HAL open science

On the Boolean complexity of real root refinement

Victor Y. Pan, Elias Tsigaridas

To cite this version:

Victor Y. Pan, Elias Tsigaridas. On the Boolean complexity of real root refinement. ISSAC, Jun 2013, Boston, United States. hal-00816214v1

HAL Id: hal-00816214

https://inria.hal.science/hal-00816214v1

Submitted on 20 Apr 2013 (v1), last revised 25 Apr 2013 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On the Boolean complexity of real root refinement

Victor Y. Pan
Depts. of Mathematics and Computer Science
Lehman College and Graduate Center
of the City University of New York
Bronx, NY 10468 USA
victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

Elias P. Tsigaridas
INRIA, Paris-Rocquencourt Center, PolSys Project
UPMC, Univ Paris 06, LIP6
CNRS, UMR 7606, LIP6
Paris, France
elias.tsigaridas@inria.fr

ABSTRACT

We assume that a real square-free polynomial A has a degree d, a maximum coefficient bitsize τ and a real root lying in an isolating interval and having no nonreal roots nearby (we quantify this assumption). Then, we combine the *Double Exponential Sieve* algorithm (also called the *Bisection of the Exponents*), the bisection, and Newton iteration to decrease the width of this inclusion interval by a factor of $t = 2^{-L}$. The algorithm has Boolean complexity $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$. Our algorithms support the same complexity bound for the refinement of r roots, for any $r \leq d$.

Categories and Subject Descriptors: F.2 [Theory of Computation]: Analysis of Algorithms and Problem Complexity; I.1 [Computing Methodology]: Symbolic and algebraic manipulation: Algorithms

Keywords real root refinement, polynomial, real root problem, Boolean complexity

General Terms Algorithms, Experimentation, Theory

1. INTRODUCTION

Given a polynomial A, of degree d and maximum coefficient bitsize τ, and an interval with rational endpoints that contains one of its real roots (isolating interval), we devise an algorithm that refines this *inclusion interval* to decrease its width by a factor $t = 2^{-L}$, for some positive integer L.

The problem of real root refinement appears very often as an important ingredient of various algorithms in computer algebra and nonlinear computational geometry, for example in algorithms for computing the topology of real plane algebraic curves [11, 22], solving systems of polynomial equations [24, 14, 17], isolating the real roots of polynomials with coefficients in an extension field [37, 20], cylindrical algebraic decomposition [2, 13], and many others.

For the complexity of approximating (all) the roots of a polynomial we refer the reader to [29], see also [27], [25, Chapter 15] where a Boolean complexity of $\widetilde{\mathcal{O}}_B(d^3 + d^2L)$ is proved, which is "within polylogarithmic factors from the optimum", provided $\tau = \mathcal{O}(L)$.

The problem of refinement is also an important ingredient for algorithms that tackle the real root problem, that is the problem of isolating and approximating the real roots of a polynomial that has only real roots. In this context we refer to the work of Ben-Or and Tiwari [3] that introduced interlacing polynomials and *Double Exponential Sieve*. Pan and Linzer [30] and Bini and Pan [8] in a sequence of works, see also [6, 7], modified the approach of [3] (they called it *Bisection of the Exponents*) to approximate the eigenvalues of a real symmetric tridiagonal matrix by using Courant-Fischer minimax characterization theorem. In [31] a variant of the refinement algorithms in [30, 8] is used, for approximating all the real roots of a polynomial.

Collins and Krandick [12] presented a variant of Newton's algorithm where all the evaluations involve only dyadic numbers, as well as a comparison with the case where operations are performed with rationals of arbitrary size. Quadratic convergence of Newton's iterations is guaranteed by point estimates and α-theory of Smale, e.g. [9]. For robust approximation of zeros based on bigfloats operations we refer to [36]. A very interesting and efficient algorithm that combines bisection and Newton iterations is the *Quadratic Interval Refinement* (QIR) by Abbott [1]. For a detailed analysis of the Boolean complexity of QIR we refer the reader to [21]. Kerber and Sagraloff [22] modify QIR to use interval arithmetic and approximations and they achieve a bound of $\widetilde{\mathcal{O}}_B(d^3\tau^2 + dL)$. A factor of τ could be saved if we use fast algorithms for root isolation of univariate polynomials, e.g. [34], [29, 35]. We should also mention [33] that is based on Kantorovich point estimates which is efficient in practice but of unknown complexity.

We revisit the approach of [3], [30, 8] to devise our *Real Root Refinement* ($\mathtt{R}_3$) algorithm and present a detailed analysis in the bit complexity model, based on exact operations with rationals (Thm. 12). We also introduce an approximate variant ($\alpha\mathtt{R}_3$) based on interval arithmetic, Sec. 2.1, where we use multi-precision floating point numbers for computations and to represent the endpoints of intervals, and we estimate in advance the maximum precision needed. For this we use tools from Kerber and Sagraloff [22] for evaluating a polynomial at a rational number using interval arithmetic. We also study the Newton operator both from an exact and approximate point of view (Sec. 2.3). We provide Boolean complexity bounds for approximate variants of Double Exponential Sieve (Lem. 4) and Newton iteration (Lem. 10).

The Boolean complexity of $\mathtt{R}_3$ and $\alpha\mathtt{R}_3$ is $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$5.00.

(Theorem 12). The same algorithms support the bounds $\widetilde{\mathcal{O}}_B(d^2\tau + d^2 L)$ and $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$, respectively, for the refinement of up to d roots (Section 2.4). We assume that there is no complex root of the polynomial in the complex disc that has the isolating interval as diameter. Such an interval could be the outcome of root-finding algorithms. We detail this in Section 2.5.

The rest of the paper is structured as follows. First we introduce our notation. Section 2 presents a high level description of the real root refinement algorithm. We detail its three steps, in Sec. 2.1, Sec. 2.2, and Sec. 2.3. Section 3 estimates the expected number of steps of DES and αDES when the input polynomial is random of type Weyl, Sec. 3.1, or $SO(2)$, Sec. 3.2. Finally, in Section 4 we conclude and suggest directions for furhter study.

Notation. In what follows $\mathcal{O}_B$, resp. $\mathcal{O}$, means bit, resp. arithmetic, complexity and the $\widetilde{\mathcal{O}}_B$, resp. $\widetilde{\mathcal{O}}$, notation means that we are ignoring logarithmic factors. For a polynomial $A = \sum_{i=0}^{d} a_i\, x^i \in \mathbb{Z}[x]$, $\mathtt{deg}(A) = d$ denotes its degree and $\mathcal{L}(A) = \tau$ the maximum bitsize of its coefficients, including a bit for the sign. For $a \in \mathbb{Q}$, $\mathcal{L}(a) \geq 1$ is the maximum bitsize of the numerator and the denominator. $\mathsf{M}(\tau)$ denotes the bit complexity of multiplying two integers of size τ; we have $\mathsf{M}(\tau) = \widetilde{\mathcal{O}}_B(\tau)$. 2^{Γ} is an upper bound on the magnitude of the roots of A. We write $\Delta_\alpha(A)$ or just Δ_α to denote the minimum distance between a root α of a polynomial A and any other root, we call this quantity *local separation bound*. If we are considering the i-th root, α_i, then we also write Δ_i instead of Δ_{α_i}. $\Delta(A) = \min_\alpha \Delta_\alpha(A)$ or just Δ denotes the *separation bound*, that is the minimum distance between all the roots of A. The Mahler bound (or measure) of A is $\mathcal{M}(A) = a_d \prod_{|\alpha|\geq 1} |\alpha|$, where α runs through the complex roots of A, e.g. [26, 39]. If $A \in \mathbb{Z}[x]$ and $\mathcal{L}(A) = \tau$, then $\mathcal{M}(A) \leq \|A\|_2 \leq \sqrt{d+1}\|A\|_\infty = 2^\tau\sqrt{d+1}$. If we evaluate a function F (e.g. $F = A$) at a number c using interval arithmetic, then we denote the resulting interval by $[F(c)]$, provided that we fix the evaluation algorithm and the precision of computing. Let $D(c, r) = \{x : |x - c| \leq r\}$.

2. THE $\mathtt{R}_3$ AND $\alpha\mathtt{R}_3$ ALGORITHMS

In what follows $A = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x]$ with $\mathcal{L}(A) = \tau$. Let α_1 be the real root of A that lies in an (isolating) interval $I = [a\,..\,b]$. The width of I, $w = (b-a)/2$, has bitsize $\mathcal{L}(w) = \mathcal{O}(\lg \Delta(A)) = \mathcal{O}(d\tau)$, in the worst case (see also Prop. 1). We write $|I| = b - a$ and $m = (a+b)/2$. We wish to refine I to include α_1 into a subinterval of the width $t = 2^{-L}w$. We define the isolation ratio of a real isolating interval I of a root α of A as $\mathtt{ir}(I) = 2\,|m - \alpha_c|/|I|$, where α_c is the root of A that is the closest to α.

The high-level description that we present follows [3] and [30]. For details and various improvements we refer the reader to [30, 6, 8, 31]. The algorithm for refining the isolating interval of a real root α_1 consists of three steps: Double Exponential Sieve (DES), Bisection (BIS) and Newton iteration (NEWTON). We denote the approximate variants by αDES, αBIS, and αNEWTON, respectively. The three procedures are as follows:

(1) DES or αDES achieves an isolation ratio at least 3.

(2) Sufficiently many bisections (BIS or αBIS), but $\widetilde{\mathcal{O}}(\lg(d))$, increase the isolation ratio.

(3) Then Newton iteration, NEWTON or αNEWTON, converges quadratically and yields an inclusion interval of the desired width.

In the sequel we describe in detail and analyze the complexity of the three sub-algorithms.

The following proposition estimates the separation bounds for a univariate (integer) polynomial. For variants and proof techniques we refer the reader to e.g. [13, 15, 26, 20]. We use a variant from [23, Thm. 4].

Proposition 1. *Suppose $A \in \mathbb{Z}[x]$ is a square-free univariate polynomial of degree d, $\mathcal{L}(A) = \tau$, and $\alpha_1, \ldots, \alpha_d$, are the d distinct roots of A. Let $\mathbf{SR}_0(A, A')$ be the resultant of A and its derivative, A', w.r.t. x, and $\Delta_i = \Delta_i(A) = \Delta_{\alpha_i}(A)$, where $i = 1, \ldots, d$. Then it holds*

$$|\alpha_i| \leq 2^{\Gamma} \leq 2\|A\|_\infty \leq 2^{\tau+1} \quad , \tag{1}$$

$$-\sum_i \lg \Delta_i(A) \leq 30\, d \lg \mathcal{M}(A) + 3\,\mathcal{L}\left(\mathbf{SR}_o(A, A')\right) \tag{2}$$
$$\leq 36\, d\,\tau + 42\, d \lg d \quad .$$

Lemma 2. *[22] Suppose we evaluate A at c, where $|c| \leq 2^{\Gamma+2}$, and suppose we use a working precision (or fixed precision arithmetic) ρ. Then*

$$\mathtt{width}[A(c)] \leq 2^{-\rho+1}(d+1)^2 2^{\tau+d(\Gamma+2)} \quad .$$

The following lemma generalizes [22, Lemma 4].

Lemma 3. *Let x_0 be such that $|x_0 - \alpha_i| \geq \Delta_i/c$ for all* real *α_i such that $i \neq 1$ and $c \geq 2$. Then*

$$|A(x_0)| > |a_d|\,|x_0 - \alpha_1|\, c^{-d}\, \mathcal{M}(A)^{-1}\, 2^{\lg \prod_i \Delta_i} \quad .$$

Proof: Let $\Im(\alpha)$ be the imaginary part of $\alpha \in \mathbb{C}$. It holds $|x_0 - \alpha_i| \geq |\Im(\alpha_i)| \geq \Delta_i/2 \geq \Delta_i/c$ and so $|x_0 - \alpha_i| \geq \Delta_i/c$ is true for all the roots of A. Now

$$\begin{aligned}|A(x_0)| &= |a_d| \prod_{i=1}^{d} |x_0 - \alpha_i| = |a_d||x_0 - \alpha_1| \prod_{i=2}^{d} |x_0 - \alpha_i| \\ &\geq |a_d||x_0 - \alpha_1| \frac{c}{\Delta_1} \prod_{i=1}^{d} \frac{\Delta_i}{c} \\ &\geq |a_d||x_0 - \alpha_1| c^{-d} \mathcal{M}(A)^{-1}\, 2^{\lg \prod_i \Delta_i} \quad .\end{aligned}$$

For the last inequality we use $\Delta_1 \leq 2\,\mathcal{M}(A)$, that in turn relies on $\Delta_1 = |\alpha_1 - \alpha_{c_1}| \leq |\alpha_1| + |\alpha_{c_1}| \leq 2\,\mathcal{M}(A)$, where α_{c_1} is a the root closest to α_1. □

2.1 Double Exponential Sieve

In this subsection we follow [3], [8] and [31] to compute an interval that contains the real root and has endpoints "far away" from the endpoints of the initial interval. The difficult case is when the real root is very close to one of the endpoints of I. Next we outline this procedure referring the reader to [31] for its detailed treatment and efficient implementation.

Initially let $\alpha_1 \in I = [a\,..\,b]$ for $a < b$. We compute a new interval $\bar{I} = [\bar{a}\,..\,\bar{b}]$ containing α_1 and such that either $0 \leq \bar{b} - \bar{a} \leq 2\,t$ or $\mathtt{ir}(\bar{I}) \geq 3$. In the first case the midpoint of $\bar{I}$, $\bar{m} = (\bar{a}+\bar{b})/2$, approximates α_1 within a desired error bound t and hence we return either $[\bar{a}\,..\,\bar{m}]$ or $[\bar{m}\,..\,\bar{b}]$, depending on the sign of $A(\bar{m})$. In the second case we can apply bisections to increase the isolation ratio to the level supporting Newton's iteration. We present the analysis of the bisection iteration in the next subsection.

During the first step of DES, we decide whether α_1 lies in $[a\,..\,\frac{a+b}{2}]$ or $[\frac{a+b}{2}\,..\,b]$. W.l.o.g. assume that $a = 0$ and $b = 2$

because our claims are invariant in the shifts and scaling the variable x. Furthermore assume that $\alpha_1 \in [0\,..\,1]$ for the other case is treated similarly. Now the bound $\mathtt{ir}(\bar{I}) \geq 3$ follows where

$$\bar{b} \leq 2\,\bar{a} \ . \tag{3}$$

Next, we write $a_0 = 0$, $b_0 = 1$, $I_0 = [0\,..\,1]$ and evaluate A at the sequence of points $c_k = a_0 + 2w_0/2^{2^k}$, $k = 1, \ldots, g_1$, where $2w_0 = b_0 - a_0$ and $g_1 - 1$ is the maximum index such that $\alpha_1 \in [0\,..\,c_{g_1}]$. If $g_1 = 1$, then we write $a' = c_{g_1}$, $b' = 1$, obtain that $\alpha_1 \in [a'\,..\,b']$ and $\mathtt{ir}([a'\,..\,b']) \geq 5/3$, and yield an interval $\bar{I}$ with $\mathtt{ir}(\bar{I}) \geq 3$ in at most two bisections. Otherwise (if $g_1 > 1$) we reapply the DES procedure to the interval $[c_{g_1}\,..\,1]$, denote by g_2 the number of evaluations of $A(x)$ with $x < \alpha_1$ in this process and ensure (3) unless $g_2 < g_1$. Recursively we obtain a strictly decreasing sequence of intervals I_i, each defined by means of g_i evaluations of $A(x)$ where the sequence $g_1, g_2, \ldots$ strictly decreases. This means that the overall number of evaluations of $A(x)$ in the DES procedure is at most $1 + \sum_{i=1}^{g} g_i \leq 1 + (g_1+1)g_1/2$ for $u \leq g_1$. Moreover, $g_1 = \lceil \lg(\lg(w) + L - 1) \rceil = \mathcal{O}(\lg(\tau + L))$ because otherwise we would have $0 \leq \bar{b} - \bar{a} \leq 2\,t$.

The next lemma provides an approximate variant of the algorithm where at each step of the procedure we use exactly the number of bits needed. We call this variant αDES, from *approximate* Double Exponential Sieve.

Lemma 4 (αDES). *The procedure αDES compresses the isolating interval I to an interval J such that $|J| \leq 1/2^L$ or $\mathtt{ir}(J) \geq 3$ using a working precision and time*

$$\mathcal{O}(-g_1 \lg w + 2^{g_1} + g_1(\lg \mathcal{M}\,(A) + \tau + d\Gamma + \lg \prod_i \Delta_i)) \quad \textit{bits,}$$

$$\widetilde{\mathcal{O}}_B(d^2\tau g_1^2 - dg_1^2 \lg w + dg_1 2^{g_1} + dg_1^2(\lg \mathcal{M}\,(A) + \tau + d\Gamma + \lg \prod_i \Delta_i))$$

or $\widetilde{\mathcal{O}}(d\tau + L)$ and $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$, respectively.

Proof: Initially α_1 lies in the interval $I = I_0 = [a\,..\,b]$. Let $w = w_0 = |b - a|$ be its width. We want to compute the maximum integer g_1 such that $\alpha_1 \in (a\,..\,a + w/2^{2^{g_1}})$. For this we need to evaluate A on $a + w/2^{2^k}$, for $k = 1, \ldots, g_1$. It might happen that the evaluation of A at one of these numbers is zero. To avoid this, at each step, we evaluate A at two points, instead of one. This multiple evaluation is borrowed from [22].

For each step k we define m_1 and m_2 such that

$$a < m_1 = a + w/2^{2^k+1} < m_2 = a + w/2^{2^k} < b$$

and evaluate A over them. At least one of them is not a zero of A. Let $j \in \{1, 2\}$, then for all $i \neq 1$ it holds $|m_j - \alpha_i| \geq w/2^{2^k+1} \Rightarrow w \leq 2^{2^k+1}|m_j - \alpha_i|$ and $|\alpha_1 - m_j| \leq w/2^{2^k}$. Then

$$\begin{aligned}\Delta_i &\leq |\alpha_1 - \alpha_i| \leq |\alpha_1 - m_j| + |m_j - \alpha_i| \\ &\leq w/2^{2^k} + |m_j - \alpha_i| \leq |m_j - \alpha_i| 2^{2^k+1}/2^{2^k} + |m_j - \alpha_i| \\ &\leq 3|m_j - \alpha_i| \ .\end{aligned}$$

Using Lemma 3 with $c = 3$ we get

$$|A(m_j)| > |m_j - \alpha_1||a_d|\, 3^{-d} \mathcal{M}\,(A)^{-1}\, 2^{\lg \prod_i \Delta_i} \ .$$

For at least one of m_j's it holds that $|m_j - \alpha_1| > w/2^{2^k+2}$ (actually this is the half the distance between m_1 and m_2) and hence

$$\begin{aligned}|A(m_j)| &> |a_d| \frac{w}{2^{2^k+2}}\, 3^{-d} \mathcal{M}\,(A)^{-1}\, 2^{\lg \prod_i \Delta_i} \\ &\geq w\, 3^{1-d}\, 2^{-2^{g_1}-2+\lg \prod_i \Delta_i - \lg \mathcal{M}(A) + \lg|a_d|} \ .\end{aligned}$$

Using Lemma 2 the precision needed for this step, ρ_1, satisfies the equation

$$2^{-\rho_1+1}(d+1)^2 2^{\tau + d(\Gamma+2)} < w\, 3^{-d}\, 2^{-2^{g_1}-2+\lg \prod_i \Delta_i - \lg \mathcal{M}(A) + \lg|a_d|} \ ,$$

and thus

$$\rho_1 = \widetilde{\mathcal{O}}(-\lg w + 2^{g_1} + \lg \mathcal{M}\,(A) + \tau + d\Gamma - \lg \prod_i \Delta_i) \ .$$

To support our computation of the desired interval J we double the precision of computing at every DES step. We perform $\lg(\rho_1)$ steps overall; each is essentially an evaluation of A. By applying Horner's rule we yield the cost bound $\widetilde{\mathcal{O}}_B(d(d\tau + \rho_1))$. Similarly, at the i-th step of αDES we perform $g_i \lg(\rho_i)$ evaluations, each at the cost of $\widetilde{\mathcal{O}}_B(d(d\tau + \rho_i))$, where

$$\rho_i = \widetilde{\mathcal{O}}(-\lg w_{i-1} + 2^{g_i} + \lg \mathcal{M}\,(A) + \tau + d\Gamma - \lg \prod_i \Delta_i) \ .$$

Summarizing, the overall cost of performing v steps is bounded by

$$\widetilde{\mathcal{O}}_B\left(\sum_{i=1}^{v} d(d\tau + \rho_i) g_i \lg \rho_i\right) \ .$$

The sequence of g_i's is strictly decreasing, that is $g_i > g_{i+1}$, and so $v < g_1^2$. The produced intervals I_i have widths $w_i < w/2^{2^i}$, and so

$$\sum_{i=0}^{v-1} \lg w_i = g_1 \lg w - 2^{g_1} + 1 \ ,$$

$$\sum_i \rho_i = \mathcal{O}(-g_1 \lg w + 2^{g_1} + g_1(\lg \mathcal{M}\,(A) + \tau + d\Gamma - \lg \prod_i \Delta_i)) \ ,$$

and the overall cost is bounded by

$$\widetilde{\mathcal{O}}_B(d^2\tau g_1^2 - dg_1^2 \lg w + dg_1 2^{g_1} + dg_1^2(\lg \mathcal{M}\,(A) + \tau + d\Gamma - \lg \prod_i \Delta_i)) \ .$$

By noticing that $|a_d| \geq 1$, $\lg w = \mathcal{O}(d\tau)$, $\Gamma = \mathcal{O}(\tau)$, $\mathcal{O}(\lg \mathcal{M}\,(A)) = \mathcal{O}(\tau + \lg d)$ and using Prop. 1 to bound $\lg \prod_i \Delta_i$, we get that the maximum precision needed is $\widetilde{\mathcal{O}}(d\tau + L)$. Finally, the complexity of αDES is $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$. □

Remark 5 (DES). *We call this procedure DES if it uses only exact arithmetic with rational numbers. Then, in the worst case, we perform $g_1^2 = \mathcal{O}(\lg^2(\tau + L))$ evaluations of A at numbers of bitsize $\mathcal{O}(L)$.*

Using Horner's rule, each evaluation costs $\widetilde{\mathcal{O}}_B(d^2(\tau + L))$. Hence, the overall complexity is $\widetilde{\mathcal{O}}_B(d^2(\tau + L)\lg^2(\tau + L)) = \widetilde{\mathcal{O}}_B(d^2(\tau + L))$. This bound is greater by a factor of d than the one supported by αDES.

However, since we are working with exact arithmetic in the bit complexity model, Horner's arithmetic is not optimal. To see this, notice that the output of the evaluation is of bitsize $\widetilde{\mathcal{O}}(d(\tau + L))$. If we use the divide and conquer approach [10, 19], each evaluation costs $\widetilde{\mathcal{O}}_B(d(\tau + L))$ and the overall complexity is $\widetilde{\mathcal{O}}_B(d(\tau + L)\lg^2(\tau + L)) = \widetilde{\mathcal{O}}_B(d(\tau + L))$. This bound is the same as the one supported by αDES.

Nevertheless, this approach has some drawbacks. First, we are forced to work with full precision right from the beginning. Even though this does not affect the worst case bound it is a serious disadvantage for implementations. Second, this approach does not scale well, in the case where we want to refine all the real roots of A. Then, we have to multiply the bound by d, which is not the case for the approximate algorithm. For further details we refer to Section 2.4.

2.2 Bisection(s)

This section covers the second step of the refinement algorithm. Recall that the isolation ratio of a real isolating interval I of a root α_1 is defined as $\mathtt{ir}(I) = 2\,|m - \alpha_{c_1}|/|I|$, where α_{c_1} is the root of A closest to α [30]. Our goal is, using bisections, to achieve an isolation ratio of $\mathtt{ir}(I) \geq 5d^2$, which ensures the quadratic convergence of Newton iteration right from the start [32, Corollary 4.5].

If $\mathtt{ir}(I) = 1 + \delta$ for an interval $I = [a\,..\,b]$, then after k bisection steps, we obtain an interval I_k, such that $\mathtt{ir}(I_k) \geq 1 + 2^k\delta$. Details of the algorithm appear in Alg. 2. We increase the isolation ratio by applying bisections, for which we need to evaluate A. We set *dir*, equal to -1, 0, or 1 to indicate the search direction. If the initial isolation ratio is $\mathtt{ir}(I) = 1+2\delta$, then after k bisections, in the worst case, we increase the isolation ratio to $\mathtt{ir}(I_k) = 1 + 2^k\delta$, where I_k is the new, refined, isolating interval. The variable *dir* takes the values `left` or `right` and specifies if the roots is on the left or on the right of m. If we know in advance whether the closest root of A not belonging to the interval I lies to the left or the right of the midpoint of I then we may set *dir* accordingly. Otherwise we set $dir = \emptyset$. In the latter case, after the first bisection, we are sure about the value of *dir*. Even if this observation does not affect the asymptotic behavior of the algorithm, it can save us a constant number of bisections, which might be important in practice.

If the initial isolation ratio is $r = 1 + 2 \cdot 2^{\lg(r-1)-1}$, then after k steps it becomes $1 + 2^k \cdot 2^{\lg(r-1)-1}$. If our goal is to achieve an isolation ratio R, then

$$1 + 2^k \cdot 2^{\lg(r-1)-1} \geq R \Leftrightarrow 2^k \cdot 2^{\lg(r-1)-1} \geq R - 1 \Leftrightarrow$$
$$k + \lg(r-1) - 1 \geq \lg(R-1) \Leftrightarrow k \geq 1 + \lg\frac{R-1}{r-1} \ .$$

In our case $R = 5d^2$. From the previous step α`DES` guarantees an isolation ratio at least 3 and thus we need to perform $k = \mathcal{O}(\lg(d))$ bisections.

Each bisection consists of an evaluation of A over the midpoint of the corresponding interval and setting *dir* accordingly. We will perform this in an approximate way using the algorithm from [22]. We need the following lemma for the approximate variant, α`BIS`.

Lemma 6. *[22] The approximate bisection for a root $\alpha_1 \in I = [a\,..\,b]$ of A requires a working precision of $\rho = \widetilde{\mathcal{O}}(-\lg w + \tau + d\Gamma + \lg\prod_i \Delta_i))$ bits and has bit complexity $\widetilde{\mathcal{O}}_B(d(-\lg w + \tau + d\Gamma + \lg\prod_i \Delta_i))$, where w is the width of I.*

A single bisection halves an interval of width w, k bisections decrease the width to $w_k = w/2^k$.

In the worst case the interval has the width $w\,2^{-L}$, and so the number of bits needed in the worst case is $\rho = \widetilde{\mathcal{O}}(-\lg w + L + \tau + d\Gamma + \lg\prod_i \Delta_i)$. We perform $\mathcal{O}(\lg(d))$ bisections, and so the overall cost is $\widetilde{\mathcal{O}}_B(d(-\lg w + L + \tau + d\Gamma + \lg\prod_i \Delta_i))$.

Recall that $-\lg w = -\mathcal{O}(\lg\Delta(A)) = \widetilde{\mathcal{O}}(d\tau)$.

Lemma 7 (α`BIS`). *The cost of α`BIS` is $\widetilde{\mathcal{O}}_B(d(-\lg w + L + \tau + d\Gamma + \lg\prod_i \Delta_i))$ or $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$.*

Remark 8 (`BIS`). *If we perform the bisection step using only exact arithmetic with rational numbers, then we perform $\mathcal{O}(\lg(d))$ evaluations of A over numbers of bitsize $\mathcal{O}(L - \lg w)$, in the worst case. Each evaluation costs $\widetilde{\mathcal{O}}_B(d(\tau - \lg w + L))$, using the divide and conquer scheme [19, 10]. Hence, the overall complexity is $\widetilde{\mathcal{O}}_B(d(\tau - \lg w + L)\lg(d)) = \widetilde{\mathcal{O}}_B(d(d\tau + L)) = \widetilde{\mathcal{O}}_B(d^2\tau + dL)$.*

2.3 Bounding the Newton operator

The last step of the refinement algorithm consists in performing a suitable number of Newton iterations to refine the isolating interval up to the required width. The bisections of the previous step ensure that the interval is small enough that Newton iteration converges quadratically, right from the beginning. Actually it satisfies the conditions of the following theorem [32, Corollary 4.5].

Theorem 9. *Suppose both discs $D(m, r)$ and $D(m, r/s)$ for $s \geq 5d^2$ contain a single simple root of a polynomial $A = A(x)$ of degree d. Then Newton's iteration*

$$x_{k+1} = x_k - A(x_k)/A'(x_k), k = 0, 1, \ldots \tag{4}$$

converges quadratically to the root α right from the start provided $x_0 = m$.

First, we estimate the precision needed at each of Newton iteration. Given an interval $[a_k\,..\,b_k]$, let $m_k = (a_k + b_k)/2$ be its middle point, where we apply the Newton operator, that is

$$NA(m_k) = m_k - \frac{A(m_k)}{A'(m_k)} \ ,$$

where $A_k = A(m_k)$, $A'_k = A'(m_k)$, and A' is the derivative of A. We assume that $A_k > 0$ and $A'_k < 0$. The other sign combinations could be treated similarly. Suppose we compute A_k and A'_k using interval arithmetic and a working precision ρ to be specified later. We can assume that their interval representation is $[A_k] = [A_k - \epsilon\,..\,A_k + \epsilon]$ and $[A'_k] = [A'_k - \epsilon\,..\,A'_k + \epsilon]$, both having the same width, ϵ.

The interval evaluation of Newton operator, using the same working precision ρ, results in the interval

$$[NA(m_k)] = \left[m_k - \epsilon - \frac{A_k - \epsilon}{A'_k - \epsilon}\,..\,m_k + \epsilon - \frac{A_k + \epsilon}{A'_k + \epsilon}\right] \ .$$

The width of $[NA(m_k)]$ is $2\epsilon - \frac{A_k+\epsilon}{A'_k+\epsilon} + \frac{A_k-\epsilon}{A'_k-\epsilon}$, and now we ensure its upper bound $t = 2^{-L}w$.

$$2\epsilon - \frac{A_k + \epsilon}{A'_k + \epsilon} + \frac{A_k - \epsilon}{A'_k - \epsilon} \leq t \Rightarrow$$
$$P(\epsilon) = 2\epsilon^3 - t\epsilon^2 - 2(A_k - A'_k + {A'_k}^2)\epsilon + t{A'_k}^2 \geq 0 \ .$$

The coefficient list of P has 2 sign variations and hence from Descartes' rule of signs it follows that P has at most two positive real roots. If there exists such a pair of roots, let them be $\epsilon_1 < \epsilon_2$ and assume that P is positive between 0 and ϵ_1. For the width of $[NA(m_k)]$ to be smaller than $t = 2^{-L}w$, it suffices ϵ to satisfy $0 < \epsilon \leq \min\{1, \epsilon_1\}$. To guarantee this we estimate a (positive) lower bound on the roots of the P and require ϵ to be smaller than it. Combine

Lemma 2 and Cauchy's bound, e.g. [39, 26] to obtain

$$\epsilon \leq \frac{t {A'_k}^2}{t + 2(A_k - A'_k + {A'_k}^2)} \ .$$

by working with a precision ρ that satisfies

$$2^{-\rho+1}(d+1)^2 2^{\tau+d(\Gamma+2)} \leq \epsilon \leq \frac{t {A'_k}^2}{t + 2(A_k - A'_k + {A'_k}^2)} \ . \quad (5)$$

Hence, we can express ρ as a function of the desired width.

It remains to bound the evaluations A_k and A'_k. At the k-th step, given an interval I_k, we apply Newton operator on its midpoint, m_k, and deduce that $\Delta_i < 2|m_k - \alpha_i|$ for all $i \neq 1$. Indeed for $m_k \leq \alpha_1 \leq \alpha_i$, it holds $|m_k - \alpha_i| \geq |\alpha_1 - \alpha_i| \geq \Delta_i$, whereas for $\alpha_1 \leq m_k \leq \alpha_i$ it holds $|\alpha_1 - m_k| \leq |m_k - \alpha_i|$, because α_i lies outside the isolating interval. Therefore $\Delta_i \leq |\alpha_i - \alpha_1| \leq |\alpha_i - m_k| + |\alpha_1 - m_k| \leq 2|m_k - \alpha_i|$.

So using Lemma 3 we get

$$|A_k| \geq |m_k - \alpha_1||a_d|\, 3^{-d} \mathcal{M}(A)^{-1}\, 2^{\lg \prod_i \Delta_i} \ . \quad (6)$$

For the approximations achieved by Newton iterations [31, 8], when the convergence is quadratic, we have

$$|m_k - \alpha_1| = 2^{4-2^k} |m_0 - \alpha_1| \ . \quad (7)$$

Obviously

$$|m_0 - \alpha_1| \geq t = 2^{-L} w \ ,$$

since the required width is not achieved from the initial interval, and so

$$|m_k - \alpha_1| = 2^{4-2^k} |m_0 - \alpha_1| \geq 2^{4-2^k - L + \lg w} \ , \quad (8)$$

which leads to

$$|A_k| \geq |a_d|\, 2^{4-2^k - L + \lg w}\, 3^{-d} \mathcal{M}(A)^{-1}\, 2^{\lg \prod_i \Delta_i} \ . \quad (9)$$

We need a similar bound for $|A'_k|$. Let α'_i be the roots of A'. We assume that A' is square-free. This is no loss of generality since we can estimate the required quantities using the square-free part of A'.

Let the two roots of A' that are closer to α_1 be α'_1 and α'_2, which are located to the left and to the right of α_1, respectively. Let α' denote any other root of A'. Then it holds that $|m_k - \alpha'_i| \geq |\alpha' - \alpha'_i| \geq \Delta'_i$, where Δ'_i is the local separation bound of α'_i and where α' is either α'_1 or α'_2 depending on which side from m_k the root α'_i lies. To see this assume that α'_i lies on the right of α'_2. Then it holds $m_k \leq \alpha'_2 \leq \alpha'_i$, and so $|m_k - \alpha'_2| \geq |\alpha'_2 - \alpha'_i| \geq \Delta'_i$. A similar argument holds when α'_i lies on the left of α'_1. Therefore

$$\begin{aligned}
|A'_k| = |A'(m_k)| = |d\, a_d| \prod_{i=1}^{d-1} |m_k - \alpha'_i| = \\
= |d\, a_d||m_k - \alpha'_1||m_k - \alpha'_2| \prod_{i=3}^{d} |m_k - \alpha'_i| \\
\geq |d\, a_d||m_k - \alpha'_1||m_k - \alpha'_2| \frac{1}{\Delta'_1 \Delta'_2} \prod_{i=1}^{d} \Delta'_i \\
\geq |d\, a_d||m_k - \alpha'_1||m_k - \alpha'_2| \mathcal{M}(A')^{-2}\, 2^{-2}\, 2^{\lg \prod_i \Delta'_i} \ ,
\end{aligned}$$

where we used the inequality $\Delta'_i \leq 2\mathcal{M}(A')$.

We bound $|m_k - \alpha'_1|$ and $|m_k - \alpha'_2|$ using the width of I_k. We notice that m_k is closer to α_1 than to α'_1 and α'_2. This is so because both α'_1 and α'_2 lie outside the interval of interest, which holds because of the quadratic convergence of Newton operator.

The complex disc with diameter the interval I_0 satisfies the assumptions of Thm. 9, and this is also the case for all discs with the centers at m_k and radii $|I_k|/2$. In other words, the roots α'_1 and α'_2 lie outside the $5d^2$-dilations of these discs, that is the centers m_k lie much closer to α_1 than to α'_1 and α'_2.

So using Eq. (8) we obtain $|m_k - \alpha'_1| \geq |m_k - \alpha_1| \geq 2^{4-2^k - L + \lg w}$. Hence

$$|A'_k| \geq |d\, a_d|\, 2^{8-2^{k+1}-2L+2\lg w}\, \mathcal{M}(A')^{-2}\, 2^{-2+\lg \prod_i \Delta'_i} \ . \quad (10)$$

Now we can bound the right-hand side of inequality (5). First, we bound the numerator. For all m_k it holds that $|m_k| \leq 2^{\Gamma} \leq 2^{\tau+1}$, see Eq. (1). Thus

$$|A'_k| = |A'(m_k)| \leq \sum_{i=1}^{d} i\, |a_i|\, 2^{(i-1)\Gamma} \leq d\, 2^{\tau}\, 2^{d\Gamma} \leq 2^{2d\tau + \lg d} \ ,$$

and so

$$t {A'_k}^2 \leq 2^{-L + \lg w + 6d\tau + 2\lg d} \ . \quad (11)$$

Next, we bound the denominator of the upper bound in (5). To simplify the notation let $|A_k| \geq 2^{-\ell_1} > 0$ and $|A'_k| \geq 2^{-\ell_2} > 0$. For the exact bounds we refer the reader to equations (9) and (10). Recall that we assume A'_k to be negative and hence $-A'_k > 0$. Then

$$\begin{aligned}
t + 2(A_k - A'_k + {A'_k}^2) &\geq 2^{\min\{-\ell_1, -2\ell_2, -L + \lg w\}} \\
&\geq 2^{-\max\{\ell_1, 2\ell_2, L - \lg w\}} \ ,
\end{aligned}$$

and so

$$\frac{1}{t + 2(A_k - A'_k + {A'_k}^2)} \leq 2^{\max\{\ell_1, 2\ell_2, L - \lg w\}} \ . \quad (12)$$

Combining (9) and (10) with (11), (12) and (5) we deduce that the required precision is

$$\widetilde{\mathcal{O}}\left(L - \lg w + 2^k + \tau + \lg\left(\mathcal{M}(A)\,\mathcal{M}(A')\right) - \lg \prod_i \Delta_i \prod_i \Delta'_i \right) \ .$$

We want to achieve $|m_k - \alpha_1| \leq w/2^L$. Using Eq. (7) we get

$$|m_k - \alpha_1| \leq 2^{4-2^k} \frac{w}{2} \leq \frac{w}{2^L} \Rightarrow k \geq \lg(L+3) \ .$$

Hence, we need to apply the Newton operator $k = \mathcal{O}(\lg(L))$ times, to refine the interval by a factor of 2^{-L}. So the overall bit complexity of this step is

$$\widetilde{\mathcal{O}}_B\left(d\left(L - \lg w + d\tau + \lg\left(\mathcal{M}(A)\,\mathcal{M}(A')\right) - \lg \prod_i \Delta_i \prod_i \Delta'_i \right)\right) \ .$$

Note that $|a_d| \geq 1$, $\mathcal{O}(\lg \mathcal{M}(A)) = \mathcal{O}(\tau + \lg d)$, $\mathcal{O}(\lg \mathcal{M}(A')) = \mathcal{O}(\tau \lg d)$, $-\lg w = \mathcal{O}(d\tau)$, use Prop. 1 to bound $\lg \prod_i \Delta_i$ and $\lg \prod_i \Delta'_i$, and obtain that the precision $\widetilde{\mathcal{O}}(d\tau + L)$ is sufficient for us. So the Boolean complexity of αDES is $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$.

Lemma 10 (αNEWTON). *The maximum number of bits needed by Newton iterations is $\widetilde{\mathcal{O}}(d\tau + L)$ and the total complexity of the Newton step is $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$.*

We should mention that there is no need to realize the Newton iteration using interval arithmetic. However, it is easier to estimate theoretically the working precision needed using the formalization of interval arithmetic.

Remark 11 (NEWTON). *We can also estimate the complexity of Newton iterations in the case where only exact arithmetic with rational number is used. We need to perform $\mathcal{O}(\lg(L))$ Newton iterations, each of which consists of an evaluation of A and its derivative over numbers of bitsize $\mathcal{O}(\tau + L - \lg w)$, in the worst case. The cost of the evaluations is $\widetilde{\mathcal{O}}_B(d(\tau + L - \lg w))$. Hence the overall complexity is $\widetilde{\mathcal{O}}_B(d(\tau + L - \lg w)\lg(L)) = \widetilde{\mathcal{O}}_B(d^2\tau + dL))$.*

2.4 Overall complexity of $\mathtt{R}_3$ and $\alpha\mathtt{R}_3$

Theorem 12 ($\alpha\mathtt{R}_3$). *We can refine an isolating interval of a real root of A to decrease its width by a factor of 2^{-L} by using $\alpha\mathtt{R}_3$ or $\mathtt{R}_3$ with Boolean complexity $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$.*

The bound of Th. 12 for $\alpha\mathtt{R}_3$ holds even if we want to refine all the real roots of A. The main operation needed is the evaluation of a polynomial at, at most, r rational numbers, where $r \le d$ is the number of real roots that we need refine. We can evaluate a polynomial at d points by using $\mathcal{O}(d\lg^2(d))$ arithmetic (field) operations [5, 38]. These fast multipoint evaluation algorithms are numerically unstable, see, e.g. [4], but from Lemmata 4, 6 and 10 we deduce that the maximum working precision needed by the three subroutines of $\alpha\mathtt{R}_3$ is $\widetilde{\mathcal{O}}(d\tau + L)$. Therefore, the overall complexity is $\widetilde{\mathcal{O}}_B(d^2\tau + dL))$ for refining all the roots.

In the exact version, $\mathtt{R}_3$, we perform evaluations at numbers of bitsize L. The output of such an evaluation results in rational numbers of bitsize $\widetilde{\mathcal{O}}(d(\tau + L))$. When we isolate all the r real roots, the bitsize of the output is $\widetilde{\mathcal{O}}(r\,d(\tau + L)) = \widetilde{\mathcal{O}}(d^2(\tau + L))$, which is also a lower bound on the Boolean complexity of the refinement process. This exceeds the bound for $\alpha\mathtt{R}_3$ by a factor of d.

2.5 Requirements for the isolating intervals

Our algorithms support the complexity bound of Thm. 12 provided that we are given a real m and a positive r such that the *root-isolation disc* $D(m, r) = \{x : |x - m| < r\}$ contains a single simple real root α of A, and no other roots of A, and furthermore α is not very close to the boundary circle of the disc, namely

$$|\alpha - m|(1 + c'/d^c) \le r \ , \qquad (13)$$

for two real constants $c' > 0$ and c. Our argument in Section 2.2 shows that under the latter assumption it is sufficient to apply $\mathcal{O}(\log d)$ bisections to strengthen bound (13) to the level $5d^2|\alpha - m| \le r$. Then we can apply Theorem 9 to ensure quadratic convergence of Newton's iteration, and then complete our algorithms and proofs.

Is it simple to ensure bound (13) at a low cost? For the worst case input this is not simpler than to approximate the root α very closely. Indeed the divide-and-conquer algorithms (cf. [29], [35]) can compute a real isolation interval for a single simple root, but produce such intervals already well isolated from all other roots, and then our construction is not needed. On the other hand root-finders working on the real line such as the subdivision algorithms produce such intervals independently of the distribution of nonreal roots on the complex plane. In this case we cannot exclude any unfavorable distribution of them. On the average input, however, violation of the isolation assumption (13) seems to be rather pathological (see also the next section).

Algorithm 1: ISOLATION_DISCS(A, $I_1, \ldots, I_r$)

Input: $A \in \mathbb{Z}[X]$, and isolating intervals $I_1, \ldots, I_r$
Output: Isolation ratio for convergence of Newton.

1. Apply a fixed number of bisections as well as bisections of exponenent to all r input intervals $I_1, \ldots, I_r$, which transforms them into subintervals $\bar{I}_1, \ldots, \bar{I}_r$.
2. Recall that the Möbius map $z = \frac{x+\sqrt{-1}}{x-\sqrt{-1}}$ maps the real line $\{x : \Im(x) = 0\}$ into the unit circle $C_1 = \{z : |z| = 1\}$. Note that $x = \sqrt{-1}\frac{z-1}{z+1}$ and compute the coefficients of the polynomial $B(z) = (z + 1)^n A(\sqrt{-1}\frac{z-1}{z+1})$.
3. Apply the root-radii algorithm (cf. [29], [35]) to approximate the root radii of this polynomial. In particular this defines an annulus about the circle C_1, which contains the images of the r real roots of the polynomial $A(x)$ and no images of its nonreal roots.
4. Compute the boundary circles of the image of this annulus in the converse map $x = \sqrt{-1}\frac{z-1}{z+1}$.
5. For all subintervals $\bar{I}_1, \ldots, \bar{I}_r$ compute at first the distances $\bar{d}_1, \ldots, \bar{d}_r$ from their midpoints to these two boundary circles and then the ratios $2d_1/|\bar{I}_1, \ldots, 2d_r/|\bar{I}_r|$.
6. RETURN the minimum of the ratios as the quaranteed isolation ratio for the input intervals.

A natural question arises: How can we test whether this assumption holds for a given polynomial A and a real interval I containing its single simple root α? In fact very easily: we can just apply our algorithms. They compute a sequence of real inclusion intervals $(a_h \,..\, b_h)$, for $h = 0, 1, \ldots,$ where $(a_0 \,..\, b_0) = I$ and $b_h > a_h$ for all h. We verify the inclusion property by checking whether $A(a_h)A(b_h) < 0$ and either observe that h bisection steps decrease the width of the isolating interval by a factor of 2^h or otherwise conclude that the assumption (13) is certainly violated. This *test by action* requires negligible extra cost.

Alternatively, given m and r, we can test whether the disc $D(m, r)$ contains only one root by applying the Schur–Conn test, partial inverses of Descartes' rule of sign, e.g. [16] or the root-radii algorithms of [35] (cf. [28, Section 4]), which approximate the distances from m to all roots of A within, say 1% error. These a priori tests, however, have no advantage versus the test by action and have a little greater cost.

Remark 13. *Suppose we have r real intervals $I_1, \ldots, I_r$, each containing a single simple root of A. In this case our algorithm* ISOLATION_DISCS *in Alg. 1 is a single a priori test of the existence of all the r root-isolation discs. Our present paper does not use this algorithm, but it may be of independent interest, and our next research plan includes estimation of its Boolean complexity, which seems to be dominated at Stage 3. This stage is quite inexpensive, according to the estimates in [35] and [29].*

3. AVERAGE ANALYSIS OF DES AND αDES

The most time consuming part of $\mathtt{R}_3$ and $\alpha\mathtt{R}_3$ is the αDES procedure. It requires, in the worst case,

$$g_1^2 = \lceil \lg(\lg(w) + L - 1)\rceil^2 = \mathcal{O}(\lg^2(\tau + L))$$

evaluations of our input polynomial A. This occurs where (other) roots of A lie very close to the endpoints of the initial interval I. However, in practice this behavior is rare, if it occurs at all. To explain this phenomenon, Pan and Linzer [30] estimated the average number of steps of DES under the assumption that a real root is uniformly distributed in an interval and concluded that in this case $\mathtt{R}_3$, and hence $\alpha\mathtt{R}_3$, needed a constant number of steps, with a high probability.

Even though the assumption on the equidistribution of the real roots in [30] is plausible, we are not aware of any distribution on the coefficients that results in such a behavior for the roots. We consider an average case analysis in the case where the root we approximate is a real root of a random Weyl or $SO(2)$ polynomial [18]. The density function of the real roots is considerably different in these cases. We will also arrive to the same conclusion, that is $\mathtt{R}_3$ and $\alpha\mathtt{R}_3$ perform a constant number of steps with high probability.

3.1 Weyl polynomials

Weyl polynomials, are random polynomials of the form $A = \sum_{i=0}^{d} a_i x^i/\sqrt{i!}$, where the coefficients a_i are independent standard normals. Alternatively, we could consider them as $A = \sum_{i=0}^{d} a_i x^i$, where a_i are normals of mean zero and variance $1/\sqrt{i!}$. As the degree grows, the real roots, except a constant number of them, are in the interval $[-\sqrt{d}\,..\,\sqrt{d}]$. Their (asymptotic) probability density function is $f(t) = \frac{1}{2\sqrt{d}}$.

The probability that $\alpha_1 \in [a\,..\,a + w/2^{2^k}]$ is

$$\Pr[\alpha_1 \in [a\,..\,a+w/2^{2^k}]] = \Pr[g_1 > k] = \int_a^{a+w/2^{2^k}} f(t)dt = \frac{w}{2^{2^k+1}\sqrt{d}} \ ,$$

which rapidly goes to zero as k, but also as d, grows.

3.2 $SO(2)$ polynomials

Random polynomials of the form $A = \sum_{i=0}^{d} a_i x^i$, where the coefficients are i.d. normals with mean zero and variances $\binom{d}{i}$, where $0 \le i \le d$ are called $SO(2)$ polynomials. Alternatively, we could consider them as $A = \sum_{i=0}^{d} \sqrt{\binom{d}{i}}\, a_i\, x^i$, where a_i are i.i.d. standard normals. They are called $SO(2)$ because the joint probability distribution of their zeros is $SO(2)$ invariant, after homogenization.

The (asymptotic) probability density function of the real roots of $SO(2)$ random polynomials is $f(t) = \frac{1}{\pi(1+t^2)}$.

Assume that the isolating interval $I = [a\,..\,b]$ is a subset of $(0\,..\,1)$, and let its width be w. We can treat the case where I is subset of $(1\,..\,\infty)$ similarly. The probability that $\alpha_1 \in [a\,..\,a + w/2^{2^k}]$ is

$$\Pr\left[\alpha_1 \in [a\,..\,a + w/2^{2^k}]\right] = \Pr[g_1 > k]$$
$$= \int_a^{a+w/2^{2^k}} f(t)dt = \arctan(a) - \arctan(a + w/2^{2^k})$$
$$= \arctan \frac{w/2^{2^k}}{1 + a^2\, w/2^{2^k}} \le \frac{w}{2^{2^k} + a^2 w} - \frac{1}{3}\left(\frac{w}{2^{2^k} + a^2 w}\right)^3 \ ,$$

which rapidly goes to zero as in the case of Weyl polynomials. However, now there is no dependence on the degree but only on the endpoints of the (initial) isolating interval.

4. CONCLUSIONS AND FUTURE WORK

We present an approximate variant of a real root refinement algorithm that is based on the *Bisection of the Exponents*, or *Double Exponential Sieve* algorithm, bisection and Newton operator. The complexity of the algorithm is $\widetilde{\mathcal{O}}_B(d^2\tau + dL)$. We are currently implementing the presented algorithm for real root refinement and the first results are quite encouraging. We plan to report a detailed experimental analysis in the near future.

Can we combine αDES with the approximate version of QIR in [22] to provide an alternative method to guarantee quadratic behavior? We believe that this is a very interesting approach to explore.

For random polynomials, it is reasonable to assume that we can derive an even faster algorithm for real root refinement that takes advantage of the distribution of the roots.

Acknowledgments

VP is supported by NSF Grant CCF–1116736 and PSC CUNY Awards 64512–0042 and 65792–0043. ET is partially supported by the EXACTA grant of the National Science Foundation of China (NSFC 60911130369) and the French National Research Agency (ANR-09-BLAN-0371-01), GeoLMI (ANR 2011 BS03 011 06), HPAC (ANR ANR-11-BS02-013) and an FP7 Marie Curie Career Integration Grant.

5. REFERENCES

[1] J. Abbott. Quadratic interval refinement for real roots. In *ISSAC 2006, poster presentation*, 2006. http://www.dima.unige.it/~abbott/.

[2] S. Basu, R. Pollack, and M-F.Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2nd edition, 2006.

[3] M. Ben-Or and P. Tiwari. Simple algorithms for approximating all roots of a polynomial with real roots. *Journal of Complexity*, 6(4):417–442, 1990.

[4] D. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, pages 127–173, 2000.

[5] D. Bini and V. Pan. *Polynomial and Matrix Computations*, volume 1: Fundamental Algorithms. Birkhäuser, Boston, 1994.

[6] D. Bini and V. Y. Pan. Parallel complexity of tridiagonal symmetric eigenvalue problem. In *Proc. 2nd Annual ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 384–393, 1991.

[7] D. Bini and V. Y. Pan. Practical improvement of the divide-and-conquer eigenvalue algorithms. *Computing*, 48:109–123, 1992.

[8] D. Bini and V. Y. Pan. Computing matrix eigenvalues and polynomial zeros where the output is real. *SIAM J. Comput.*, 27(4):1099–1115, 1998.

[9] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1998.

[10] M. Bodrato and A. Zanoni. Long integers and polynomial evaluation with estrin's scheme. In *Proc. 11th Int'l Symp. on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 39–46. IEEE, 2011.

[11] J. Cheng, S. Lazard, L. M. Peñaranda, M. Pouget, F. Rouillier, and E. P. Tsigaridas. On the topology of planar algebraic curves. In J. Hershberger and E. Fogel, editors, *Proc. 25th Annual ACM Symp. Comput. Geom. (SoCG)*, pages 361–370, Århus, Denmark, 2009.

[12] G. E. Collins and W. Krandick. A hybrid method for high precision calculation of polynomial real roots. In *Proc. ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 47–52, 1993.

[13] J. H. Davenport. Cylindrical algebraic decomposition. Technical Report 88–10, School of Mathematical Sciences, University of Bath, England, available at: http://www.bath.ac.uk/masjhd/, 1988.

[14] D. I. Diochnos, I. Z. Emiris, and E. P. Tsigaridas. On the asymptotic and practical complexity of solving bivariate

systems over the reals. *J. Symbolic Computation*, 44(7):818–835, 2009. (Special issue on ISSAC 2007).

[15] Z. Du, V. Sharma, and C. K. Yap. Amortized bound for root isolation via Sturm sequences. In D. Wang and L. Zhi, editors, *Int. Workshop on Symbolic Numeric Computing*, pages 113–129, School of Science, Beihang University, Beijing, China, 2005. Birkhauser.

[16] A. Eigenwillig. *Real root isolation for exact and approximate polynomials using Descartes' rule of signs*. PhD thesis, Universität des Saarlandes, 2008.

[17] P. Emeliyanenko and M. Sagraloff. On the complexity of solving a bivariate polynomial system. In *Proc. 37th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 154–161, Grenoble, France, July 2012. ACM.

[18] I. Z. Emiris, A. Galligo, and E. P. Tsigaridas. Random polynomials and expected complexity of bisection methods for real solving. In S. Watt, editor, *Proc. 35th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 235–242, Munich, Germany, July 2010. ACM.

[19] W. Hart and A. Novocin. Practical divide-and-conquer algorithms for polynomial arithmetic. In *Proc. CASC*, volume 6885 of *LNCS*, pages 200–214. Springer, 2011.

[20] J. R. Johnson. *Algorithms for Polynomial Real Root Isolation*. PhD thesis, The Ohio State University, 1991.

[21] M. Kerber. On the complexity of reliable root approximation. In *In Proc. 11th Int'l Wkshp on Computer Algebra in Scientific Computing (CASC)*, volume 5743 of *LNCS*, pages 155–167, 2009.

[22] M. Kerber and M. Sagraloff. Efficient real root approximation. In *Proc. 36th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 209–216, San Jose, CA, USA, June 2011. ACM.

[23] M. Kerber and M. Sagraloff. A worst-case bound for topology computation of algebraic curves. *J. Symb. Comput.*, 47(3):239–258, 2012.

[24] A. Mantzaflaris, B. Mourrain, and E. P. Tsigaridas. On continued fraction expansion of real roots of polynomial systems, complexity and condition numbers. *Theor. Comput. Sci.*, 412(22):2312–2330, 2011.

[25] J. M. McNamee and V. Y. Pan. *Numerical methods for roots of polynomials (II)*, chapter 15. Elsevier, 2013.

[26] M. Mignotte. *Mathematics for Computer Algebra*. Springer-Verlag, New York, 1991.

[27] V. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Comp. and Math. (with Appl.)*, 31:97–138, 1996.

[28] V. Pan. Approximating complex polynomial zeros: modified Weyl's quadtree construction and improved Newton's iteration. *J. of Complexity*, 16(1):213–264, 2000.

[29] V. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and rootfinding. *J. Symbolic Computation*, 33(5):701–733, 2002.

[30] V. Pan and E. Linzer. Bisection acceleration for the symmetric tridiagonal eigenvalue problem. *Numerical Algorithms*, 22(1):13–39, 1999.

[31] V. Y. Pan, B. Murphy, R. E. Rosholt, G. Qian, and Y. Tang. Real root-finding. In S. M. Watt and J. Verschelde, editors, *Proc. 2nd ACM Int'l Work. Symbolic Numeric Computation (SNC)*, pages 161–169, 2007.

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

[33] F. Rouillier. On solving systems of bivariate polynomials. *Mathematical Software–ICMS 2010*, 6327:100–104, 2010.

[34] M. Sagraloff. When Newton meets Descartes: A simple and fast algorithm to isolate the real roots of a polynomial. In *Proc. 37th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 297–304, Grenoble, France, July 2012. ACM.

[35] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany, 1982. URL: http://www.iai.uni-bonn.de/~schoe/fdthmrep.ps.gz.

[36] V. Sharma, Z. Du, and C. Yap. Robust approximate zeros. *Proc. European Symposium of Algorithms (ESA)*, pages 874–886, 2005.

[37] A. Strzeboński and E. P. Tsigaridas. Univariate real root isolation in an extension field. In A. Leykin, editor, *Proc. 36th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 321–328, San Jose, CA, USA, June 2011. ACM.

[38] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge Univ. Press, Cambridge, U.K., 2nd edition, 2003.

[39] C. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.

Algorithm 2:
IncrIsolationRatio_by_subdiv(A, I, r, R, dir)

Input: $A \in \mathbb{Z}[X], I = [a, b],\ L \in \mathbb{Z}$
Output: (r, I), where $r \geq R$
Data: Initially $r < R$. The direction of the closest root, dir, may or may not be given.

```
if dir = ∅ then
    m ← (a+b)/2 ;
    s_m ← sgn(A(m)) ;
    if s = 0 then I = [m..m]; RETURN (r = ∞, I) ;
    if s_l · s_m < 0 then I ← [a..m]; dir = left ;
    if s_r · s_m < 0 then I ← [m..b]; dir = right ;
    r ← 2(r − 1) + 1 ;
while r < R do
    m ← (a+b)/2 ;
    s_m ← sgn(A(m)) ;
    if s = 0 then I = [m..m]; RETURN (r = ∞, I) ;
    if s_l · s_m < 0 then I ← [a..m]; cdir = left ;
    if s_r · s_m < 0 then I ← [m..b]; cdir = right ;
    if cdir ≠ dir then
        r ← 2(r − 1) + 1
    else
        r ← 2(r + 1) − 1
RETURN (r, I)
```

Algorithm 3: Newton(A, I, t)

Input: $A \in \mathbb{Z}[X], I = [a, b],\ t \in \mathbb{Z}$
Output: Interval J such that $w(J) < 2^{-L}w$
Data: It holds that $\mathtt{ir}(I) > 20d^2$

```
x_1 ← m(I) ;
x_0 ← ∞ ;
while |x_1 − x_0| > 2^{−L} w do
    SWAP(x_1, x_0) ;
    x_1 ← x_0 − A(x_0)/A'(x_0) ;
RETURN [x_0..x_1]
```