. By Lemma 1, $m \in [m^* - \frac{3}{4}\omega, m^* + \frac{3}{4}\omega]$ and, thus, $\xi \in (m^* - \frac{7}{8}\omega, m^* + \frac{7}{8}\omega)$. It follows that the leftmost two points of (3) have a different sign than the rightmost two points of (3). Since the sign of f is evaluated for at least one value on each side, the algorithm detects a sign change and, thus, succeeds. □

5. Analysis of an AQIR step

The running time of an AQIR step depends on the maximal precision ρ needed in the two while loops (Step 5, Steps 11-15) of Algorithm 3. The termination criterion of both loops is controlled by evaluations of the form $\mathfrak{B}(E, \rho)$, where E is some polynomial expression and ρ is the current working precision.

We specify recursively what we understand by evaluating E at a real valued point $x = x_0$ in precision ρ with interval arithmetic, where $\rho \in \mathbb{N}$. For $E = x$ or $E = c$, with c an arbitrary constant, we define $\mathrm{down}(E, x_0, \rho)$ to be the maximal $E_0 \leq E(x_0)$ such that $E_0 = \frac{k}{2^\rho}$ for some integer k. The same way $\mathrm{up}(E, x_0, \rho)$ is the minimal $E_0 \geq E(c)$ with E_0 of the same form. Notice that, if we can ask for all bits before and the first ρ bits after the binary representation of the values $x_0 \in \mathbb{R}$ and c, then $\mathrm{down}(E, x_0, \rho)$ and $\mathrm{up}(E, x_0, \rho)$ can be exactly determined. We extend the definitions of $\mathrm{down}(E, x_0, \rho)$ and $\mathrm{up}(E, x_0, \rho)$ to general arithmetic expressions by the following rules (we leave out x_0 and ρ for brevity):

$$\begin{aligned}
\mathrm{down}(E_1 + E_2) &:= \mathrm{down}(E_1) + \mathrm{down}(E_2) \\
\mathrm{up}(E_1 + E_2) &:= \mathrm{up}(E_1) + \mathrm{up}(E_2) \\
\mathrm{down}(E_1 \cdot E_2) &:= \mathrm{down}(\min\{\mathrm{down}(E_1)\mathrm{down}(E_2), \mathrm{up}(E_1)\mathrm{up}(E_2), \\
&\quad \mathrm{up}(E_1)\mathrm{down}(E_2), \mathrm{down}(E_1)\mathrm{up}(E_2)\}) \\
\mathrm{up}(E_1 \cdot E_2) &:= \mathrm{up}(\max\{\mathrm{down}(E_1)\mathrm{down}(E_2), \mathrm{down}(E_1)\mathrm{up}(E_2), \\
&\quad \mathrm{up}(E_1)\mathrm{down}(E_2), \mathrm{up}(E_1)\mathrm{up}(E_2)\}) \\
\mathrm{down}(1/E_1) &:= \mathrm{down}(1/\mathrm{up}(E_1)) \\
\mathrm{up}(1/E_1) &:= \mathrm{up}(1/\mathrm{down}(E_1))
\end{aligned}$$

Finally, we define the interval $\mathfrak{B}(E, x_0, \rho) := [\mathrm{down}(E, x_0, \rho), \mathrm{up}(E, x_0, \rho)]$. For short, we also write $\mathrm{down}(E(x_0), \rho)$ for $\mathrm{down}(E, x_0, \rho)$, $\mathrm{up}(E(x_0), \rho)$ for $\mathrm{up}(E, x_0, \rho)$, and $\mathfrak{B}(E(x_0), \rho)$ for $\mathfrak{B}(E, x_0, \rho)$. By definition, the exact value of E evaluated at $x = x_0$ is guaranteed to be contained in $\mathfrak{B}(E(x_0), \rho)$. We assume that polynomials $f \in \mathbb{R}[x]$ are evaluated according to the Horner scheme, and when evaluating $f(x_0)$ with precision ρ, the above rules apply in each arithmetic step. The next lemma provides a worst case bound on the size of the resulting interval $\mathfrak{B}(f(x_0), \rho)$ under certain conditions. We further remark that, in an actual implementation, $\mathfrak{B}(E(x_0), \rho)$ is usually much smaller than the worst case bound derived here. Nevertheless, our complexity analysis is based on the latter bound. Throughout the following considerations , $\Gamma \in \mathbb{N}$ denotes an integer upper bound on the root bound Γ_f, and, in particular $\log |z_i| \leq \Gamma$ for all roots z_i of f.

Lemma 3. *Let f be a polynomial as in (1), $x_0 \in \mathbb{R}$ with $|x_0| \leq 2^{\Gamma+2}$, and $\rho \in \mathbb{N}$. Then,*

$$|f(x_0) - \mathrm{down}(f(x_0), \rho)| \leq 2^{-\rho+1}(d+1)^2 2^{\tau + d(\Gamma+2)} \tag{4}$$

$$|f(x_0) - \mathrm{up}(f(x_0), \rho)| \leq 2^{-\rho+1}(d+1)^2 2^{\tau + d(\Gamma+2)} \tag{5}$$

In particular, $\mathfrak{B}(f(x_0), \rho)$ has a width of at most $2^{-\rho+2}(d+1)^2 2^{\tau + d(\Gamma+2)}$.

Proof. We do induction on d. The statement is clearly true for $d = 0$. For $d > 0$, we write $f(x_0) = a_0 + x_0 g(x_0)$ with $a_0 \in \mathbb{R}$ the constant coefficient of f and g of degree $d-1$. Note that, for any real value x, $|\text{down}(x,\rho) - x| < 2^{-\rho}$, same for up. Therefore, we can bound as follows (again, leaving ρ out for simplicity):

$$\begin{aligned}|f(x_0) - \text{down}(f(x_0))| &= |a_0 + x_0 g(x_0) - \text{down}(a_0 + x_0 g(x_0))| = \\ &= |a_0 + x_0 g(x_0) - \text{down}(a_0) - \text{down}(x_0 g(x_0))| \\ &\leq |x_0 g(x_0) - \text{down}(x_0 g(x_0))| + 2^{-\rho}\end{aligned}$$

Note that $\text{down}(x_0 \cdot g(x_0)) = \text{down}(H_1(x_0) \cdot H_2(g(x_0)))$ where $H_{1,2} = \text{down}$ or $H_{1,2} = \text{up}$. Moreover, we can write $H_1(x_0) = x_0 - \varepsilon$ with $|\varepsilon| < 2^{-\rho}$. Therefore, we can rearrange

$$\begin{aligned}& |x_0 g(x_0) - \text{down}(x_0 g(x_0))| + 2^{-\rho} \leq |x_0 g(x_0) - (x_0 - \varepsilon) \cdot H_2(g(x_0))| + 2^{-\rho+1} \\ \leq\ & |x_0 g(x_0) - x_0 \cdot H_2(g(x_0))| + |\varepsilon| \cdot |H_2(g(x_0))| + 2^{-\rho+1} \\ \leq\ & |x_0| \cdot |g(x_0) - H_2(g(x_0))| + 2^{-\rho} |H_2(g(x_0))| + 2^{-\rho+1}\end{aligned}$$

By a simple inductive proof on the degree, we can show that both $|\text{up}(g(x_0))|$ and $|\text{down}(g(x_0))|$ are bounded by $d 2^{\tau + d(\Gamma+2)}$. Using that and the induction hypothesis yields

$$\begin{aligned}& |x_0| \cdot |g(x_0) - h(g(x_0))| + 2^{-\rho} |H_2(g(x_0))| + 2^{-\rho+1} \\ <\ & 2^{\Gamma+2} 2^{-\rho+1} d^2 2^{\tau + (d-1)(\Gamma+2)} + 2^{-\rho} d 2^{\tau + d(\Gamma+2)} + 2^{-\rho+1} \\ \leq\ & 2^{-\rho+1} (d^2 + d + 1) 2^{\tau + d(\Gamma+2)} \leq 2^{-\rho+1} (d+1)^2 2^{\tau d}\end{aligned}$$

The bound for $|f(x_0) - \text{up}(f(x_0))|$ follows in the same way. □

For the sake of simplicity, we decided to assume fixed-point arithmetic instead of floating point arithmetic, that means, ρ determines the number of bits *after the binary point*. We remark that there exists extensive work on interval arithmetic (e.g. [30]), especially for floating point arithmetic; e.g., in [28, Theorem], a similar bound on the absolute error of floating point arithmetic as in Lemma 3 is given. However, for the sake of self contained representation, we integrated the results that are needed for our algorithm in this section. We further remark that we are confident that our overall complexity results carry over when using floating point arithmetic. Namely, polynomial evaluation is the crucial operation used in the overall algorithm and our estimates for the needed precision can eventually be traced back to Lemma 3.

We can now analyze the required working precision of approximate bisection and of an AQIR step next. We exploit that, whenever we evaluate f at t subdivision points, $t-1$ of them have a certain minimal distance to the root in the isolating interval. The following lemma gives a lower bound on $|f(x_0)|$ for such a point x_0, given that it is sufficiently far away from any other root of f.

Lemma 4. *Let f be as in (1), $\xi = z_{i_0}$ a real root of f and x_0 be a real value with distance $|x_0 - z_i| \geq \frac{\sigma_i}{4}$ to all* real *roots $z_i \neq z_{i_0}$. Then,*

$$|f(x_0)| > |\xi - x_0| \cdot 2^{-(2d+\Gamma+\Sigma_f)}.$$

(recall the notations from Section 1 for the definitions of σ_i and Σ_f)

Proof. For each non-real root z_i of f, there exists a complex conjugate root $\bar{z}_i$ and, thus, we have $|x_0 - z_i| \geq \mathrm{Im}(z_i) \geq \frac{\sigma_i}{2} > \frac{\sigma_i}{4}$ for all $i = m+1, \ldots, d$ as well. It follows that

$$\begin{aligned} |f(x_0)| = |a_d \prod_{i=1}^{d} (x_0 - z_i)| &= |a_d| \cdot |\xi - x_0| \cdot \prod_{i=1,\ldots,d : i \neq i_0} |x_0 - z_i| \\ &\geq |\xi - x_0| \cdot \frac{4}{\sigma_{i_0}} \cdot \prod_{i=1}^{d} \frac{\sigma_i}{4} > |\xi - x_0| \cdot 2^{-2d-\Gamma} \cdot 2^{-\Sigma_f}, \end{aligned}$$

where the last inequality uses that $|z_i| \leq 2^{\Gamma}$ and, thus, $\sigma(z_i) \leq 2^{\Gamma+1}$. □

We next analyze an approximate bisection step.

Lemma 5. *Let f be a polynomial as in (1), $I = (a,b) \subset (-2^{\Gamma+2}, 2^{\Gamma+2})$ be an isolating interval for a root $\xi = z_{i_0}$ of f and $s = \mathrm{sign}(f(a))$. Then, Algorithm 2 applied on (f, I, s) requires a maximal precision of*

$$\begin{aligned} \rho_0 &:= 2\log(b-a)^{-1} + 4\log(d+1) + 8d + 10 + 2(d+1)\Gamma + \tau + 2\Sigma_f \\ &= O(\log(b-a)^{-1} + \tau + d\Gamma + \Sigma_f), \end{aligned}$$

and its bit complexity is bounded by $\tilde{O}(d(\log(b-a)^{-1} + \tau + d\Gamma + \Sigma_f))$.

Proof. Consider the three subdivision points $m_j := a + j \cdot \frac{b-a}{4}$, where $1 \leq j \leq 3$, and an arbitrary real root $z_i \neq \xi$ of f. Note that $|m_j - z_i| > \frac{b-a}{4}$ because the segment from m_j to z_i spans at least a quarter of (a,b). Moreover, $|\xi - m_j| \leq \frac{3}{4}(b-a)$, and so

$$\sigma_i \leq |\xi - z_i| \leq |\xi - m_j| + |m_j - z_i| \leq \frac{3}{4}(b-a) + |m_j - z_i| \leq 4|m_j - z_i|.$$

It follows that m_j has a distance to z_i of at least $\frac{\sigma_i}{4}$. Hence, we can apply Lemma 4 to each m_j, that is, we have $|f(m_j)| > |\xi - m_j| \cdot 2^{-(2d+\Gamma+\Sigma_f)}$. Since the signs of f at the endpoints of I are known, it suffices to compute the signs of f at two of the three subdivision points. For at least two of these points, the distance of m_j to ξ is at least $\frac{b-a}{8}$, thus, we have $|f(m_j)| > |b-a| \cdot 2^{-(2d+3+\Gamma+\Sigma_f)}$ for at least two points. Then, due to Lemma 3, we can use interval arithmetic with a precision ρ to compute these signs if ρ satisfies

$$2^{-\rho+2}(d+1)^2 2^{\tau+d(\Gamma+2)} \leq (b-a) \cdot 2^{-(2d+3+\Gamma+\Sigma_f)},$$

which is equivalent to $\rho \geq \frac{\rho_0}{2}$. Since we double the precision in each step, we will eventually succeed with a precision smaller than ρ_0. The bit complexity for an arithmetic operation with fixed precision ρ is $\tilde{O}(\rho + d\tau)$. Namely, since the absolute value of each subdivision point is bounded by $O(\tau)$, the results in the intermediate steps have magnitude $O(d\tau)$ and we consider ρ bits after the binary point. At each subdivision point, we have to perform $O(d)$ arithmetic operations for the computation of $f(m_j)$, thus, the costs for these evaluations are bounded by $\tilde{O}(d(d\tau+\rho))$ bit operations. Since we double the precision in each iteration, the total costs are dominated by the last successful evaluation and, thus, we have to perform $\tilde{O}(d(\rho_0 + d\tau)) = \tilde{O}(d(\log(b-a)^{-1} + d\tau + \Sigma_f))$ bit operations. □

We proceed with the analysis of an AQIR step. In order to bound the required precision, we need additional properties of the isolating interval.

Definition 6. *Let f be as in (1), $I := (a,b)$ be an isolating interval of a root ξ of f. We call I* normal[2] *if*

- $I \subseteq (-2^{\Gamma+2}, 2^{\Gamma+2})$,
- $|p - z_i| > \frac{\sigma_i}{4}$ *for every* $p \in I$ *and* $z_i \neq \xi$, *and*
- $\min\{|f(a)|, |f(b)|\} \geq 2^{-(28+2\tau+17d\Gamma+2\Sigma_f-5\log(b-a))}$.

In simple words, a normal isolating interval has a reasonable distance to any other root of f, and the function value at the endpoints is reasonably large. We will later see that it is possible to get normal intervals by a sequence of approximate bisection steps.

[2]The reader may notice that the definition of "normal" depends on the upper bound Γ on Γ_f. Throughout our argument, we assume that such an initial Γ is given. We will finally choose a Γ which approximates Γ_f up to an (addative) error of $O(\log d)$.

Algorithm 3 Approximate Quadratic interval refinement

INPUT: $f \in \mathbb{R}[x]$ square-free, $I = (a,b)$ isolating, $N = 2^{2^i} \in \mathbb{N}$, $s = \text{sign}(f(a))$
OUTPUT: (J, N') with $J \subseteq I$ isolating and $N' \in \mathbb{N}$

procedure AQIR($f, I = (a,b), N$)
  if $N = 2$, **return** (APPROXIMATE_BISECTION(f, I, s),4).
  $\omega \leftarrow \frac{b-a}{N}$
  $\rho \leftarrow 2$
  while $J \leftarrow \mathfrak{B}(N\frac{f(a)}{f(a)-f(b)}, \rho)$ has width $> \frac{1}{4}$, set $\rho \leftarrow 2\rho$
  $m^* \leftarrow a + \text{round}(\text{mid}(J)) \cdot \omega$
  if $m^* = a$, $s \leftarrow 4, V \leftarrow [m^*, m^* + \frac{1}{2}\omega, m^* + \frac{7}{8}\omega, m^* + \omega], S \leftarrow [s,0,0,0]$
  if $m^* = b$, $s \leftarrow 4, V \leftarrow [m^* - \omega, m^* - \frac{7}{8}\omega, m^* - \frac{1}{2}\omega, m^*], S \leftarrow [0,0,0,-s]$
  if $a < m^* < b$, $s \leftarrow 7, V \leftarrow [m^* - \omega, m^* - \frac{7}{8}\omega, m^* - \frac{1}{2}\omega, m^*, m^* + \frac{1}{2}\omega, m^* + \frac{7}{8}\omega, m^* + \omega], S \leftarrow [0,0,0,0,0,0,0]$
  $\rho \leftarrow 2$
  while S contains more than one zero **do**
    for i=1,...,s **do**
      If $S[i] = 0$, set $S[i] \leftarrow \text{sign}\,\mathfrak{B}(f(V[i]), \rho)$
    end for
    $\rho \leftarrow 2\rho$
  end while
  If $\exists v, w : S[v] \cdot S[w] = -1 \wedge (v+1 = w \vee (v+2 = w \wedge S[v+1] = 0))$ **return** $((V[v], V[w]), N^2)$
  Otherwise, return $(I, \sqrt{N})$
end procedure

Lemma 7. *Let f be a polynomial as in (1), $I = (a,b)$ be a normal isolating interval for a root $\xi = z_{i_0}$ of f with $s = \text{sign}(f(a))$, and let $N \leq 2^{2(\Gamma+4-\log(b-a))}$. Then, the* AQIR *step for (f, I, N, s) requires a precision of at most*

$$\rho_{max} := 87d\tau + 17d\Gamma + 4\Sigma_f - 14\log(b-a)$$

and, therefore, its bit complexity is bounded by

$$\tilde{O}(d(\tau + d\Gamma + \Sigma_f - \log(b-a))).$$

Moreover, the returned interval is again normal.

Proof. We have to distinguish two cases. For $N > 2$, we consider the two while-loops in Algorithm 3. In the first loop (Step 5), we evaluate $N\frac{f(a)}{f(a)-f(b)}$ via interval arithmetic, doubling the precision ρ until the width of the resulting interval J is less than or equal to $1/4$. The following considerations show that we can achieve this if ρ satisfies

$$2^{-\rho+2}(d+1)^2 2^{\tau+d(\Gamma+2)} \leq \frac{\min(|f(a)|,|f(b)|)}{32N}. \tag{6}$$

W.l.o.g., we assume $f(a) > 0$. If ρ satisfies the above condition, then, due to Lemma 3, $\mathfrak{B}(N \cdot f(a), \rho)$ is contained within the interval

$$[Nf(a) - \frac{|f(a)|}{32}, Nf(a) + \frac{|f(a)|}{32}] = Nf(a) \cdot [1 - \frac{1}{32N}, 1 + \frac{1}{32N}]$$

and $\mathfrak{B}(f(a) - f(b), \rho)$ is contained within the interval

$$[f(a) - f(b) - \frac{|f(a) - f(b)|}{32N}, f(a) - f(b) + \frac{|f(a) - f(b)|}{32N}] = (f(a) - f(b)) \cdot [1 - \frac{1}{32N}, 1 + \frac{1}{32N}],$$

where the latter result uses the fact that $f(a)$ and $f(b)$ have different signs. It follows that $\mathfrak{B}(N\frac{f(a)}{f(a)-f(b)}, \rho)$ is contained within $\frac{Nf(a)}{f(a)-f(b)} \cdot [(1-\frac{1}{32N})/(1+\frac{1}{32N}), (1+\frac{1}{32N})/(1-\frac{1}{32N})]$, and a simple computation shows that $N \cdot [(1-\frac{1}{32N})/(1+\frac{1}{32N}), (1+\frac{1}{32N})/(1-\frac{1}{32N})]$ has width less than $1/4$. Hence, since $\frac{f(a)}{f(a)-f(b)}$ has absolute value less than 1, $\mathfrak{B}(N\frac{f(a)}{f(a)-f(b)}, \rho)$ has width less than $1/4$ as well. The bound (6) on ρ also writes as

$$\rho \geq 7 + 2\log(d+1) + \tau + d\Gamma + 2d + \log N + \log\min(|f(a), f(b)|)^{-1}$$

and since we double ρ in each iteration, computing $N\frac{f(a)}{f(a)-f(b)}$ via interval arithmetic up to an error of $1/4$ demands for a precision

$$\begin{aligned}\rho &< 14 + 4\log(d+1) + 2\tau + 2d\Gamma + 4d + 2\log N + 2\log\min(|f(a), f(b)|)^{-1} \\ &< 14 + 2\tau + 10d\Gamma + 2\log N + 2\log\min(|f(a), f(b)|)^{-1},\end{aligned}$$

Since I is normal and because of the posed condition on N, we can bound this by

$$\begin{aligned}\rho &< 11d\tau + 4(\tau + 5 - \log(b-a)) + 2(32d\tau + 2\Sigma_f - 5\log(b-a)) \\ &< 87d\tau + 4\Sigma_f - 14\log(b-a) < \rho_{max}.\end{aligned}$$

We turn to the second while loop of Algorithm 3 (Steps 11-15) where f is evaluated at the subdivision points $m^* - \omega, m^* - \frac{7\omega}{8}, \ldots, m^* + \omega$ as defined in (3). Since the interval is normal, we can apply Lemma 4 to each of the seven subdivision points. Furthermore, at least six of these points have distance $\geq \frac{b-a}{16N}$ to the root ξ and, thus, for these points, $|f|$ is larger than $\frac{b-a}{16N} \cdot 2^{-(2d+\tau+\Sigma_f)}$. Then, according to Lemma 6, it suffices to use a precision ρ that fulfills

$$2^{-\rho+2}(d+1)^2 2^{\tau+d(\Gamma+2)} \leq \frac{b-a}{16N} \cdot 2^{-(2d+\Gamma+\Sigma_f)}, \text{ or}$$

$$\rho \geq \rho_1 := 6 + 2\log(d+1) + \tau + d\Gamma + 4d + \Gamma + \Sigma_f + \log N - \log(b-a).$$

The same argumentation as above then shows that the point evaluation will be performed with a maximal precision of less than

$$\begin{aligned} 2\rho_1 &< 2(6 + \tau + 7d\Gamma + \Gamma + \Sigma_f + \log N - \log(b-a)) \\ &\leq 12 + 2\tau + 14d\Gamma + 2\Gamma + 2\Sigma_f + 4(\Gamma + 4 - \log(b-a)) - \log(b-a) \\ &\leq 28 + 2\tau + 17d\Gamma + 2\Sigma_f - 5\log(b-a) \end{aligned}$$

which is bounded by ρ_{max}. Moreover, at the new endpoints a' and b', $|f|$ is at least

$$2^{-2\rho_1} \geq 2^{-(28+2\tau+17d\Gamma+2\Sigma_f-5\log(b-a))} \geq 2^{-(28+2\tau+17d\Gamma+2\Sigma_f-5\log(b'-a'))}$$

which proves that $I' = (a', b')$ is again normal.

It remains the case of $N = 2$, where a bisection step is performed. It is straightforward to see with Lemma 5 that the required precision is bounded by ρ_{max}, and in an analogue way as for the point evaluations for $N > 2$, we can see that the resulting interval is again normal. By the same argument as in Lemma 5, the overall bit complexity of the AQIR step is bounded by

$$\tilde{O}(d\rho_{max}) = \tilde{O}(d(d\tau + \Sigma_f - \log(b-a))). \quad \square$$

6. Root refinement

We next analyze the complexity of our original problem: Given a polynomial f as in (1) and isolating intervals for all its real roots, refine the intervals to a size of at most 2^{-L}. Our refinement method consists of two steps. First, we turn the isolating intervals into normal intervals by applying bisections repeatedly. Second, we call the AQIR method repeatedly on the intervals until each has a width of at most 2^{-L}. Algorithm 5 summarizes our method for root refinement.

We remark that depending on the properties of the root isolator used to get initial isolating intervals, the normalization can be skipped; this is for instance the case when using the isolator from [39]. We also emphasize that the normalization is unnecessary for the correctness of the algorithm, but required for the complexity bound: without normalized intervals, we cannot compute a comparable bound on the working precision demand of a single AQIR step.

6.1. Normalization

If there is only one isolating interval, it is easily shown that $(-2^{\Gamma+2}, 2^{\Gamma+2})$ is already a normal interval that isolates the corresponding root. Hence we assume that at least two isolating intervals are present. The normalization (Algorithm 4) consists of two steps: first, the isolating intervals are refined using approximate bisection until the distance between two consecutive intervals is at least three times larger than the size of the larger of the two involved intervals. This ensures that all points in an isolating interval are reasonably far away from any other root of f. In the second step, each interval is enlarged on both sides by an interval of at least the same size as itself. This ensures that the endpoints are sufficiently far away from any root of f to prove a lower bound of f at the endpoints. W.l.o.g., we also assume that the input intervals are contained in $(-2^{\Gamma+1}, 2^{\Gamma+1})$ because all roots are contained in that interval, so the leftmost and rightmost intervals can just be cut if necessary. Obviously, the resulting intervals are still isolating and disjoint from each other. Moreover, they do not become too small during the bisection process:

Lemma 8. *For $J_1, \ldots, J_m$ as returned by Alg. 4, $w(J_k) > \frac{1}{4}\sigma_k$.*

Proof. After the first for-loop, the distance d_k between any two consecutive intervals I_k and I_{k+1} satisfies $d_k \geq 3\max\{w(I_k), w(I_{k+1})\}$, thus $\sigma_k < w(I_k) + w(I_{k+1}) + d_k < 2d_k$. Hence, in the last step, each I_k is enlarged by *more than* $\sigma_k/8$ on each side. This proves that the corresponding enlarged intervals J_k have size *more than* $\sigma_k/4$. □

Lemma 9. *Algorithm 4 is correct, i.e., returns normal intervals.*

Proof. Let $J_1, \ldots, J_m$ denote the returned intervals, and fix some interval J_k containing the root z_k of f. We have to prove the three properties of Definition 6. The first property is clear because the initial interval are assumed to lie in $(-2^{\Gamma+1}, 2^{\Gamma+1})$, and they are extended by not more than 2^{Γ} to each side. In the proof of Lemma 8, we have already shown that I_k is eventually enlarged by more than $\sigma_k/8$ on each

Algorithm 4 Normalization

INPUT: $f \in \mathbb{R}[t]$ a polynomial as in (1), $I_1 = (a_1, b_1), \ldots, I_m = (a_m, b_m)$ disjoint isolating intervals in ascending order, $m \geq 2$, $s_1, \ldots, s_m$ with $s_k = \operatorname{sign}(f(\min I_k))$
OUTPUT: normal isolating intervals $J_1, \ldots, J_m$ with $z_k \in I_k \cap J_k$

procedure NORMALIZE($f, I_1, \ldots, I_m$)
  for k=1,...,m-1 **do**
    while $\min I_{k+1} - \max I_k < 3 \max\{w(I_k), w(I_{k+1})\}$ **do**
      if $w(I_k) > w(I_{k+1})$
      then APPROXIMATE_BISECTION(f, I_k, s_k)
      else APPROXIMATE_BISECTION(f, I_{k+1}, s_{k+1})
    end while
    $d_k \leftarrow \min I_{k+1} - \max I_k$
  end for
  $d_0 \leftarrow d_1$, $d_m \leftarrow d_{m-1}$
  for k=1,...,m **do**
    $J_k \leftarrow (a_k - d_{k-1}/4, b_k + d_k/4)$ ▷ enlarge I_k by more than $w(I_k)$ at both sides
  end for
  return $J_1, \ldots, J_m$
end procedure

side. More precisely, the right endpoint of J_k has distance at least $d_k/4 > \sigma_{k+1}/8$ to J_{k+1}, and the left endpoint of J_k has distance at least $d_{k-1}/4 > \sigma_{k-1}/8$ to J_{k-1}. It follows that, for each $x_0 \in J_k$, we have $|x_0 - z_{k\pm1}| > \sigma_{k\pm1}/4$, respectively. Hence, the second property in Definition 6 is satisfied for all real roots, and it is clearly satisfied for all non-real roots because the distance of such a root z_i to z_k is at least half the distance to the complex conjugate of z_i. For the third property of Definition 6, let e be one of the endpoints of J_k. We have just proved that the distance to every root z_i except z_k is more than $\frac{\sigma_i}{4}$ and $|e - z_k| > \sigma_k/8$. With an estimation similar as in the proof of Lemma 4, we obtain:

$$|f(e)| > \frac{\sigma_k}{8} \prod_{i \neq k} \frac{\sigma_i}{4} = \frac{1}{8} \cdot \frac{1}{4^{d-1}} 2^{-\Sigma_f} = 2^{-(2d+\Sigma_f+1)},$$

and $2^{-(2d+\Sigma_f+1)} \geq 2^{-(28+2\tau+17d\Gamma+2\Sigma_f-5\log(b-a))}$ because $\log(b-a) \leq \Gamma+2$ and $-\Sigma_f \leq d(\Gamma+1) < 2d\Gamma$. □

Algorithm 5 Root Refinement

INPUT: $f = \sum a_i x^i \in \mathbb{R}[t]$ a polynomial as in (1), isolating intervals $I_1, \ldots, I_m$ for the real roots of f in ascending order, $L \in \mathbb{Z}$
OUTPUT: isolating intervals $J_1, \ldots, J_m$ with $w(J_k) \leq 2^{-L}$

procedure ROOT_REFINEMENT($f, L, I_1, \ldots, I_m$)
  $s_k := \operatorname{sign}(a_d) \cdot (-1)^{m-k+1}$ ▷ $s_k = \operatorname{sign}(f(\min I_k))$
  $J_1, \ldots, J_m \leftarrow$ NORMALIZE($f, I_1, \ldots, I_m$)
  for k=1,...,m **do**
    $N \leftarrow 4$
    while $w(J_k) > 2^{-L}$ **do** $(J_k, N) \leftarrow$AQIR(f, J_k, N, s_k)
  end for
  return $J_1, \ldots, J_m$
end procedure

Lemma 10. *Algorithm 4 has a complexity of*

$$\tilde{O}(d(d\Gamma + \Sigma_f)(\tau + d\Gamma + \Sigma_f))$$

Proof. As a direct consequence of Lemma 8, each interval I_k is only bisected $O(\Gamma + \log(\sigma_k)^{-1})$ many times because each starting interval is assumed to be contained in $(-2^{\Gamma+1}, 2^{\Gamma+1})$. So the total number of bisections adds up to $O(d\Gamma + \Sigma_f)$ considering all roots of f. Also, the size of the isolating interval I_k is lower bounded by $\frac{3}{20} \cdot \sigma_k = 2^{-O(\Sigma_f + d\Gamma)}$, so that one approximate bisection step has a complexity of $\tilde{O}(d(\tau + d\Gamma + \Sigma_f))$ due to Lemma 5. □

6.2. The AQIR sequence

It remains to bound the cost of the calls of AQIR. We mostly follow the argumentation from [19], mostly referring to that article for technical proofs. We introduce the following convenient notation:

Definition 11. *Let $I_0 := I$ be a normal isolating interval for some real root ξ of f, $N_0 := 4$ and $s := \operatorname{sign}(\min I_0)$. The* AQIR sequence $(S_0, S_1, \ldots, S_{v_\xi})$ *is defined by*

$$S_0 := (I_0, N_0) = (I, 4) \quad S_i = (I_i, N_i) := \text{AQIR}(f, I_{i-1}, N_{i-1}, s) \text{ for } i \geq 1,$$

where v_ξ is the first index such that the interval I_{v_ξ} has width at most 2^{-L}. We say that $S_i \overset{\text{AQIR}}{\to} S_{i+1}$ succeeds *if* AQIR*(f, I_i, N_i, s) succeeds, and that $S_i \overset{\text{AQIR}}{\to} S_{i+1}$* fails *otherwise.*

As in [19], we divide the QIR sequence into two parts according to the following definition.

Definition 12. *For ξ a root of f, we define*

$$C_\xi := \frac{|f'(\xi)|}{8\left(\frac{d^2}{\sigma(\xi,f)}|f'(\xi)| + \sum_{i=2}^{d}\left(\frac{\sigma(\xi,f)}{d^2}\right)^{i-2}|f^{(i)}(\xi)|\right)}.$$

For $(S_0,\dots,S_{v_\xi})$ the QIR sequence of ξ, define k as the minimal index such that $S_k = (I_k,N_k) \overset{\text{AQIR}}{\to} S_{k+1}$ succeeds and $w(I_k) \le C_\xi$. We call $(S_0,\dots,S_k)$ linear sequence *and $(S_k,\dots,S_{v_\xi})$* quadratic sequence *of ξ*

Note that [19] defined a different threshold for splitting the QIR sequence, and the linear sequence was called *initial sequence* therein. We renamed it to avoid confusion with the initial normalization phase in our variant.

Quadratic convergence. We start by justifying the name "quadratic sequence". Indeed, it turns out that all but one AQIR step in the quadratic sequence are successful, hence, N is squared in (almost) every step and therefore, the refinement factor of the interval is doubled in (almost) every step. We first prove two important properties of C_ξ as defined in Defition 12:

Lemma 13. *Let $\xi \in \mathbb{C}$ be a root of f.*

1. $0 < C_\xi \le \frac{\sigma(\xi,f)}{8d^2}$
2. *Let $\mu \in \mathbb{C}$ be such that $|\xi-\mu| < C_\xi$. Then*

$$C_\xi < \frac{|f'(\xi)|}{8|f''(\mu)|}.$$

Proof. Note that all summands in the denominator of C_ξ are non-negative. Therefore, the first property follows immediately by removing all but the first summand in the denominator.

For the second property, we consider the Taylor expansion of $f''(\mu)$ in ξ:

$$f''(\mu) = \sum_{i=2}^{d}(\mu-\xi)^{i-2}\frac{f^{(i)}(\xi)}{(i-2)!}.$$

Because $|\mu - \xi| < C_\xi < \frac{\sigma(\xi)}{d^2}$ by the first property, we can bound

$$|f''(\mu)| < \sum_{i=2}^{d} \left(\frac{\sigma(\xi)}{d^2}\right)^{i-2} |f^{(i)}(\xi)|.$$

It follows that

$$\frac{|f'(\xi)|}{8|f''(\mu)|} > \frac{|f'(\xi)|}{8\left(\sum_{i=2}^{d}\left(\frac{\sigma(\xi)}{d^2}\right)^{i-2}|f^{(i)}(\xi)|\right)} > C_\xi$$

□

The following bound follows from considering the Taylor expansion of f at ξ in the expression for m:

Lemma 14. [19, Thm. 4.8] *Let (a,b) be isolating for ξ with width $\delta < C_\xi$ and m as in Lemma 2 (i.e., $m = a + \frac{f(a)}{f(a)-f(b)}(b-a)$). Then, $|m-\xi| \leq \frac{\delta^2}{8C_\xi}$.*

Proof. We consider the Taylor expansion of f at ξ. For a given $x \in (a,b)$, we have

$$f(x) = f'(\xi)(x-\xi) + \frac{1}{2}f''(\tilde{\xi})(x-\xi)^2$$

with some $\tilde{\xi} \in [x,\xi]$ or $\tilde{\xi} \in [\xi,x]$. Thus, we can simplify

$$|m-\xi| = \left|\frac{f(b)(a-\xi)-f(a)(b-\xi)}{f(b)-f(a)}\right| = \left|\frac{\frac{1}{2}(f''(\tilde{\xi}_1)(b-\xi)^2(a-\xi) - f''(\tilde{\xi}_2)(a-\xi)^2(b-\xi))}{f(b)-f(a)}\right|$$
$$\leq \frac{1}{2}|b-\xi||a-\xi| \cdot \frac{|f''(\tilde{\xi}_1)|(b-\xi)+|f''(\tilde{\xi}_2)|(\xi-a)}{|f(b)-f(a)|} \leq \frac{\delta^2 \max\{|f''(\tilde{\xi}_1)|,|f''(\tilde{\xi}_2)|\}}{2|f'(\nu)|}$$

for some $\nu \in (a,b)$. The Taylor expansion of f' yields $f'(\nu) = f'(\xi) + f''(\tilde{\nu})(\nu - \xi)$ with $\tilde{\nu} \in (a,b)$. Since $\delta \leq C_\xi$, it follows with Lemma 13

$$|f''(\tilde{\nu})(\nu-\xi)| \leq |f''(\tilde{\nu})|C_\xi \leq \frac{1}{8}|f'(\xi)|.$$

Therefore $|f'(\nu)| > \frac{7}{8}|f'(\xi)| > \frac{1}{2}|f'(\xi)|$, and it follows again with Lemma 13 that

$$|m-\xi| \leq \frac{\delta^2 \max\{|f''(\tilde{\xi}_1)|,|f''(\tilde{\xi}_2)|\}}{|f'(\xi)|} \leq \frac{\delta^2}{8\frac{|f'(\xi)|}{8\max\{|f''(\tilde{\xi}_1)|,|f''(\tilde{\xi}_2)|\}}} < \frac{\delta^2}{8C_\xi} \quad \square$$

Corollary 15. *Let I_j be an isolating interval for ξ of width $\delta_j \leq \frac{C_\xi}{N_j}$. Then, each call of the* AQIR *sequence*

$$(I_j, N_j) \overset{\text{AQIR}}{\to} (I_{j+1}, N_{j+1}) \overset{\text{AQIR}}{\to} \dots$$

succeeds.

Proof. We use induction on i. Assume that the first i AQIR calls succeed. Then, another simple induction shows that $\delta_{j+i} := w(I_{j+i}) \leq \frac{N_j \delta_j}{N_{j+i}} < \frac{C_\xi}{N_{j+i}}$, where we use that $N_{j+i} = N_{j+i-1}^2$. Then, according to Lemma 14, we have that

$$|m - \xi| \leq \delta_{j+i}^2 \frac{1}{8C_\xi} \leq \delta_{j+i} \frac{C_\xi}{N_{j+i}} \frac{1}{8C_\xi} = \frac{1}{8} \frac{\delta_{j+i}}{N_{j+i}},$$

with m as above. By Lemma 2, the AQIR call succeeds. □

Corollary 16. [19, Cor. 4.10] *In the quadratic sequence, there is at most one failing* AQIR *call.*

Proof. Let $(I_i, N_i) \overset{\text{AQIR}}{\to} (I_{i+1}, N_{i+1})$ be the first failing AQIR call in the quadratic sequence. Since the quadratic sequence starts with a successful AQIR call, the predecessor $(I_{i-1}, N_{i-1}) \overset{\text{AQIR}}{\to} (I_i, N_i)$ is also part of quadratic sequence, and succeeds. Thus we have the sequence

$$(I_{i-1}, N_{i-1}) \overset{\substack{\textit{Sucess} \\ \text{AQIR}}}{\to} (I_i, N_i) \overset{\substack{\textit{Fail} \\ \text{AQIR}}}{\to} (I_{i+1}, N_{i+1})$$

One observes easily that $w(I_{i+1}) = w(I_i) = \frac{w(I_{i-1})}{N_{i-1}} \leq \frac{C_\alpha}{N_{i-1}}$, and $N_{i+1} = \sqrt{N_i} = \sqrt{N_{i-1}^2} = N_{i-1}$. By Corollary 15, all further AQIR calls succeed. □

Notice that the quantity C_ξ as defined in Definition 12 defines a threshold for the width of an interval such that quadratic convergence of the QIR method can be guaranteed. For Newton iteration, there exist similar bounds (e.g. [6, Theorem 1]) based on Smale's α-theory, and we consider the quantity C_ξ as a corresponding counterpart to these bounds for the secant method.

Cost of the linear sequence. We bound the costs of refining the isolating interval of ξ to size C_ξ with AQIR. We first show that, on average, the AQIR sequence refines by a factor two in every second step. This shows in particular that refining using AQIR is at most a factor of two worse than refining using approximate bisection.

Lemma 17. *Let $(S_0,\dots,S_\ell)$ denote an arbitrary prefix of the* AQIR *sequence for ξ, starting with the isolating interval I_0 of width δ. Then, the width of I_ℓ is not larger than $\delta 2^{-(\ell-1)/2}$.*

Proof. Consider a subsequence $(S_i,\dots,S_{i+j})$ of $(S_0,\dots,S_\ell)$ such that $S_i \overset{\text{AQIR}}{\to} S_{i+1}$ is successful, but any other step in the subsequence fails. Because there are j steps in total, and thus $j-1$ consecutive failing steps, the successful step must have used a N with $N \geq 2^{2^{j-1}}$. Because $2^{j-1} \geq \frac{j}{2}$, it holds that

$$w(I_{i+j}) \leq \frac{w(I_i)}{N} \leq w(I_{i+j})2^{-2^{j-1}} \leq w(I_{i+j})2^{-j/2}.$$

Repeating the argument for maximal subsequences of this form, we get that either $w(I_\ell) \leq w(I_0)2^{-\ell/2}$ if the sequence starts with a successful step, or $w(I_\ell) \leq w(I_0)2^{-(\ell-1)/2}$ otherwise, because the second step must be successful in this case. □

We want to apply Lemma 7 to bound the bit complexity of a single AQIR step. The following lemma shows that the condition on N from Lemma 7 is always met in the AQIR sequence.

Lemma 18. *Let $(I_j,N_j) \overset{\text{AQIR}}{\to} (I_{j+1},N_{j+1})$ be a call in an* AQIR *sequence and $I_j := (a,b)$. Then, $N_j \leq 2^{2(\Gamma+4-\log(b-a))}$.*

Proof. We do induction on j. Note that $I_0 \subset (-2^{\Gamma+2},2^{\Gamma+2})$ by normality, hence $b-a \leq 2^{\Gamma+3}$. It follows that $2^{2(\Gamma+4-\log(b-a))} \geq 4 = N_0$. Assume that the statement is true for $j-1$. If the previous step $(I_{j-1},N_{j-1}) \overset{\text{AQIR}}{\to} (I_j,N_j)$ is failing, then $N_j = \sqrt{N_{j-1}}$ and the isolating interval remains unchanged, so the statement is trivially correct. If the step is successful, then it holds that $(b-a) \leq \frac{2^{\Gamma+3}}{\sqrt{N_j}}$. By rearranging terms, we get that $N_j \leq 2^{2(\Gamma+3-\log(b-a))}$. □

It follows inductively that the conditions of Lemma 7 are met for each call in the AQIR sequence because I_0 is normal by construction. Therefore, the linear sequence for a root ξ of f is computed with a bit complexity of

$$\tilde{O}((\Gamma+\log(C_\xi)^{-1})d(\log(C_\xi^{-1})+\tau+d\Gamma+\Sigma_f)) \tag{7}$$

because $O(\Gamma+\log(C_\xi^{-1}))$ steps are necessary to refine the interval to a size smaller than C_ξ by Lemma 17, and the bit complexity is bounded by $\tilde{O}(d(\log(C_\xi^{-1})+\tau+$

$d\Gamma+\Sigma_f))$ with Lemma 7. It remains to bound $\log(C_\xi)^{-1}$; we do so by bounding the sum of all $\log(C_\xi)^{-1}$ with the following lemma.

Lemma 19. $\sum_{i=1}^{m}\log(C_{z_i})^{-1} = O(d(\Gamma+\log d)+\Sigma_f))$

Proof. We note that

$$\sum_{\ell=1}^{m}\log(C_{z_\ell})^{-1} = \sum_{\ell=1}^{m}\log\left(8\cdot\left(\frac{d^2}{\sigma_\ell}+\sum_{i=2}^{d}\left(\frac{\sigma_\ell}{d^2}\right)^{i-2}\left|\frac{f^{(i)}(z_\ell)}{f'(z_\ell)}\right|\right)\right).$$

We focus on the quotient $\left|\frac{f^{(i)}(z_\ell)}{f'(z_\ell)}\right|$. Let $z'_1,\dots,z'_{d-1}$ denote the (not necessarily distinct) roots of f'. Note that for $x\in\mathbb{C}$ and any $i\geq 1$,

$$f^{(i)}(x) = a_d \sum_{\substack{X\subseteq\{1,\dots,n-1\}\\|X|=i-1}}\prod_{\substack{j\in\{1,\dots,d-1\}\\ j\notin X}}(x-z'_j)$$

Therefore, the quotient writes as

$$\left|\frac{f^{(i)}(z_\ell)}{f'(z_\ell)}\right| = \left|\sum_{\substack{X\subseteq\{1,\dots,d-1\}\\|X|=i-1}}\prod_{j\in X}\frac{1}{z_\ell-z'_j}\right| \leq \sum_{\substack{X\subseteq\{1,\dots,d-1\}\\|X|=i-1}}\prod_{j\in X}\frac{1}{|z_\ell-z'_j|}.$$

Since $|z_\ell-z'_j|\geq\frac{\sigma_\ell}{d}$ [14, Thm.8], we can further bound this to

$$\sum_{\substack{X\subseteq\{1,\dots,d-1\}\\|X|=i-1}}\prod_{j\in X}\frac{1}{|z_\ell-z'_j|} \leq \sum_{\substack{X\subseteq\{1,\dots,d-1\}\\|X|=i-1}}\prod_{j\in X}\frac{d}{\sigma_\ell} \leq \sum_{\substack{X\subseteq\{1,\dots,d-1\}\\|X|=i-1}}\left(\frac{d}{\sigma_\ell}\right)^{i-1} \leq d^{i-1}\left(\frac{d}{\sigma_\ell}\right)^{i-1} = \frac{d^{2i-2}}{\sigma_\ell^{i-1}},$$

and, therefore,

$$\sum_{i=2}^{d}\left(\frac{\sigma_\ell}{d^2}\right)^{i-2}\left|\frac{f^{(i)}(z_\ell)}{f'(z_\ell)}\right| \leq \sum_{i=2}^{d}\left(\frac{\sigma_\ell}{d^2}\right)^{i-2}\frac{d^{2i-2}}{\sigma_\ell^{i-1}} = \sum_{i=2}^{d}\frac{d^2}{\sigma_\ell} = (d-1)\frac{d^2}{\sigma_\ell}.$$

Plugging in into the overall sum yields

$$\sum_{\ell=1}^{m}\log(C_{z_\ell})^{-1} = \sum_{\ell=1}^{m}\log\left(8\cdot\left(\frac{d^2}{\sigma_\ell}+(d-1)\frac{d^2}{\sigma_\ell}\right)\right) = 3d+\sum_{\ell=1}^{m}\log\frac{d^3}{\sigma_\ell}$$

$$= \quad 3d+3m\log d+\Sigma_f+\sum_{\ell=m+1}^{d}\log\sigma_\ell \leq 3d+3d\log d+\Sigma_f+d(\Gamma+1) = O(d(\Gamma+\log d)+\Sigma_f) \quad \square$$

Lemma 20. *The linear sequences for all real roots are computed within a total bit complexity of*

$$\tilde{O}(d(d\Gamma+\Sigma_f)(\tau+d\Gamma+\Sigma_f)$$

Proof. The total cost of all linear sequences is bounded by

$$\tilde{O}(\sum_{i=1}^{m}(\Gamma+\log(C_{z_i}^{-1}))d(\log(C_{z_i}^{-1})+\tau+d\Gamma+\Sigma_f)).$$

By rearranging terms, we obtain

$$=\tilde{O}(d^2\Gamma(\tau+d\Gamma+\Sigma_f)+d(\tau+d\Gamma+\Sigma_f)\sum\log(C_{z_i}^{-1})+d(\sum\log(C_{z_i}^{-1}))^2)$$

which equals $\tilde{O}(d(d\Gamma+\Sigma_f)(\tau+d\Gamma+\Sigma_f))$ with Lemma 19. □

Cost of the quadratic sequence. Let us fix some root ξ of f. Its quadratic sequence consists of at most $1+\log L$ steps, because N is squared in every step (except for at most one failing step) and the sequence stops as soon as the interval is smaller than 2^{-L}. Since we ignore logarithmic factors, it is enough to bound the costs of one QIR step in the sequence. Clearly, since the interval is not smaller than 2^{-L} in such a step, we have that $\log(b-a)^{-1}\leq L$. Therefore, the required precision is bounded by $O(L+\tau+d\Gamma+\Sigma_f)$. It follows that an AQIR step performs up to $\tilde{O}(d(L+\tau+d\Gamma+\Sigma_f))$ bit operations.

Lemma 21. *The quadratic sequences for one real root is computed within a bit complexity of*

$$\tilde{O}(d(L+\tau+d\Gamma+\Sigma_f)).$$

Total cost. We have everything together to prove the main result:

Theorem 22. *Algorithm 5 performs root refinement within*

$$\tilde{O}(d(d\Gamma_f+\Sigma_f)^2+dL)$$

bit operations for a single real root[3] *of f, and within*

$$\tilde{O}(d(d\Gamma_f+\Sigma_f)^2+d^2L)$$

[3]In its initial formulation, Algorithm 5 assumes that isolating intervals for *all* real roots are given. If only one isolating interval I_k for a root z_k is given, we have to normalize I_k first and, then, compute the signs of f at the endpoints of I.

for all real roots. The coefficients of f need to be approximated to $\tilde{O}(L+d\Gamma_f+\Sigma_f)$ bits after the binary point. For a polynomial with integer coefficients, the above bounds simplify to $\tilde{O}(d^3\tau^2+dL)$ and $\tilde{O}(d^3\tau^2+d^2L)$, respectively.

Proof. We first restrict to the case where $1 \leq |a_d| < 2$. The so far achieved complexity bounds are formulated in terms of an arbitrary (but given) upper bound $\Gamma \in \mathbb{N}$ on Γ_f. In [39, Section 6.1], it is shown how to compute a Γ with $\Gamma_f \leq \Gamma < \Gamma_f + 4\log d$ using $\tilde{O}((d\Gamma_f)^2)$ bit operations and approximations of f to $\tilde{O}(d\Gamma_f)$ bits after the binary point. Furthermore, the latter construction also shows that $\tau = \lceil \log(\max_i |a_i|) \rceil = O(d\Gamma)$ if $1 \leq |a_d| < 2$. By Lemma 10, the normalization for all isolating intervals requires $\tilde{O}(d(d\Gamma+\Sigma_f)(\tau+d\Gamma+\Sigma_f))$ bit operations. The linear subsequences of the AQIR sequence are computed in the same time by Lemma 20. The quadratic subsequences are computed with $\tilde{O}(d^2L+d^2\tau+d^3\Gamma+d^2\Sigma_f)$ bit operations by Lemma 21; the latter three terms are all dominated by $\tilde{O}(d(d\Gamma+\Sigma_f)(\tau+d\Gamma+\Sigma_f))$. Hence, with $\Gamma = O(\Gamma_f + \log d)$ as above and $\tau = \tilde{O}(d\Gamma_f)$, the claimed bound on the bit complexity to refine all roots follows. The maximal number of required bits follows from Lemma 7 because the maximal required precision in any AQIR step is bounded by $O(L+\tau+d\Gamma+\Sigma_f) = \tilde{O}(L+d\Gamma_f+\Sigma_f)$. The bound on refining a single root follows easily when considering the cost of the quadratic sequence for this root only.

For the more general case, where $1 \leq |a_d| < 2$ is not necessarily given, we first shift the coefficients by $s = \lfloor \log |a_d| \rfloor$ bits such that we can apply the above result to the shifted polynomial. Since this coefficient shift does not change the roots, our bit complexity bound follows immediately. For the required precision, we need $\tilde{O}(L+d\Gamma_f+\Sigma_f)-s$ since we need an approximation of the shifted polynomial to $\tilde{O}(L+d\Gamma_f+\Sigma_f)$ bits after the binary point.

If f has integer coefficients of absolute value less than 2^τ, then $\Gamma_f \leq \tau+1$ according to Cauchy's root bound. Furthermore, we have $\Sigma_f = O(d(\log d+\tau))$; e.g. see [39, Appendix A.2] for a proof. Hence, the complexity bounds for integer polynomials follow. □

7. Experimental Results

We compare the asymptotic bounds of EQIR and AQIR and their practical behavior for increasing input sizes in the case of integer polynomials. We have implemented both algorithms exactly as described in this paper (without the tech-

niques presented in the forthcoming Section 8), in the context of the CGAL[4] library, written in C++. We used GMP, version 5.0.4, for integer and rational arithmetic. We generated integer polynomials of various types (described below) using the Maple routine *randpoly*, isolated their real roots using Descartes method, and measured the time to refine them to a predefined refinement precision on a laptop with dual Pentium core clocked at 2.4 GHz with 3MB cache size each, and a total RAM of 4 GB, running Debian squeeze. Both the source code and the benchmark instances can be sent on request.

In the first run, we chose polynomials with 20-bit-coefficients chosen uniformly at random and a degree between 50 and 1600. The refinement quality was set to 10000 bits after the binary point. Table 1 (top) lists the results. We also generated two bivariate dense polynomials, each with randomly chosen 10-bit coefficients and total degrees between 5 and 40, and computed the resultant of them. The results are listed in Table 1 (bottom).

First of all, the quotient between the running times for EQIR and AQIR is proportional to d which matches the asymptotic bound proved in this paper. Moreover, both in the exact and approximate version, only a small number of bisections are performed during the refinement. That means that quadratic convergence takes place almost immediately. The normalization phase (which only exists in the approximate version) also performs just a small number of bisections. This implies that the normalization phase and the linear sequence have a minor impact on the practical running time of the algorithm, and that the cost is dominated by the quadratic sequence. Recall that a single root can be refined in $\tilde{O}(d^3\tau^2 + dL)$ with AQIR, where the first term is caused by the normalization and the linear sequence, the second by the quadratic sequence. Indeed, the running time per root increases linearly in d for the approximate variant, as suggested by the second term of the complexity bound. For EQIR, the complexity is $\tilde{O}(d^4\tau^2 + d^2L)$ for a single root, with the second term accounting for the quadratic sequence. We can observe that the running time per root grows quadratically with d. Note that also in the second table, the running times of both QIR versions are only moderately worse despite the coefficient growth of the input instances.

We investigate the dependance on the refinement precision L by fixing a degree of 100 and a coefficient bitsize of 20, and let the final precision L grow from 2000 to 128000 (Table 2). As we can observe, the quotient of the running times of both refinement variants stabilizes for high values of L. However, the growth

[4]Computational Geometry Algorithms Library, `www.cgal.org`

d	EQIR		AQIR			
	$\frac{\#\text{ bis.}}{\#\text{ roots}}$	$\frac{\text{time}}{\#\text{ roots}}$	$\frac{\#\text{ bis-norm}}{\#\text{ roots}}$	$\frac{\#\text{ bis-refine}}{\#\text{ roots}}$	$\frac{\text{time}}{\#\text{ roots}}$	$\frac{t_{\text{exact}}}{t_{\text{approx}}}$
50	2.5	0.438	1.5	3	7.12	0.0615
100	1.5	1.63	1	2.5	14.2	0.115
200	3.5	6.40	3	3	30.7	0.209
400	3.6	24.2	2.4	2.3	60.0	0.403
800	3	97.6	2	1.3	124	0.790
1600	4.3	392	2.3	2.3	249	1.58

(d_1,d_2)	τ	EQIR		AQIR			
		$\frac{\#\text{ bis.}}{\#\text{ roots}}$	$\frac{\text{time}}{\#\text{ roots}}$	$\frac{\#\text{ bis-norm}}{\#\text{ roots}}$	$\frac{\#\text{ bis-refine}}{\#\text{ roots}}$	$\frac{\text{time}}{\#\text{ roots}}$	$\frac{t_{\text{exact}}}{t_{\text{approx}}}$
(10,5)	161	1	0.445	1	0.5	7.18	0.0620
(10,10)	226	3.3	1.67	1.8	2.5	14.6	0.114
(20,10)	353	2.2	6.38	1.8	2.5	30.1	0.212
(20,20)	487	1.8	25.2	2.2	1.7	60.1	0.414
(40,20)	755	2.9	104	1.8	2.1	127	0.813
(40,40)	1042	3.6	426	1.8	1.9	274	1.556

Table 1: Experimental results for polynomials with random 20-bit coefficients (first table) and for resultants of bivariate polynomials with random 10-bit coefficients (second table). For the latter, the degree is $d_1 \cdot d_2$, and the maximal coefficient bitsize is displayed in the second column. In all cases, the final precision L is set to 10000. For each degree, we generated 5 instances and measured the time of root refinement for EQIR and AQIR. The displayed numbers refer to the instance whose quotient of running times (last column) is the median among the 5 instances. The other columns display (from left to right) the number of bisections the EQIR method performs internally per root, the refinement time of EQIR per root, the number of bisections in the normalization of AQIR per root, the number of bisections AQIR performs per root after normalizing, the refinement time of AQIR per root, and the ratio of the total running times of EQIR and AQIR.

L	$\frac{t_{\text{exact}}}{\#\text{ roots}}$	$\frac{t_{\text{approx}}}{\#\text{ roots}}$	$\frac{t_{\text{exact}}}{t_{\text{approx}}}$
2000	0.0817	1.68	0.0486
4000	0.220	2.89	0.0760
8000	0.605	6.06	0.100
16000	1.60	13.9	0.115
32000	4.36	35.6	0.122
64000	11.6	94.6	0.122
128000	28.9	242	0.120

Table 2: Experimental results for polynomials with random 20-bit coefficients of degree 100. Again, the table lists the median over 5 independent instances.

factor of the running time is not linear in L; we observe that the running time roughly increases by a factor of about 2.6 when L doubles, which corresponds to a growth of roughly $L^{1.4}$. To explain this super-linear behavior, we remark that our analysis ignored logarithmic factors in L; at least one such factor is included from fast integer arithmetic. Also, GMP does only switch to asymptotically fast arithmetic for very large integers and uses asymptotically inferior methods for smaller instances.

Finally, we investigate the case of growing coefficient sizes. For that, we fix a degree of 100 and a final precision of 10000 bits and vary the coefficient size. We see in Table 3 that the running time grows very moderately for increasing bitsizes. Also, AQIR handles large coefficients worse than EQIR (we also have tested a polynomial with 128000-bit coefficients where the ratio drops to about 0.01). Recall that our implemented version of AQIR uses absolute precision arithmetic and therefore does not round the coefficients during the computation. Consequently, it suffers from high coefficient sizes in every step where it uses interval arithmetic. An improved version of AQIR using relative precision would remove this drawback.

To summarize our experiments, the cost of the quadratic sequence dominates the refinement process, and the cost of this sequence is proportional to dL^α for AQIR and proportional to d^2L^α for EQIR in practice, with $\alpha \approx 1.4$. It shows that the approximate version is not just a theoretical trick to reduce the complexity, but has a practical impact.

On the possible disagreement that AQIR is faster than EQIR only for quite large values of d and L, we reply that our version of AQIR is rather designed for a simple

τ	$\frac{t_{\text{exact}}}{\#\text{ roots}}$	$\frac{t_{\text{approx}}}{\#\text{ roots}}$	$\frac{t_{\text{exact}}}{t_{\text{approx}}}$
40	1.63	14.2	0.115
80	1.64	14.1	0.116
160	1.66	14.5	0.114
320	1.68	14.9	0.112
640	1.75	15.5	0.111
1280	1.82	16.7	0.109
2560	1.96	19.4	0.101
5120	2.11	24.7	0.085

Table 3: Experimental results for polynomials of degree 100, and a refinement precision of 10000 bits, with coefficients chosen uniformly at random. Again, the table lists the median over 5 independent instances.

complexity analysis than for a fast implementation. Some optimizations include to use relative instead of absolute precision, to leave out the additional subdivision points at $m^* \pm \frac{7}{8}\omega$ (which are formally needed for quadratic convergence, but should be insignificant in practice), and to choose the internal working precision more adaptively (instead of always setting $\rho \leftarrow 2$ before each while-loop). We believe that such improvements lead to an implementation which shows its strength for much smaller instances.

Finally, we remark that the recently introduced CGAL-package on algebraic computations [4] represents algebraic numbers by their isolating intervals and uses QIR to refine them. The implemented version therein can be considered as a "light version" of the techniques presented in this paper, using relative approximations to speed up polynomial evaluations, but falling back to exact methods in the case of failure. The results of our experimental evaluations motivate an integration of a fully approximate variant (that is, AQIR with the described optimizations) into CGAL.

8. Fast multipoint evaluation

It is well known that, roughly speaking, evaluating a univariate polynomial of degree d in $O(d)$ positions simultanuously has the same arithmetic complexity as evaluating it at a single position, up to logarithmic factors [41, Corollary 10.8]. These techniques are called *fast multipoint evaluation*; it suggests itself to apply them on our AQIR algorithm since polynomial evaluation is the dominant

operation. However, since all our evaluations are only approximate with a fixed working precision, we need an approximate variant of fast multipoint evaluation. The following result already appears in [22], and has been restated in [24] in a slightly more general form, which we use in our algorithm:

Theorem 23. *[24, Thm. 10] Let $F \in \mathbb{C}[x]$ be a polynomial of degree d with $\|F\|_1 \leq 2^\tau$, with $\tau \geq 1$, and let $x_1, \ldots, x_d \in \mathbb{C}$ be complex points with absolute values bounded by 2^Γ, where $\Gamma \geq 1$. Then, approximate multipoint evaluation up to a precision of 2^{-L} for some integer $L \geq 0$, that is, computing $\tilde{y}_j$ such that $|\tilde{y}_j - F(x_j)| \leq 2^{-L}$ for all j, is possible with*

$$\tilde{O}(d(L + \tau + d\Gamma))$$

bit operations. The precision demand on F and the points x_j is bounded by $L + O(\tau + d\Gamma + d \log d)$ bits after the binary point.

Note that, with the notations of the theorem, $[\tilde{y}_j - 2^L, \tilde{y}_j + 2^L]$ is guaranteed to contain $F(x_j)$; therefore, the theorem gives an alternative to interval arithmetic with bounded precision. Specifically, we can replace the usage of interval arithmetic in line 7 of Algorithm 2 and in lines 5 and 13 of Algorithm 3 by the multipoint evaluation algorithm in [24] (for now, just applied at a single point). The precision quality is adaptively increased during the execution of the while loop, and we can prove the same asymptotic bounds (up to an additional term $O(d \log d)$) on the maximal precision as in Lemmas 5 and 7.

Of course, we want to exploit that Theorem 23 bounds the cost of evaluating a polynomial at multiple points. For that goal, we adapt our root refinement algorithm as follows: think of multipoint evaluation as a virtual machine with d input slots and d output slots which returns $\tilde{y}_1, \ldots, \tilde{y}_d$ for input $x_1, \ldots, x_d$ as described in Theorem 23. The idea is to perform the refinement of all real roots simultaneously and to use that machine whenever a polynomial has to be evaluated. To be a bit more precise, reconsider Algorithm 5. We leave the normalization subprocedure unchanged (we could use multipoint evaluation here as well, but it would not change the complexity). Instead of the for-loop, we initialize an integer P to 1, find all isolating intervals of length at least 2^{-P} and call a modified version of AQIR for them that we describe below; if all intervals are smaller, we double P and repeat. That means that intervals which are comparably very small are not further refined until the other intervals are roughly of the same size.

The modification of AQIR are as follows: we apply Algorithm 3 to all isolating intervals and divide them into two groups: those for which $N = 2$ (that is, an

approximate bisection is performed) and those for which $N > 2$. For the first group, we execute the while loop of Algorithm 2 simultanuously for all intervals; it makes sense to think about this as a parallel process with execution branches – we can easily simulate parallelism by a sequential algorithm that cycles through the different branches. Every branch fills one input slot of the virtual machine and then waits for the other branches to fill their slots, (or send a signal that they have left the loop already). Once all slots are filled, the machine starts the evaluation and all branches continue their execution until the next loop iteration requires an evaluation with increased precision. This process continues until all branches have left the loop. For the group of intervals that perform an AQIR step with $N > 2$, the same strategy is used.

Regarding the complexity of the described method; note that all computations except for the calls of the virtual machine are negligible.[5] Moreover, for a fixed value of P (as defined above), every interval of length at least 2^{-P} is refined by at least one half per iteration (in an amortized sense). It follows that there are at most $O(P)$ iterations of the modified AQIR procedure, and afterwards, all intervals are of size at most 2^{-P}. On the other hand, if all intervals have entered the quadratic sequence, the virtual machine spends at most $O(\log P)$ iterations before doubling P because there is at most one failing QIR call per isolating interval.

We analyze the complexity similar to Section 6.2: Set $C := \max_\xi C_\xi$, where the maximum is taken over all real roots ξ of f and C_ξ is defined as in Definition 12. Let P_0 be the smallest power of two that is larger than 2^{-C}. We bound the complexity to refine all intervals to size 2^{-P_0} or less: As we said above, we need $O(P_0)$ calls of the multipoint version of AQIR for that. Each call, in turn, requires at most

$$\tilde{O}(d(P_0 + \tau + d\Gamma + \Sigma_f))$$

bit operations (compare Lemma 7 and Theorem 23). Since $P_0 \leq 2C \leq 2\sum_\xi C_\xi = O(d(\Gamma + \log d) + \Sigma_f))$ (Lemma 19), the cost of refining all intervals to size less than 2^{-P_0} is bounded by

$$\tilde{O}(d(d\Gamma + \Sigma_f)(\tau + d\Gamma + \Sigma_f)$$

[5] We remark that, for each AQIR step, we also have to compute an approximation of the fraction of the values $f(a)$ and $f(b) - f(a)$, provided that sufficiently good approximations of $f(a)$ and $f(b)$ are already computed. The cost for the computation of one fraction is then bounded by $\tilde{O}(n\Gamma + \tau + \rho)$, where ρ denotes the required output precision. Hence, when processing up to n intervals in parallel, the total cost is bounded by $\tilde{O}(n(n\Gamma + \tau + \rho))$ bit operations which matches the complexity for one call of the virtual machine with output precision ρ.

with the same argumentation as in Lemma 20.

The benefit of multipoint evaluation takes effect in the second part of the complexity analysis: suppose that all intervals have entered the quadratic sequence, then, as mentioned above, there are at most $O(\log P)$ calls per P, and there are only $\log L$ different P-values reached during the refinement. It follows that (up to logarithmic factors) the cost is determined by a single execution of the multipoint version of AQIR which is

$$\tilde{O}(d(L+\tau+d\Gamma+\Sigma_f)).$$

Notice that this matches the previous cost of the quadratic sequence for a single root. Putting everything together, we can prove in analogy to Theorem 22 that the multipoint evaluation variant of AQIR needs

$$\tilde{O}(d(d\Gamma+\Sigma_f)^2+dL)$$

bit operations to refine *all* isolating intervals to a width of at most 2^{-L}.

In summary, we obtain the following result:

Theorem 24. *When using fast approximate multipoint evaluation, Algorithm 5 performs root refinement within*

$$\tilde{O}(d(d\Gamma_f+\Sigma_f)^2+dL)$$

bit operations for all real roots of f. The coefficients of f need to be approximated to $\tilde{O}(L+d\Gamma_f+\Sigma_f)$ bits after the binary point. For a polynomial with integer coefficients of absolute value less than 2^τ, the above bound on the bit complexity simplifies to $\tilde{O}(n^3\tau^2)$.

9. Concluding Remarks

We have presented a complete solution to the root refinement problem using validated numerical methods in this paper. Despite the relative simplicity of the approach, we obtain a bit complexity that is essentially competitive to the best known bounds as achieved by much more sophisticated algorithms. Moreover, we have demonstrated that our approach is easily implementable and leads to practical improvements even when implemented in the most naive form.

We have shown that the complexity of approximating roots of a real polynomial only depends on the geometry of the roots and not on the complexity or the

type of the coefficients. For instance, we used this fact in [21] to derive considerably improved complexity bounds for the topology computation of algebraic plane curves.

Although the focus of this work was the asymptotic complexity, the presented algorithm also aims for a practically efficient solution of the root approximation problem. Indeed, a simplified version of our approach (for integer coefficients) is included in the recently introduced CGAL-package on algebraic computations [4]. Experimental comparisons in the context of [3] have shown that the approximate version of QIR gives significantly better running times than its exact counterpart. These observations underline the practical relevance of our approximate version and suggest a practical comparison with state-of-the-art solvers as further work.

Acknowledgements. We would like to thank the anonymous reviewers for carefully reading preliminary manuscripts and for their suggestions that helped us to considerably improve the overall presentation. The first author acknowledges support by the Max Planck Center for Visual Computing and Communication.

References

[1] J. Abbott. Quadratic Interval Refinement for Real Roots. arXiv:1203.1227, 2012. Originally presented as a "Poster" at the *International Symposium on Symbolic and Algebraic Computation (ISSAC) 2006.*

[2] E. Berberich, P. Emeliyanenko, A. Kobel, and M. Sagraloff. Exact symbolic-numeric computation of planar algebraic curves. *Theoretical Computer Science*, 491:1–32, 2013.

[3] E. Berberich, P. Emeliyanenko, and M. Sagraloff. An elimination method for solving bivariate polynomial systems: Eliminating the usual drawbacks. In *Workshop on Algorithm Engineering & Experiments (ALENEX)*, pages 35–47, 2011.

[4] E. Berberich, M. Hemmer, and M. Kerber. A generic algebraic kernel for non-linear geometric applications. In *Proceedings of the Symposium on Computational Geometry (SoCG'11)*, pages 179–186, 2011.

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

[6] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer New York, 1998.

[7] J. Bus and T.J.Dekker. Two efficient algorithms with guaranteed convergence for finding a zero of a function. *ACM Transactions on Mathematical Software*, 1(4):330–345, 1975.

[8] J. Cheng, S. Lazard, L. Pearanda, M. Pouget, F. Rouillier, and E. Tsigaridas. On the topology of real algebraic plane curves. *Mathematics in Computer Science*, 4:113–137, 2010.

[9] G. Collins and R. Loos. Real zeros of polynomials. In *Computer Algebra*, volume 4 of *Computing Supplementum*, pages 83–94. Springer, 1982.

[10] G. E. Collins and A. G. Akritas. Polynomial Real Root Isolation Using Descartes' Rule of Signs. In *Proceedings of the Symposium on Symbolic and Algebraic Computation (SYMSAC)*, pages 272–275, 1976.

[11] F. Cucker, P. Koiran, and S. Smale. A polynomial time algorithm for diophantine equations in one variable. *Journal of Symbolic Computation*, 27(1):21 – 29, 1999.

[12] L. Dai and B. Xia. logcf: An efficient tool for real root isolation. *CoRR*, abs/1209.3555, 2012.

[13] Z. Du, V. Sharma, and C. Yap. Amortized bound for root isolation via Sturm sequences. In *Symbolic-Numeric Computation*, Trends in Mathematics, pages 113–129. Birkhäuser Basel, 2007.

[14] A. Eigenwillig. On multiple roots in Descartes' rule and their distance to roots of higher derivatives. *Journal of Computational and Applied Mathematics*, 200(1):226–230, March 2007.

[15] A. Eigenwillig, M. Kerber, and N. Wolpert. Fast and exact geometric analysis of real algebraic plane curves. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 151–158, 2007.

[16] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. A Descartes algorithm for polynomials with bit-stream coefficients. In *International Workshop on Computer Algebra in Scientific Computing (CASC)*, volume 3718 of *LNCS*, pages 138–149, 2005.

[17] I. Z. Emiris, A. Galligo, and E. P. Tsigaridas. Random polynomials and expected complexity of bisection methods for real solving. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 235–242, 2010.

[18] I. Z. Emiris, V. Y. Pan, and E. P. Tsigaridas. Algebraic algorithms. In *Computing Handbook, 3rd ed. (1)*, pages 10: 1–30. 2014.

[19] M. Kerber. On the complexity of reliable root approximation. In *International Workshop on Computer Algebra in Scientific Computing (CASC)*, volume 5743 of *LNCS*, pages 155–167, 2009.

[20] M. Kerber and M. Sagraloff. Efficient real root approximation. In *Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation (ISSAC 2011)*, pages 209–216, 2011.

[21] M. Kerber and M. Sagraloff. A worst-case bound for topology computation of algebraic curves. *Journal of Symbolic Computation*, 47(3):239–258, 2012.

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

[23] A. Kobel. Certified Numerical Root Finding. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 2011.

[24] A. Kobel and M. Sagraloff. Fast approximate polynomial multipoint evaluation and applications. arXiv:1304.8069, 2013.

[25] A. Kobel and M. Sagraloff. On the complexity of computing with planar algebraic curves. *Journal of Complexity*, 2014. To appear. See *http://arxiv.org/abs/1401.5690* for an online version.

[26] J. M. McNamee and V. Y. Pan. Efficient polynomial root-refiners: A survey and new record efficiency estimates. *Computers & Mathematics with Applications*, 63(1):239–254, 2012.

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

[28] K. Mehlhorn, R. Osbild, and M. Sagraloff. A general approach to the analysis of controlled perturbation algorithms. *Computational Geometry*, 44(9):507–528, 2011.

[29] K. Mehlhorn, M. Sagraloff, and P. Wang. From approximate factorization to root isolation. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 283–290, 2013.

[30] R. Moore, R. Kearfott, and M. Cloud. *Introduction to Interval Analysis*. SIAM e-books. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[31] V. Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers and Mathematics with Applications*, 31(12):97–138, 1996.

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

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

[34] V. Y. Pan. Root-refining for a polynomial equation. In *International Workshop on Computer Algebra in Scientific Computing (CASC)*, volume 7442 of *LNCS*, pages 283–293. 2012.

[35] V. Y. Pan and E. Tsigaridas. On the boolean complexity of real root refinement. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 299–306, 2013.

[36] V. Y. Pan and E. Tsigaridas. Nearly optimal refinement of real roots of a univariate polynomial. *HAL: hal-00960896*, 2014. see *http://hal.inria.fr/hal-00960896* for an online version.

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

[38] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2004.

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

[40] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity, 1982; updated 2004. Manuscript, Department of Mathematics, University of Tübingen. See *www.informatik.uni-bonn.de/ schoe/fdthmrep.ps.gz* for an online version.

[41] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999. 1st Edition.