Univariate real root isolation in an extension field and applications

Adam Strzebonski, Elias Tsigaridas

Univariate real root isolation in an extension field and applications

Adam Strzebonski* Elias P. Tsigaridas†

Abstract

We present algorithmic, complexity and implementation results for the problem of isolating the real roots of a univariate polynomial in $B_\alpha \in L[y]$, where $L = \mathbb{Q}(\alpha)$ is a simple algebraic extension of the rational numbers.

We revisit two approaches for the problem. In the first approach, using resultant computations, we perform a reduction to a polynomial with integer coefficients and we deduce a bound of $\widetilde{\mathcal{O}}_B(N^8)$ for isolating the real roots of B_α, where N is an upper bound on all the quantities (degree and bitsize) of the input polynomials. The bound becomes $\widetilde{\mathcal{O}}_B(N^7)$ if we use Pan's algorithm for isolating the real roots. In the second approach we isolate the real roots working directly on the polynomial of the input. We compute improved separation bounds for the roots and we prove that they are optimal, under mild assumptions. For isolating the real roots we consider a modified Sturm algorithm, and a modified version of DESCARTES' algorithm. For the former we prove a Boolean complexity bound of $\widetilde{\mathcal{O}}_B(N^{12})$ and for the latter a bound of $\widetilde{\mathcal{O}}_B(N^5)$. We present aggregate separation bounds and complexity results for isolating the real roots of all polynomials B_{α_k}, when α_k runs over all the real conjugates of α. We show that we can isolate the real roots of all polynomials in $\widetilde{\mathcal{O}}_B(N^6)$. Finally, we implemented the algorithms in C as part of the core library of MATHEMATICA and we illustrate their efficiency over various data sets.

Keywords real root isolation, algebraic polynomial, field extension, separation bounds, Sturm sequences, Descartes' rule of sign

1 Introduction

Real root isolation is a very important problem in computational mathematics. Many algorithms are known for isolating the real roots of a polynomial having integer or rational coefficients that are either based solely on operations with rational numbers, see [8, 13, 26, 35] and references therein, or they follow a numerical, but certified approach, see [28, 40] and references therein. In this paper we consider a variation of the problem where the coefficients of the polynomial are polynomial functions of a real algebraic number, that is they belong to a simple algebraic extension of the rationals.

The reader might refer to the end of the Introduction for a thorough presentation of the Notation that we use in this paper. We consider the following problem:

Problem 1. *Let α be a real algebraic number with isolating interval representation $\alpha \cong (A, \mathcal{I})$, where $A = \sum_{i=0}^{m} a_i\, x^i$, $\mathcal{I} = [\mathsf{a}_1, \mathsf{a}_2]$, $\mathsf{a}_{1,2} \in \mathbb{Q}$, $\mathsf{deg}(A) = m$, and $\mathcal{L}(A) = \tau$. By $\mathcal{L}(A)$ we denote the maximum bitsize of the coefficients of A. Let $B_\alpha = \sum_{i=0}^{n} b_i(\alpha)\, y^i \in \mathbb{Z}(\alpha)[y]$ be square-free, where $b_i(x) = \sum_{j=0}^{\eta_i} c_{i,j}\, x^j \in \mathbb{Z}[x]$, $\mathcal{L}(c_{i,j}) \leq \sigma$, b_n and A are relatively prime, and $\eta_i < m$, for $0 \leq i \leq n$. What is the Boolean complexity of isolating the real roots of B_α?*

*Wolfram Research Inc.,100 Trade Centre Drive, Champaign, IL 61820, U.S.A. Email: adams@wolfram.com

†INRIA, Paris-Rocquencourt Center, POLSYS Project.
Sorbonne Universités, UPMC Univ Paris 06, CNRS, INRIA, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France.
Email: elias.tsigaridas@lip6.fr

Rump [37], see also [36], presented an algorithm for the problem that is an extension of Collins and Loos [7] algorithm for integral polynomials. Johnson [18] presented and compared various algorithms for Problem 1. He considered a norm based algorithm that reduces the problem to root isolation of integral polynomial (this is the approach that we consider in Sec. 3) and extended three algorithms used for integral polynomials, i.e. Sturm (we present it in Sec. 4.3), the algorithm based on a derivative sequence and Rolle's theorem [7], and the algorithm based on Descartes' rule of sign [6] (we present a modified version in Sec. 4.4). Johnson and Krandick [17] modified the latter and managed to replace exact arithmetic, when possible, with certified floating point operations; a novelty that speeds up considerably the computations. Along the same lines, Rouillier and Zimmermann [35] presented an algorithm for integral polynomials that exploits adaptive multiprecision techniques that could be used for Problem 1, if we approximate the real algebraic number up to a sufficient precision. In a series of works [11, 12, 23] a bitstream version of Descartes' algorithm was introduced. The coefficients of the input polynomial are considered to be real numbers that we can approximate up to arbitrary precision. We use the most recent version of this approach, which is due to Sagraloff [38], see also [24], to tackle Problem 1. We also refer the reader to [39] for an exact approach with improved complexity bounds. Last but not least, let us also mention the numerical algorithms due to Pan [28] and Schönhage [40], that could be also used if we approximate α in our problem up to a sufficient precision.

Rioboo [34] considered various symbolic algorithms for operations with real algebraic numbers, based on quasi Sylvester sequences. These algorithms could be used for Problem 1, and they are closely connected with the Sturm algorithm that we present (Sec. 4.3). However, we use different subalgorithms for sign evaluations and solving polynomials. The focus in [34] is on efficient implementation of the real closure in AXIOM.

Problem 1 is closely related to real root isolation of of triangular systems and regular chains. In [5, 22, 44, 45] algorithms and implementations are presented for isolating the real roots of triangular polynomial systems, based on interval arithmetic and the so-called *sleeve* polynomials. In the case of two variables the problem at study is similar to Problem 1. In this line of research the coefficients of the algebraic polynomial are replaced with sufficiently refined intervals, hence obtaining upper and lower bounds (i.e. a sleeve) for the polynomial. Isolation is performed using evaluations and exclusion predicates that involve the non-vanishing of the derivative. To our knowledge there is no complexity analysis of the algorithms. Nevertheless in [5] evaluation bounds are presented, which are crucial for the termination of the algorithm, based on separation bounds of polynomial systems. However, the systems used for the bounds involve the derivative of the polynomial (this is needed for the exclusion criterion), which is not the case for our approach. In [4] the problem of real root isolation of 0-dim square-free regular chains is considered. A generalization of Vincent-Collins-Akritas (or Descartes) algorithm is used to isolate the real roots of of polynomials with real algebraic numbers as coefficients. This approach is similar to the direct strategy that we study. To our knowledge the authors do not present a complexity analysis since they focus on efficient algorithms and implementation in MAPLE.

We revisit two approaches for isolating the real roots of a square-free polynomial with coefficients in a simple algebraic extension of the rational numbers. The first, indirect, approach (Sec. 3), already presented in [18], is to find a polynomial with integer coefficients which is zero at all roots of B_α, isolate its real roots, and identify the intervals which contain the roots of B_α. We compute separation bounds for the resulting polynomial (Lem. 9) similar to Rump's [37] and a new aggregate version. The complexity of the algorithm is $\widetilde{\mathcal{O}}_B(N^8)$, where N is an upper bound on all the quantities (degrees and bitsizes) of the input. Using Pan's algorithm [28] for solving the bound becomes $\widetilde{\mathcal{O}}_B(N^7)$. The second approach (Sec. 4.1) is to isolate the roots of the input polynomial directly, using either Sturm's algorithm or a modified Descartes algorithm. We analyze the worst-case asymptotic complexity of both algorithms and we obtain Boolean complexity bounds $\widetilde{\mathcal{O}}_B(N^{12})$ and $\widetilde{\mathcal{O}}_B(N^5)$, respectively. To achieve these complexity bounds, we estimate improved separation bounds for the roots (Sec. 4.1 and Lem. 11); we also establish the optimality of the separation bounds (Sec. 4.2). The bounds are better than the previously known ones [18, 36] by, at least, a factor of N. We also present new aggregate separation bounds (Lem. 14), and consider the complexity of isolating the real roots of B_α, when α runs over all the real roots of A, see the definition of Problem 1. The derived worst case complexity bound for isolating the roots of a single polynomial matches that of isolating the roots of

all polynomials B_α.

We empirically compare the performance of the indirect approach and the direct approach based on Sagraloff's modified Descartes algorithm [39]. The algorithms were implemented in C as part of the core library of MATHEMATICA, and we illustrate their behavior on various datasets (Sec. 6). The complexity bounds that we present are many factors better than the previously known ones. However, a fair and explicit comparison with the bounds in [18] is rather difficult, if possible at all, since, besides the improved separation bounds that we present, the complexity bounds of many sub-algorithms that are used have been dramatically improved over the last 20 years, and it is not clear how to take this into account in the comparison. Nevertheless, the separation and complexity bounds of Sec. 5 are, to the best of our knowledge, new.

A preliminary version of our work appeared in [41]. In the current version we present improved bounds using fast algorithms for univariate real root isolation. We fix an error in the complexity of STURM solver and we analyze the complexity of solving all possible polynomials B_{α_k}. Finally, we give more details in our experimental section.

The rest of the paper is structured as follows: First we introduce our notations, and in Sec. 2 we present some preliminaries and known results that we will use throughout the paper. In Sec. 3 we present our first, indirect, approach for tackling Problem 1 and in Sec. 4 the two direct algorithms. In Sec. 5 we present new aggregate separation bounds. Finally, in Sec. 6 we present our implementation and experiments.

Notation $\mathcal{O}_B$ means bit complexity and the $\widetilde{\mathcal{O}}_B$-notation means that we are ignoring logarithmic factors. For $A = \sum_{i=1}^{d} a_i x^i \in \mathbb{Z}[x]$, $\mathsf{deg}(A)$ denotes its degree. $\mathcal{L}(A)$ denotes an upper bound on the bitsize of the coefficients of A, including a bit for the sign. For $\mathsf{a} \in \mathbb{Q}$, $\mathcal{L}(\mathsf{a}) \geq 1$ is the maximum bitsize of the numerator and the denominator.

Let $\alpha_1, \ldots, \alpha_d$ be the distinct complex roots of A. The Mahler's measure, e.g. [2, 25, 46], of A is $\mathcal{M}(A) = |a_d| \prod_{i=1}^{d} \max\{1, |\alpha_i|\}$. It holds $\mathcal{M}(A) \leq \|A\|_2$.

Let $\Delta_i = |\alpha_i - \alpha_{c_i}|$, where α_{c_i} is the root closest to α_i. $\Delta(A) = \min_i \Delta_i(A)$ is the separation bound, that is the smallest distance between two (real or complex, depending on the context) roots of A. By $\Sigma(A) = -\sum_{i=1}^{n} \lg \Delta_i(A)$, we denote an upper bound on the numbers of bits needed to represent isolating rational numbers for all the roots of A.

We represent a real algebraic number, α, by the *isolating interval* representation. It includes a square-free polynomial which vanishes on α and a (rational) interval containing α and no other root.

Given two polynomials, possibly multivariate, f and g, let $\mathtt{res}_x(f, g)$ denote their resultant with respect to x.

2 Preliminaries

Real algebraic numbers are the real roots of univariate polynomials with integer coefficients; let their set be $\mathbb{R}_{\mathsf{alg}}$. We represent them in the so-called *isolating interval representation*. If $\alpha \in \mathbb{R}_{\mathsf{alg}}$ then the representation consists of a square-free polynomial with integer coefficients, $A \in \mathbb{Z}[x]$, that has α as a real root, and an isolating interval with rational endpoints, $\mathcal{I} = [\mathsf{a}_1, \mathsf{a}_2]$, that contains α and no other root of the polynomial. We write $\alpha \cong (A, \mathcal{I})$.

The following proposition provides bounds on the roots of a univariate polynomial. Various versions of the proposition could be found in e.g. [8, 10, 42]. We present a version that appears in [19]. We refer the reader to [15] for a detailed treatment and an extension to the multivariate case.

Theorem 1. *Let $f = \sum_{i=0}^{d} a_i x^i \in \mathbb{R}[x]$ be such that $a_0 a_d \neq 0$, not necessarily square-free. Let the distinct roots of f be $\alpha_1, \ldots, \alpha_r$. For any nonzero root α_k it holds*

$$\frac{|a_0|}{2\,\|f\|_\infty} \leq |\alpha_k| \leq 2\frac{\|f\|_\infty}{|a_d|} \quad . \tag{1}$$

Let K be any subset of $\{1,\ldots,r\}$. It holds

$$\prod_{k\in K}\Delta_k \geq d^{-18d}\,\mathcal{M}\left(f\right)^{-15d}\,\left|\mathbf{sr}_{d-r}(f,f')\right|^3\ , \tag{2}$$

where $\mathbf{sr}_r(f,f')$ is the $(d-r)$-th subresultant coefficient of the subresultant sequence of f and its derivative f'. If f is square-free, then we can replace $\mathbf{sr}_0$ with the resultant of f and f', $\mathtt{res}(f,f')$.

If f has integer coefficients, then we have the following bounds:

Corollary 2. *With the above notation, if $f\in\mathbb{Z}[x]$, then let $\mathcal{L}\left(f\right)=\tau$. In this case Eq. (1) simplifies to*

$$2^{-\tau-1}\leq\frac{1}{2\left\|f\right\|_\infty}\leq\left|\alpha_k\right|\leq 2\left\|f\right\|_\infty\leq 2^{\tau+1}\ , \tag{3}$$

following Eq. (2) we get

$$\prod_{k\in K}\Delta_k\geq d^{-18d}\left\|f\right\|_2^{-15d}\Rightarrow-\lg\prod_{k\in K}\Delta_k\leq 15\,d\tau+33\,d\,\lg d=\widetilde{\mathcal{O}}(d\tau)\ .$$

Proposition 3. *Let $f\in\mathbb{Z}[x]$ have degree d and bitsize τ. We compute the isolating interval representation of its real roots and their multiplicities in $\widetilde{\mathcal{O}}_B(d^3\tau)$ [27, 39, 40] or $\widetilde{\mathcal{O}}_B(d^2\tau)$ [28]. The endpoints of (all) the isolating intervals have bitsize $\mathcal{O}(d\,\tau)$ and $\mathcal{L}\left(f_{red}\right)=\mathcal{O}(d+\tau)$, where f_{red} is the square-free part of f. If $N=\max\{d,\tau\}$ then complexity bounds become $\widetilde{\mathcal{O}}_B(N^4)$ and $\widetilde{\mathcal{O}}_B(N^3)$, respectively.*

Proposition 4. *[30, 32] Let $f\in\mathbb{Z}[x]$ have degree d and bitsize τ. We can compute isolating intervals for all the real roots of f with width less than or equal to 2^{-L} in $\widetilde{\mathcal{O}}_B(d^2\tau+dL)$.*

Proposition 5. *Given a real algebraic number $\alpha\cong(f,[\mathsf{a},\mathsf{b}])$, where $\mathsf{deg}(f)=d$, $\mathcal{L}\left(f\right)=\tau$, $\mathcal{L}\left(\mathsf{a}\right)=\mathcal{L}\left(\mathsf{b}\right)=\mathcal{O}(d\tau)$, and $g\in\mathbb{Z}[x]$, such that $\deg(g)=d$, $\mathcal{L}\left(g\right)=\tau$, we compute $\mathrm{sign}(g(\alpha))$ in bit complexity $\widetilde{\mathcal{O}}_B(d^2\tau)$.*

If we are given isolating intervals for all the roots of f, then we can compute the sign of g evaluated at all of the roots of f at the same Boolean cost.

Proof: Let h be the gcd of f and g. If α is a common root of f and g, then it is a root of h. We can compute h in $\widetilde{\mathcal{O}}_B(d^2\tau)$ and $\deg(h)=\mathcal{O}(d)$ and $\mathcal{L}\left(h\right)=\widetilde{\mathcal{O}}(d+\tau)$ [20].

Consider the polynomial $p=f\,g$. It holds $\deg(f)=\mathcal{O}(d)$, $\mathcal{L}\left(p\right)=\widetilde{\mathcal{O}}(\tau)$ and $-\lg\prod_k\Delta_k(p)=\widetilde{\mathcal{O}}(d\tau)$ (Prop. 2). This polynomial, p, has as roots the roots of f and the roots of g. Thus, its separation bound provides a bound on how close are the (non-common) roots of f and the roots of g.

If we refine the isolating interval, I, of α up to this accuracy, then we are certain that α is the only root of f and g that is contained in it. Using Prop. 4 we can do this $\widetilde{\mathcal{O}}_B(d^2\tau)$. Notice that we can refine all the roots of g up to this accuracy at the same cost (up to poly-logarithmic factors).

Next, we evaluate h, that represents the common roots of f and g, at the endpoints of this refined interval. Each evaluation costs $\widetilde{\mathcal{O}}_B(d^2\tau)$, using a divide and conquer approach [3, 16]. Then, one the following happens.

If α is a root of h, and thus a common root of f and g, then the two evaluations will have different signs. In this case $\mathrm{sgn}(f(\alpha))=0$.

If α is not a root of h, and thus not a root of f, then the two evaluations have the same sign. As there is no root of f in this refined interval, f has a constant sign at this interval. Therefore, the sign of the evaluations is the same as $\mathrm{sgn}(f(\alpha))$.

The (Boolean) cost of evaluating a polynomial at one point and the cost of evaluating at multiple, d, points is the same up to poly-logarithmic factors [29, 31]. Hence, the same bound holds if we want to compute the sign of f evaluate at *all* the roots of g.

□

For the proofs of the following results and for the related work we refer the reader to [9]. Let $f, g \in (\mathbb{Z}[x])[y]$ be such that $\deg_x(f) = p$, $\deg_x(g) = q$, $\deg_y(f), \deg_y(g) \leq d$, $\tau = \max(\mathcal{L}(f), \mathcal{L}(g))$. By $\mathbf{SR}(f, g\,;\, \mathsf{a})$ we denote the evaluation of the signed polynomial remainder sequence of f and g with respect to x over a, and by $\mathbf{SR}_j(f, g\,;\, \mathsf{a})$ we denote the j-th element in this sequence.

Proposition 6. *We can compute* $\mathtt{res}(f, g)$ *w.r.t.* x *or* y *in* $\widetilde{\mathcal{O}}_B(pq \max\{p, q\} d\tau)$.

Proposition 7. *We compute* $\mathbf{SR}(f, g\,;\, \mathsf{a})$, *where* $\mathsf{a} \in \mathbb{Q} \cup \{\infty\}$ *and* $\mathcal{L}(\mathsf{a}) = \sigma$, *in* $\widetilde{\mathcal{O}}_B(pq \max\{p, q\} d \max\{\tau, \sigma\})$. *For the polynomials* $\mathbf{SR}_j(f, g\,;\, \mathsf{a}) \in \mathbb{Z}[y]$, *except for* f, g, *we have* $\deg_y(\mathbf{SR}_j(f, g\,;\, \mathsf{a})) = \mathcal{O}((p + q)d)$ *and* $\mathcal{L}(\mathbf{SR}_j(f, g\,;\, \mathsf{a})) = \mathcal{O}(\max\{p, q\}\tau + \min\{p, q\}\sigma)$.

3 Reduction to integer coefficients

3.1 Some useful bounds

The roots of B_α in Problem 1 are algebraic numbers, hence they are roots of a polynomial with integer coefficients. We estimate bounds on the degree and the bitsize of this polynomial and we use them to analyze the Boolean complexity of the real root isolation algorithm.

Consider a real algebraic number $\alpha \in \mathbb{R}_{\mathsf{alg}}$, in isolating interval representation $\alpha \cong (A, \mathcal{I})$, where $A = \sum_{i=0}^{m} a_i\, x^i$, $\mathcal{I} = [\mathsf{a}_1, \mathsf{a}_2]$, $\mathsf{a}_{1,2} \in \mathbb{Q}$, $\mathsf{deg}(A) = m$, and $\mathcal{L}(A) = \tau$. Since A is square-free, it has m, possible complex, roots, say $\alpha_1, \alpha_2, \ldots, \alpha_m$ and after a (possible) reordering let $\alpha = \alpha_1$.

Let $B_\alpha \in \mathbb{Z}(\alpha)[y]$, be a univariate polynomial in y, with coefficients that are polynomials in α with integer coefficients. More formally, let $B_\alpha = \sum_{i=0}^{n} b_i(\alpha)\, y^i$, where $b_i(x) = \sum_{j=0}^{\eta_i} c_{ij}\, x^j$ and $\eta_i < m$, $0 \leq i \leq n$. The restriction $\eta_i < m$ comes from the fact that $\mathbb{Z}(\alpha)$ is a vector space of dimension[1] m and the elements of one of its bases are $1, \alpha, \ldots, \alpha^{m-1}$. Finally, let $\mathcal{L}(B_\alpha) = \max_{i,j} \mathcal{L}(c_{ij}) = \sigma$. We assume that B_α is square-free.

Our goal is to isolate the real roots of B_α (Problem 1). Since B_α has algebraic numbers as coefficients, its roots are algebraic numbers as well. Moreover, there is a polynomial with integer coefficients that has as roots the roots of B_α, and possible other roots as well. To construct this polynomial, e.g. [8, 18, 21], we consider the following resultant w.r.t. x

$$R(y) = \mathtt{res}_x(A(x), B(x, y)) = a_m^\eta \prod_{j=1}^{m} B(\alpha_j, y), \tag{4}$$

where $\eta = \max\{\eta_i\}$, and $B(x, y) \in \mathbb{Z}[x, y]$ is obtained from B_α after replacing all the occurrences of α with x. Interpreting the resultant using the Poisson formula, $R(y)$ is the product of polynomials $B(\alpha_j, y)$, where j ranges over all the roots of A. Our polynomial $B_\alpha \in \mathbb{Z}(\alpha)[y]$ is the factor in this product for $j = 1$. Hence, R has all the roots that B_α has and maybe more.

Remark 8. *Notice that* $R(y)$ *is not square-free in general. For example consider the polynomial* $B_\alpha = y^4 - \alpha^2$, *where* α *is the positive root of* $A = x^2 - 3$. *In this case* $R(y) = \mathtt{res}_x(A(x), B(x, y)) = \mathtt{res}_x(x^2 - 3, y^2 - x^2) = (y^4 - 3)^2$.

Using Prop. 16 and by taking into account that $\eta_i < m$, we get $\deg(R) \leq mn$ and $\mathcal{L}(R) \leq m(\tau + \sigma) + 2m \lg(4mn)$. We may also write $\deg(R) = \mathcal{O}(mn)$ and $\mathcal{L}(R) = \widetilde{\mathcal{O}}(m(\sigma + \tau))$.

To construct an isolating interval representation for the real roots of B_α, we need a square-free polynomial. This polynomial, $C(y) \in \mathbb{Z}[y]$, is a square factor of $R(y)$, and so it holds $\mathsf{deg}(C) \leq mn$ and $\mathcal{L}(C) \leq m(\tau + \sigma) + 3m \lg(4mn)$, where the last inequality follows from Mignotte's bound [25].

Using the Prop. 1, we deduce the following lemma:

[1] If A is the minimal polynomial of α then the dimension is exactly m. In general, it is not (computationally) easy to compute the minimal polynomial of a real algebraic number. Thus, we usually work with a square-free polynomial that has it as real root.

Lemma 9. *Let B_α be as in Problem 1. The minimal polynomial, $C \in \mathbb{Z}[x]$, of the, possible complex, roots of B_α, γ_i, has degree $\leq mn$ and bitsize $\leq m(\tau+\sigma) + 3m\lg(4mn))$ or $\widetilde{\mathcal{O}}(m(\tau+\sigma))$. Moreover, it holds*

$$|\gamma_i| \leq 2^{\widetilde{\mathcal{O}}(m(\tau+\sigma))} , \quad (5)$$

$$\Sigma(C) = -\sum_i \lg \Delta_i(C) \leq \widetilde{\mathcal{O}}(m^2 n(n+\tau+\sigma)) . \quad (6)$$

3.2 The algorithm

The indirect algorithm for Problem 1 follows closely the procedure described in the previous section to estimate the various bounds on the roots of B_α. First, we compute the univariate polynomial with integer coefficients, R, such that the set of its real roots includes those of B_α. We isolate the real roots of R and we identify the ones that are also roots of B_α.

Let us present in detail the three steps and their complexity. We compute R using resultant computation, as presented in (4). For this we consider B as a bivariate polynomial in $\mathbb{Z}[x,y]$ and we compute $\mathtt{res}_x(B(x,y), A(x))$, using Prop. 6. Since $\deg_x(B) < m$, $\deg_y(B) = n$, $\mathcal{L}(B) = \sigma$, $\deg_x(A) = m$, $\deg_y(A) = 0$, and $\mathcal{L}(A) = \tau$, this computation costs $\widetilde{\mathcal{O}}_B(m^3 n(\sigma+\tau))$, using Prop. 6.

Now we isolate the real roots of R. This can be done in $\widetilde{\mathcal{O}}_B(m^4n^3(\tau+\sigma))$, resp. $\widetilde{\mathcal{O}}_B(m^3n^2(\tau+\sigma))$ using the first, resp. second, bound from Prop. 3. In the same complexity bound we can also compute the multiplicities of the real roots, if needed [14].

The rational numbers that isolate the real roots of R have bitsize bounded by $\widetilde{\mathcal{O}}(m^2n(\sigma+\tau))$, and the bitsize of all of them is the somewhat similar quantity $\widetilde{\mathcal{O}}(m^2n(n+\sigma+\tau))$, as Prop. 1 and Lem. 9 indicate.

It is possible that R has more roots that B_α. Thus, it remains to identify which real roots of R are roots of B_α. For sure all the real roots of B_α are roots of R. Consider a real root γ of R and its isolating interval $[\mathsf{c}_1, \mathsf{c}_2]$. If γ is a root of B_α, then since B_α is square-free, by Rolle's theorem it must change signs if we evaluate it over the endpoints of the isolating interval of γ. Therefore, to identify the real roots of R that are roots of B_α it suffices to compute the sign of B_α over all the endpoints of the isolating intervals.

It is possible to avoid the non-relevant roots of R by applying the algorithm for changing the ordering of a bivariate regular chain [33]. However, we do not elaborate this approach further.

Consider an isolating point of R, say $\mathsf{c}_\mathsf{j} \in \mathbb{Q}$, of bitsize s_j. To compute the sign of the evaluation of B_α over it, we proceed as follows. First we perform the substitution $y = \mathsf{c}_j$, and after clearing denominators, we get a number in $\mathbb{Z}[\alpha]$, for which we want to compute its sign. This is equivalent to consider the univariate polynomial $B(x, \mathsf{c}_j)$ and to compute its sign if we evaluate it over the real algebraic number α. We have $\deg(B(x,\mathsf{c}_j)) = \mathcal{O}(m)$ and $\mathcal{L}(B(x,\mathsf{c}_j)) = \widetilde{\mathcal{O}}(\sigma + ns_j)$. The computation of each coefficient of $B(x,\mathsf{c}_j)$ costs $\widetilde{\mathcal{O}}_B(\sigma + ns_j)$ and all of them cost $\widetilde{\mathcal{O}}_B(m\sigma + mns_j)$.

The sign evaluation costs $\widetilde{\mathcal{O}}_B(m^2(\tau+\sigma+ns_j)$ using Prop. 5. Summing up over all s_j's, there are $\mathcal{O}(mn)$, and taking into account that $\sum_j s_j = \widetilde{\mathcal{O}}(m^2n(\sigma+\tau+n))$ (Lem. 9), we conclude that the overall complexity of identifying the real roots of B_α is $\widetilde{\mathcal{O}}_B(m^4n^2(n+\sigma+\tau))$, or $\widetilde{\mathcal{O}}_B(N^7)$.

The overall complexity of the algorithm is dominated by that of real solving. We can state the following theorem:

Theorem 10. *The complexity of isolating the real roots of $B \in \mathbb{Z}(\alpha)[y]$ using the indirect method is $\widetilde{\mathcal{O}}_B(m^4n^3(\tau+\sigma))$, resp. $\widetilde{\mathcal{O}}_B(m^4n^2(n+\sigma+\tau))$, using the first, resp. the second bound of Prop. 3. If $N = \max\{m,n,\sigma,\tau\}$, then the previous bounds become $\widetilde{\mathcal{O}}_B(N^8)$ and $\widetilde{\mathcal{O}}_B(N^7)$, respectively.*

If the polynomial B_α is not square-free then we can apply, for example, the algorithm of [43] to compute its square-free factorization and then we apply the previous algorithm either to the square-free part or to each polynomial of the square-free factorization.

4 Two direct approaches

The computation of R, the polynomial with integer coefficients that has the real roots of B_α is a costly operation that we usually want to avoid. If possible, we would like to try to solve the polynomial B_α directly, using one of the well-known subdivision algorithms, for example STRUM or DESCARTES and BERNSTEIN, specially adopted to handle polynomials that have coefficients in an extension field. In practice, this is accomplished by obtaining, repeatedly improved, approximations of the real algebraic number α and subsequently apply DESCARTES or BERNSTEIN for polynomials with interval coefficients, e.g. [17, 35].

The fact that we compute the roots using directly the representation of B_α allows us to avoid the complexity induced by the conjugates of α. This leads to improved separation bounds, and to faster algorithms for real root isolation.

4.1 Separation bounds for B_α

We compute various bounds on the roots of B_α based on the inequalities of Thm. 1. For this we need to compute a lower bound for $|R_B(\alpha)|$ and upper bounds for $|b_i(\alpha)|$ and $\|B_\alpha\|_2$.

First we compute bounds on the coefficients of B_α. Let $\alpha_1 = \alpha, \alpha_2, \ldots, \alpha_m$ be the roots of A. We consider the resultants

$$r_i := \mathtt{res}_x(A(x), z - b_i(x)) = \mathtt{res}_x\left(A(x), z - \sum_{j=0}^{\eta_i} c_{i,j} x^j\right) \in \mathbb{Z}[z] \ .$$

It holds that

$$r_i(z) = a_m^{\eta_i} \prod_{k=1}^{m} (z - b_i(\alpha_k)) \ , \tag{7}$$

where $\eta_i < m$. The roots of r_i are the numbers $b_i(\alpha_k)$, where k runs over all the roots of A. We use Prop. 16 to bound the degree and bitsize of r_i. The degree of r_i is bounded by m and their coefficients are of bitsize $\leq m\sigma + m\tau + 5m\lg(m)$. Using Cauchy's bound, we deduce that either $b_i(\alpha_k) = 0$ or

$$2^{-T} = 2^{-m\sigma - m\tau - 5m\lg(m)} \leq |b_i(\alpha_k)| \leq 2^{m\sigma + m\tau + 5m\lg(m)} = 2^T \ , \tag{8}$$

for all i and k. Note that since b_n and A are relatively prime, $b_n(\alpha_k) \neq 0$. To bound $|\mathbf{sr}_0|$ that appears in Thm. 1 we consider the identity

$$|\mathbf{sr}_0| = |(-1)^{\frac{1}{2}n(n-1)} \mathtt{res}_y(B_\alpha, \partial B_\alpha(y)/\partial y)| = |(-1)^{\frac{1}{2}n(n-1)} R_B(\alpha)| \ ,$$

where the resultant, $R_B \in \mathbb{Z}[\alpha]$, can be computed as the determinant of the Sylvester matrix of B_α and $\partial B_\alpha(y)/\partial y$.

The Sylvester matrix is of size $(2n-1) \times (2n-1)$, the elements of which belong to $\mathbb{Z}[\alpha]$. The determinant consists of $(2n-1)!$ terms. Each term is a product of $n-1$ polynomials in α of degree at most $m-1$ and bitsize at most σ, times a product of n polynomials in α of degree at most $m-1$ and bitsize at most $\sigma + \lg n$. The first product results a polynomial of degree $(n-1)(m-1)$ and bitsize $(n-1)\sigma + (n-1)\lg m$. The second product results polynomials of degree $n(m-1)$ and bitsize $n\sigma \lg n + n \lg m$. Thus, any term in the determinant expansion is a polynomial in α of degree at most $(2n-1)(m-1)$, or $\mathcal{O}(mn)$, and bitsize at most $4(2n-1)\sigma \lg(mn)$ or $\widetilde{\mathcal{O}}(n\sigma)$. The determinant itself, is a polynomial in α of degree at most mn and of bitsize $4(2n-1)\sigma \lg(mn) + (2n-1)\lg(2n-1) \leq 5(2n-1)\sigma \lg(mn) = \widetilde{\mathcal{O}}(n\sigma)$.

To compute a bound on $R_B(\alpha)$ we consider R_B as a polynomial in $\mathbb{Z}[y]$, and we compute a bound on its evaluation over α. For this we use resultants. It holds

$$D = \mathtt{res}_x(A(x), y - R_B(x)) = a_m^{\deg(R_B)} \prod_{i=1}^{m} (y - R_B(\alpha_i)) \ . \tag{9}$$

We notice that the roots of $D \in \mathbb{Z}[x]$ are the evaluations of R_B over the roots of A. So it suffices to compute bounds on the roots of D. Using Prop. 16 we deduce that $\deg(D) \leq m$ and $\mathcal{L}(D) \leq 13mn\sigma \lg(mn) + mn\tau$ or $\mathcal{L}(D) = \widetilde{\mathcal{O}}(mn(\sigma+\tau))$. We use Eq. (1) to compute an upper and lower bound for $|R_B(\alpha)|$, and so

$$2^{-\mathcal{O}(mn\tau+mn\sigma \lg(mn))} \leq |R_B(\alpha)| \leq 2^{\mathcal{O}(mn\tau+mn\sigma \lg(mn))} \ . \tag{10}$$

It remains to bound $\mathcal{M}(B_\alpha)$ using $\|B_\alpha\|_2$. From Eq. (8) we get

$$\|B_\alpha\|_2^2 \leq \sum_{i=0}^{n} (b_i(\alpha))^2 \leq (n+1)\, 2^{2m(\sigma+\tau+5\lg(m))} \ . \tag{11}$$

We substitute Eq. (8), (10), and (11) to the inequalities of Thm. 1, and we derive the following lemma:

Lemma 11. *Let B_α be as in Problem 1, and ξ_i be its roots. Then, it holds*

$$|\xi_i| \leq 2^{\widetilde{\mathcal{O}}(m(\sigma+\tau))} \ , \tag{12}$$

$$\Sigma(B_\alpha) = -\sum_i \lg \Delta_i(B_\alpha) \leq \widetilde{\mathcal{O}}(mn(\sigma+\tau)) \ . \tag{13}$$

4.2 Almost tight separation bounds

The separation bounds of the previous lemma are close to optimal.

Let α be the root of $A(x) = x^m - ax^{m-1} - 1$, in $(a, a+1)$, for $a \geq 3$, $m \geq 3$. Then the Mignotte-like polynomial

$$B_\alpha(y) = y^n - 2(\alpha^k y - 1)^2,$$

where $k = \lfloor (m-1)/2 \rfloor$, has two roots in $(1/\alpha^k - h, 1/\alpha^k + h)$, where

$$h = \alpha^{-k(n+2)/2} < a^{-(m-2)(n+2)/4}.$$

If $a \leq 2^\tau$ and $\tau = \Omega(\lg(mn))$, then $-\lg \Delta(B_\alpha) = \Omega(mn\tau)$, which matches the upper bound in (13) of Lem. 11. This quantity, $\Omega(mn\tau)$, is also a tight lower bound for the number of steps that a bisection-based algorithm, like STURM or DESCARTES, performs. To establish the lower bound on the number of steps it suffices to reproduce the arguments from [13] made for polynomials with integer coefficients.

4.3 The STURM algorithm

Let us first study the STURM algorithm. We assume B_α as in Problem 1 to be square-free. To isolate the real roots of B_α using the STURM algorithm, we need to evaluate the Sturm sequence of $B(\alpha, y)$ and its derivative with respect to y, $\partial B(\alpha, y)/\partial y$, over various rational numbers. For the various bounds needed we will use Lem. 11.

The number of steps that a subdivision-based algorithm, and hence STURM algorithm, performs to isolate the real roots of a polynomial depends on the separation bound. To be more specific, the number of steps, $(\#T)$, that STURM performs is $(\#T) \leq 2r + r \lg \mathsf{B} + \Sigma(B_\alpha)$ [8, 10], where r is the number of real roots and B is an upper bound on the real roots. Using (12) and (13) we deduce that $(\#T) = \widetilde{\mathcal{O}}(mn(\tau+\sigma))$.

To complete the analysis of the algorithm it remains to compute the complexity of each step, i.e. the cost of evaluating the Sturm sequence over a rational number, of the worst possible bitsize. The latter is induced by the separation bound, and in our case is $\widetilde{\mathcal{O}}(mn(\tau+\sigma))$.

We consider B as polynomial in $\mathbb{Z}[x, y]$ and we evaluate the Sturm-Habicht sequence of B and $\frac{\partial B}{\partial y}$, over rational numbers of bitsize $\widetilde{\mathcal{O}}(mn(\tau+\sigma))$. The cost of this operation is $\widetilde{\mathcal{O}}_B(m^2n^4(\tau+\sigma))$ (Prop. 7).

It produces $\mathcal{O}(n)$ polynomials in $\mathbb{Z}[x]$, of degrees $\mathcal{O}(mn)$ and bitsize $\widetilde{\mathcal{O}}(mn^2(\sigma+\tau))$. For each polynomial we have to compute its sign if we evaluate it over α. Using Prop. 5 each sign evaluation costs $\widetilde{\mathcal{O}}_B(m^3n^4(\sigma+\tau))$, and so the overall cost is $\widetilde{\mathcal{O}}_B(m^3n^5(\sigma+\tau))$. If we multiply the latter bound with the number of steps, $\widetilde{\mathcal{O}}(m^4n^6(\sigma+\tau)^2)$, we get the following theorem.

Theorem 12. *The complexity of isolating the real roots of* $B \in \mathbb{Z}(\alpha)[y]$ *using the* STURM *algorithm is* $\widetilde{\mathcal{O}}_B(m^4n^6(\sigma+\tau)^2))$, *or* $\widetilde{\mathcal{O}}_B(N^{12})$, *where* $N = \max\{m, n, \sigma, \tau\}$.

4.4 A modified DESCARTES algorithm

The main idea of the modified version of DESCARTES algorithm is to approximate the coefficients of B_α up to a specified accuracy that guarantees that the roots of the resulting polynomial are close to the roots of B_α. Then, we solve the derived univariate polynomial. Finally, from the isolating intervals of the derived polynomial we can obtain isolating intervals for the real roots of B_α.

We have implemented the modified version of Descartes' algorithm due to Sagraloff [38], that applies to polynomials with bitstream coefficients. We also refer the reader to [12, 23]. For the analysis of the various bounds we rely on [24], see also [23, 38, 40], where we also refer the reader for a detailed presentation.

We need to approximate the coefficients of B_α up to accuracy $\mathcal{O}(\Sigma(B_\alpha) + n\tau_B) = \widetilde{\mathcal{O}}(mn(\sigma+\tau))$ [38], see also [23, 28, 40]. In this way the number of real roots of the approximate polynomial is the same as the number of real roots of B_α. Moreover, the isolating intervals that we compute for the approximate polynomial are also isolating intervals for the real roots of B_α.

As stated in Problem 1, let α be a real root of $A = \sum_{i=0}^{m} a_i x^i \in \mathbb{Z}[x]$, where $a_m \neq 0$ and $|a_i| < 2^\tau$ for $0 \leq i \leq m$, and let $B_\alpha = \sum_{i=0}^{n} b_i(\alpha) y^i \in \mathbb{Z}[\alpha][y]$, where $b_i = \sum_{j=0}^{\eta_i} c_{i,j} x^j \in \mathbb{Z}[x]$, $\eta_i < m$ and $|c_{i,j}| < 2^\sigma$ for $0 \leq i \leq n$ and $0 \leq j \leq \eta_i$. Recall that we also assume that B_α is square-free.

Let $\xi_1, \ldots, \xi_n$ be all the (complex) roots of B, and $\Delta_i(B_\alpha) := \min_{j \neq i} |\xi_j - \xi_i|$. Let ρ be such that $\rho = \max_j\{1, \max\{1, |\log|\xi_i||\}\}$, that is a logarithmic root bound for the roots of B_α. We need to approximate the coefficients of B_α up to accuracy $\mathcal{O}(\Sigma(B_\alpha) + n\rho)$, where $\Sigma(B_\alpha) = -\sum_{i=1}^{n} \lg(\Delta_i(B_\alpha))$. In this way the complexity of isolating the real roots would be $\widetilde{\mathcal{O}}_B(n^3 + n^2\tau_B + n\Sigma(B_\alpha))$, where $\left|\frac{b_i(\alpha)}{b_n(\alpha)}\right| \leq 2^{\tau_B}$, based on Proposition 3 and [24].

Let us estimate the various quantities. Lemma 11 indicates $\Sigma(B_\alpha) \leq \widetilde{\mathcal{O}}(mn(\tau+\sigma))$. To compute a bound on τ_B, we use Eq. (8). It holds $\left|\frac{b_i(\alpha_k)}{b_n(\alpha_k)}\right| \leq 2^{2m\sigma + 2m\tau + 10m\lg(m)}$, for all i and k. Hence,

$$\tau_B \leq 2m\sigma + 2m\tau + 10m\lg(m) = \widetilde{\mathcal{O}}(m(\sigma+\tau)) \ . \tag{14}$$

A similar bound holds for ρ using Eq. (1) of Thm. 1.

By combining the previous bounds we deduce that we should approximate the coefficients of B_α up to accuracy

$$\widetilde{\mathcal{O}}_B(\Sigma(B_\alpha) + n\rho) = \widetilde{\mathcal{O}}_B(mn(\tau+\sigma)) \ .$$

Thus, we can isolate the real roots of B_α in $\widetilde{\mathcal{O}}_B(n^3 + mn^2(\sigma+\tau))$. If $N = \max\{m, n, \sigma, \tau\}$, then the bound becomes $\widetilde{\mathcal{O}}_B(N^4)$.

It remains to estimate the cost of computing the successive approximations of $b_i(\alpha)/b_n(\alpha)$. The root isolation algorithm requires approximations of $b_i(\alpha)/b_n(\alpha)$ to accuracy of $\mathcal{O}(\Sigma(B_\alpha) + n\tau_B)$ bits after the binary point. Since $|b_i(\alpha)/b_n(\alpha)| \leq 2^{2T}$, to approximate each fraction, for $0 \leq i \leq n-1$, to accuracy L, it is sufficient to approximate $b_i(\alpha)$, for $0 \leq i \leq n$, up to precision $\mathcal{O}(L+T)$. For the definition of T refer to Eq. (8). Hence, the algorithm requires approximation of $b_i(\alpha)$, for $0 \leq i \leq n$, to precision $\mathcal{O}(\Sigma(B_\alpha)+n\tau_B+T)$.

Approximation of $c_{i,j}\alpha^j$ to accuracy of L bits requires approximation of α to accuracy of

$$L + \lg|c_{i,j}| + \lg(j) + (j-1)\lg|\alpha| \leq L + \sigma + \lg(m) + (m-1)(\tau+1) = \widetilde{\mathcal{O}}(L + \sigma + m\tau)$$

bits. Hence the accuracy of approximations of α required by the algorithm is

$$\mathcal{O}(\Sigma(B_\alpha) + n\tau_B + T) = \widetilde{\mathcal{O}}(mn(\sigma+\tau)) \ .$$

Using Proposition 4, the bit complexity of approximating all the roots of A, and hence α, to accuracy L is $\widetilde{\mathcal{O}}(m^2\tau + mL)$. Therefore, the bit complexity of computing the required approximations of $b_i(\alpha)/b_n(\alpha)$ is

$$\widetilde{\mathcal{O}}(m^2\tau + m \cdot mn(\sigma+\tau)) = \widetilde{\mathcal{O}}(m^2n(\sigma+\tau)) \ .$$

To obtain the approximations of $b_i(\alpha)$, for all i, we need to evaluate n polynomials, the b_i's, of degree at most m each at the approximation of α that has bitsize $\widetilde{\mathcal{O}}(mn(\sigma+\tau))$. Each evaluation costs $\widetilde{\mathcal{O}}(m^2n(\sigma+\tau))$ [3, 16] and all them cost $\widetilde{\mathcal{O}}(m^2n^2(\sigma+\tau))$ or $\widetilde{\mathcal{O}}_B(N^5)$.

By combining the various bounds we obtain the claimed complexity bound.

Theorem 13. *The bit complexity of isolating the real roots of B_α of Problem 1 is $\widetilde{\mathcal{O}}(n^3+m^2n^2(\sigma+\tau))$. If $N=\max\{m,n,\sigma,\tau\}$, then the bound becomes $\widetilde{\mathcal{O}}_B(N^5)$.*

5 An application: solving all the polynomials

In this section we assume that B_α is square-free for all roots, α, of A. Let $\mathtt{r}\leq m=\deg(A)$ be the number of real roots of A. We consider an aggregate version of the bound (6) for all polynomials B_{α_k}, where α_k runs over all real roots of A, that is $1\leq k\leq \mathtt{r}$. Consequently, we present a complexity bound for isolating the real roots of all possible polynomials B_{α_k}.

Lemma 14. *Let B_{α_k} be as in Problem 1, and let k run over all the real roots of A, $1\leq k\leq \mathtt{r}$. Moreover, let $\tau_k:=\max_{0\leq i\leq n}\lg\left|\frac{b_i(\alpha_k)}{b_n(\alpha_k)}\right|$, and $i_k=\arg\max_{0\leq i\leq n}\lg\left|\frac{b_i(\alpha_k)}{b_n(\alpha_k)}\right|$. Then, it holds*

$$\sum_{k=1}^{\mathtt{r}}\lg\left|\frac{b_{i_k}(\alpha_k)}{b_n(\alpha_k)}\right|=\sum_{k=1}^{\mathtt{r}}\tau_k\leq 2m\sigma+2m\tau+9m\lg(m)=\widetilde{\mathcal{O}}(m(\sigma+\tau))\ , \tag{15}$$

$$-\sum_{k=1}^{\mathtt{r}}\lg\prod_i\Delta_i(B_{\alpha_k})\leq=\widetilde{\mathcal{O}}(mn(\sigma+\tau))\ . \tag{16}$$

Proof: Note that for $0\leq i\leq n$ and $1\leq k\leq \mathtt{r}$

$$|b_i(\alpha_k)|\leq m\,2^\sigma\max\{1,|\alpha_k|\}^{m-1}\ . \tag{17}$$

Hence

$$\prod_{k=1}^{\mathtt{r}}|b_{i_k}(\alpha_k)|\leq\prod_{k=1}^{\mathtt{r}}m\,2^\sigma\max\{1,|\alpha_k|\}^{m-1}\leq\prod_{\ell=1}^{m}m\,2^\sigma\max\{1,|\alpha_\ell|\}^{m-1}=m^m\,2^{m\sigma}\left(\frac{\mathcal{M}(A)}{|a_m|}\right)^{m-1}\ , \tag{18}$$

where $\mathcal{M}(A)=|a_m|\prod_{\ell=1}^m\max\{1,|\alpha_\ell|\}$ is the Mahler measure of A. By Landau's inequality, e.g. [25, 46],

$$\mathcal{M}(A)\leq\|A\|_2\leq\sqrt{m+1}\,\|A\|_\infty\leq\sqrt{m+1}\cdot 2^\tau\ . \tag{19}$$

Since $|a_m|\geq 1$,

$$\prod_{k=1}^{\mathtt{r}}|b_{i_k}(\alpha_k)|\leq m^m\,2^{m\sigma}(m+1)^{(m-1)/2}2^{(m-1)\tau}\leq 2^{m\sigma+m\tau+2m\lg(m)}\ . \tag{20}$$

Let $r_i(z)$, for $0\leq i\leq n$, be the resultants considered in (7) and let $s(z):=z^m r_n(1/z)$. Since b_n and A are relatively prime, $b_n(\alpha_\ell)\neq 0$ for $1\leq\ell\leq m$, and the roots of $s(z)$ are the numbers $1/b_n(\alpha_\ell)$. We have

$$\prod_{k=1}^{\mathtt{r}}|1/b_n(\alpha_k)|\leq\prod_{\ell=1}^{m}\max\{1,|1/b_n(\alpha_\ell)|\}=\mathcal{M}(s)/|s_m|\leq\mathcal{M}(s)\ ,$$

where $s_m\in\mathbb{Z}$ is the leading coefficient of s. Since $\mathcal{L}(s)=\mathcal{L}(r_n)\leq m\sigma+m\tau+5m\lg(m)$, by Landau's inequality we obtain

$$\prod_{k=1}^{\mathtt{r}}|b_n(\alpha_k)|\geq 1/(\sqrt{m+1}\,\|s\|_\infty)\geq 2^{-m\sigma-m\tau-7m\lg(m)}\ . \tag{21}$$

If we combine (20) and (21) we prove (15).

To prove (16) we use inequality (2). Therefore, we need to find an upper bound on $\prod_{k=1}^{\mathtt{r}}\|B_{\alpha_k}\|_2$ to bound $\mathcal{M}\left(B_{\alpha_k}\right)$ and a lower bound for $\mathbf{sr}_0$. For the first quantity we notice that (17) implies

$$\|B_{\alpha_k}\|_2 = \sqrt{\sum_{i=0}^{n}|b_i(\alpha_k)|^2} \leq \sqrt{n+1}\cdot m\, 2^{\sigma}\max\{1,|\alpha_k|\}^{m-1} \ ,$$

and hence

$$\prod_{k=1}^{\mathtt{r}}\|B_{\alpha_k}\|_2 \leq (n+1)^{m/2}m^m\, 2^{m\sigma}\prod_{k=1}^{\mathtt{r}}\max\{1,|\alpha_k|\}^{m-1} \leq (n+1)^{m/2}m^m\, 2^{m\sigma}\mathcal{M}\left(A\right)^{m-1} \ .$$

By (19)

$$\prod_{k=1}^{\mathtt{r}}\|B_{\alpha_k}\|_2 \leq 2^{m\sigma+m\tau+m\lg(n)+2m\lg(m)} \ . \tag{22}$$

To bound $\mathbf{sr}_0$ we need to bound the resultant $R_B(\alpha_k)$. We consider $E(z) := z^m D(1/z)$, where D is the resultant defined in (9). Since B_{α_ℓ} are square-free for $1 \leq \ell \leq m$, the roots of E are the numbers $1/R_B(\alpha_\ell)$. Taking into account that the leading coefficient of E is a nonzero integer, $\mathcal{L}\left(E\right) \leq 13mn\sigma\lg(mn) + mn\tau$, and using Landau's inequality we get that

$$\prod_{k=1}^{\mathtt{r}}|1/R_B(\alpha_k)| \leq \prod_{\ell=1}^{m}\max\{1,|1/R_B(\alpha_\ell)|\} \leq \mathcal{M}\left(E\right) \leq 2^{14mn\sigma\lg(mn)+mn\tau} \ . \tag{23}$$

Using $\mathtt{r}$ times (2) and summing up all inequalities we get

$$\begin{aligned}
-\sum_{k=1}^{\mathtt{r}}\lg\prod_i\Delta_i(B_{\alpha_k}) &\leq \sum_{k=1}^{\mathtt{r}}18n\lg n + 15n\lg\|B_{\alpha_k}\|_2 - 3\lg|R_B(\alpha_k)| \\
&\leq 18mn\lg n + 15n\lg\prod_{k=1}^{\mathtt{r}}\|B_{\alpha_k}\|_2 - 3\lg\prod_{k=1}^{\mathtt{r}}\lg|R_B(\alpha_k)| \\
&\leq \mathcal{O}(mn\lg n) + \mathcal{O}(nm(\sigma+\tau+\lg(mn))) + \mathcal{O}(mn(\sigma\lg(mn)+\tau)) \\
&\leq \mathcal{O}(mn(\sigma\lg(mn)+\tau)) = \widetilde{\mathcal{O}}(mn(\sigma+\tau)) \ ,
\end{aligned}$$

that concludes the proof. □

Theorem 15. *Let B_{α_k} be as in Problem 1 and let α_k run over all the roots of A. We can isolate the real roots of all B_{α_k} in $\widetilde{\mathcal{O}}_B(mn^3 + m^2n^2(\sigma+\tau))$. If $N = \max\{m,n,\sigma,\tau\}$, then the bound becomes $\widetilde{\mathcal{O}}_B(N^5)$,*

Proof: The proof is similar to the proof of Th. 13. For each (real) root α_k we need to approximate the coefficients of each B_{α_k} up to precision $\mathcal{O}(\Sigma(B_{\alpha_k}) + n\rho_k)$, where ρ_k is the logarithmic bound on the roots of B_{α_k} (see also the discussion in Section 4.4). We solve each polynomial in $\widetilde{\mathcal{O}}_B(n^3 + n\Sigma(B_{\alpha_k}) + n^2\tau_k)$, where τ_k is roughly an upper bound on the roots and is defined in Lemma 14. For all the roots the cost is $\widetilde{\mathcal{O}}_B(\sum_{k=1}^{\mathtt{r}}n^3 + \sum_{k=1}^{\mathtt{r}}n\Sigma(B_{\alpha_k}) + \sum_{k=1}^{\mathtt{r}}n^2\tau_k)$, where $\mathtt{r} \leq m$.

The first term of this bound is $\widetilde{\mathcal{O}}_B(mn^3)$. The second term, using (16) from Lemma 14, becomes $\widetilde{\mathcal{O}}_B(mn^2(\sigma+\tau))$. The same bound holds for the third term, using (15) from Lemma 14.

We can derive bounds for ρ_k similar to the ones for τ_k. These bounds affect the complexity of approximating the roots α_k up to the required precision. However, this complexity is dominated from the cost of obtaining the approximation of the coefficients $b_i(\alpha_k)$.

The cost of approximating one root of A up to a desired precision is the same as the cost of approximating all the roots [30, 32]. It is $\widetilde{\mathcal{O}}_B(m^2n(\sigma+\tau))$.

Recall that the complexity of Theorem 13 is dominated by the complexity of evaluating the polynomials b_i at approximations of the root(s) α_k. However, the Boolean complexity of the evaluation at one point is the same, up to poly-logarithmic factors, as the complexity of evaluating at many points [29, 31]. Therefore, the complexity of this step is also $\widetilde{\mathcal{O}}_B(m^2n^2(\sigma+\tau))$. □

6 Implementation and experiments

We compare implementations of two methods of real root isolation for square-free polynomials over simple algebraic extensions of rationals. The experiments have been run on a 64-bit Linux virtual machine with a 3 GHz Intel Core i7 processor and 6 GB of RAM. The timings are in given seconds. Computations that did not finish in 10 hours of CPU time are reported as > 36000, computations that took less that 1 millisecond are reported as 0.

Reduction to integer coefficients

The first method, *ICF* (for Integer Continued Fractions), performs reduction to integer coefficients described in Section 3.2. The algorithm consists of three steps. The first step computes the resultant (4). The reported resultant computation time is the fastest of three methods implemented in the MATHEMATICA system: subresultant remainder sequence over the integers, Sylvester matrix determinant and a modular resultant algorithm. Next the algorithm isolates the real roots of the resultant. The method used is the MATHEMATICA implementation of the Continued Fractions algorithm [1]. Finally, the intervals isolating roots of the original polynomial are identified using polynomial sign computation at the interval endpoints.

A modified Descartes' algorithm

The second method, *BMD* (for Bitstream Modified Descartes), uses Sagraloff's modified version of Descartes' algorithm ([38], see Section 4.4). The algorithm has been implemented in C as a part of the MATHEMATICA system. Our implementation uses a heuristically determined initial accuracy of $L = \max(2^{\lfloor \log_2 n \rfloor + 3}, 64)$ bits for the DCM^L algorithm and the accuracy is doubled each time the algorithm returns "insufficient precision".

Example sets used in the experiments

Randomly generated polynomials For given values of m and n each problem was generated as follows. First, univariate polynomials of degree m with uniformly distributed random 10-bit integer coefficients were generated until an irreducible polynomial which had real roots was obtained. A real root r of the polynomial was randomly selected as the extension generator. Finally, a polynomial in $\mathbb{Z}[r, y]$ of degree n in y and degree $m-1$ in r with 10-bit random integer coefficients was generated. Each timing reported is an average for 10 randomly generated problems.

Generalized Laguerre Polynomials This example compares the two root isolation methods for generalized Laguerre polynomials $L_n^\alpha(x)$, where α was chosen to be the smallest root of the Laguerre polynomial $L_m(x)$. Note that $L_n^\alpha(x)$ has n positive roots for any positive α and $L_m(x)$ has m positive roots, so this example maximizes the number of real roots of both the input polynomial with algebraic number coefficients and the polynomial with integer coefficients obtained by *ICF*.

Generalized Wilkinson Polynomials This example uses the following generalized Wilkinson polynomials

$$W_{n,\alpha}(x) := \prod_{k=1}^{n} (x - k\alpha)$$

where α is the smallest root of the Laguerre polynomial $L_m(x)$.

Mignotte Polynomials The variant of Mignotte polynomials used in this example is given by

$$M_{n,\alpha}(x) := y^n - 2(\alpha^k y - 1)^2$$

where α is the root of

$$A_m(x) := x^m - 3x^{m-1} - 1$$

in $(3, 4)$, $m \geq 3$ and $k = \lfloor (m-1)/2 \rfloor$ (see Section 4.2).

Experimental results

Tables 1-4 give the total timings for both algorithms for the four example sets. The experiments suggest that for low degree extensions *ICF* is faster than *BMD*, but in all experiments as the degree of extension grows *BMD* becomes faster than *ICF*. Another fact worth noting is that *ICF* depends directly on the extension degree m, since it isolates roots of a polynomial of degree mn. On the other hand, the only part of *BMD* that depends directly on m is computing approximations of coefficients, which in practice seems to take a very small proportion of the running time. The main root isolation loop depends only on the geometry of roots, which depends on m only through the worst case lower bound on root separation. Indeed, in all examples the running time of *ICF* grows substantially with m, but the running time of *BMD* either grows at a much slower pace or, in case of generalized Wilkinson polynomials, it even decreases with m (because the smallest root α of $L_m(x)$, and hence the root separation of $W_{n,\alpha}(x)$, increase with m). The superiority of the direct approach was also observed in [18].

Table 1. Randomly generated polynomials

	Algorithm	$m=2$	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	ICF	0.002	0.004	0.010	0.077	0.588
	BMD	0.002	0.002	0.004	0.007	0.017
$n=20$	ICF	0.003	0.007	0.025	0.260	2.38
	BMD	0.006	0.006	0.008	0.014	0.034
$n=50$	ICF	0.012	0.029	0.115	1.52	15.9
	BMD	0.022	0.024	0.031	0.044	0.093
$n=100$	ICF	0.043	0.107	0.407	6.61	65.8
	BMD	0.092	0.110	0.081	0.141	0.241
$n=200$	ICF	0.132	0.394	1.81	31.7	621
	BMD	0.348	0.362	0.508	0.440	1.02

Table 2. Generalized Laguerre polynomials

	Algorithm	$m=2$	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	ICF	0.005	0.008	0.022	0.119	0.822
	BMD	0.007	0.006	0.010	0.008	0.010
$n=20$	ICF	0.012	0.034	0.098	0.792	10.7
	BMD	0.062	0.066	0.070	0.077	0.090
$n=50$	ICF	0.105	0.275	1.41	16.6	312
	BMD	1.37	1.43	1.54	1.62	1.61
$n=100$	ICF	0.762	2.16	14.0	233	10644
	BMD	43.5	48.3	46.5	43.7	69.9
$n=200$	ICF	6.99	22.7	177	3425	> 36000
	BMD	1757	1753	1685	1678	1678

Table 3. Generalized Wilkinson polynomials

	Algorithm	$m=2$	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	ICF	0.005	0.006	0.022	0.123	0.966
	BMD	0.010	0.009	0.009	0.007	0.007
$n=20$	ICF	0.009	0.023	0.093	0.949	12.7
	BMD	0.028	0.028	0.026	0.026	0.020
$n=50$	ICF	0.085	0.265	1.35	21.4	320
	BMD	1.02	0.890	0.757	0.402	0.483
$n=100$	ICF	0.590	2.35	14.7	278	6070
	BMD	22.4	18.8	15.0	10.1	4.44
$n=200$	ICF	5.86	32.0	198	6685	> 36000
	BMD	1189	815	606	407	257

Table 4. Mignotte polynomials

	Algorithm	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	ICF	0.002	0.003	0.024	0.338
	BMD	0.005	0.007	0.013	0.028
$n=20$	ICF	0.004	0.018	0.195	5.73
	BMD	0.011	0.016	0.040	0.103
$n=50$	ICF	0.038	0.412	9.65	689
	BMD	0.083	0.126	0.246	0.812
$n=100$	ICF	0.871	9.81	550	28007
	BMD	0.485	0.621	1.55	4.67
$n=200$	ICF	13.0	364	21907	>36000
	BMD	1.17	3.72	10.9	31.0

A more detailed timing profile for the *ICF* algorithm is given in Tables 5-8. Rows marked *Res*, *Isol*, and *Ident*, give the timings of the three main steps of the *ICF* algorithm, namely the resultant computation, the real root isolation of the resultant, and the identification of the intervals isolating roots of the original polynomial. For random polynomials within the (m, n) range used in the experiments the resultant computation dominates the timing. However, one can observe that with a fixed m and increasing n, the proportion of time used by the real root isolation increases. For the other three sets of polynomials the computation time is dominated by the real root isolation step.

Table 5. *ICF*: Randomly generated polynomials

	Step	$m=2$	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	Res	0.001	0.002	0.007	0.070	0.562
	$Isol$	0	0	0.001	0.003	0.015
	$Ident$	0.001	0.001	0.002	0.004	0.010
$n=20$	Res	0.001	0.003	0.018	0.239	2.27
	$Isol$	0.001	0.002	0.003	0.012	0.099
	$Ident$	0.002	0.002	0.004	0.008	0.019
$n=50$	Res	0.001	0.009	0.083	1.40	15.1
	$Isol$	0.006	0.014	0.021	0.099	0.850
	$Ident$	0.005	0.006	0.010	0.021	0.048
$n=100$	Res	0.001	0.026	0.302	5.81	61.1
	$Isol$	0.030	0.066	0.085	0.760	4.66
	$Ident$	0.012	0.015	0.020	0.046	0.098
$n=200$	Res	0.002	0.090	1.16	26.4	467
	$Isol$	0.109	0.275	0.602	5.20	154
	$Ident$	0.021	0.028	0.042	0.098	0.439

Table 6. *ICF*: Generalized Laguerre polynomials

	Step	$m=2$	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	Res	0.001	0.001	0.007	0.053	0.302
	$Isol$	0.002	0.002	0.008	0.056	0.502
	$Ident$	0.002	0.005	0.007	0.010	0.018
$n=20$	Res	0.001	0.003	0.023	0.291	5.03
	$Isol$	0.005	0.020	0.059	0.470	5.58
	$Ident$	0.006	0.010	0.016	0.031	0.072
$n=50$	Res	0.001	0.014	0.176	5.17	126
	$Isol$	0.069	0.201	1.13	11.1	186
	$Ident$	0.035	0.059	0.102	0.297	0.501
$n=100$	Res	0.004	0.076	1.42	63.8	2329
	$Isol$	0.613	1.84	12.2	168	8312
	$Ident$	0.145	0.245	0.412	0.931	2.72
$n=200$	Res	0.021	0.597	15.2	810	?
	$Isol$	6.35	20.2	159	2607	?
	$Ident$	0.620	1.83	3.36	7.57	?

Table 7. *ICF*: Generalized Wilkinson polynomials

	Step	$m=2$	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	Res	0.001	0.001	0.009	0.051	0.376
	$Isol$	0.002	0.002	0.007	0.064	0.575
	$Ident$	0.002	0.003	0.006	0.008	0.015
$n=20$	Res	0	0.002	0.020	0.341	7.21
	$Isol$	0.004	0.012	0.059	0.580	5.44
	$Ident$	0.005	0.009	0.014	0.028	0.052
$n=50$	Res	0.001	0.015	0.181	7.52	133
	$Isol$	0.053	0.201	1.08	13.7	187
	$Ident$	0.031	0.049	0.084	0.188	0.758
$n=100$	Res	0.003	0.075	1.72	92.0	1557
	$Isol$	0.460	1.86	12.2	184	4508
	$Ident$	0.127	0.414	0.722	2.35	4.71
$n=200$	Res	0.013	0.971	20.4	1320	?
	$Isol$	4.65	28.0	172	5351	?
	$Ident$	1.19	3.02	4.86	13.5	?

Table 8. *ICF* : Mignotte polynomials

	Step	$m=3$	$m=5$	$m=10$	$m=20$
$n=10$	Res	0.001	0.001	0.007	0.088
	$Isol$	0	0.002	0.016	0.249
	$Ident$	0.001	0	0.001	0.001
$n=20$	Res	0	0.001	0.008	0.155
	$Isol$	0.003	0.017	0.187	5.58
	$Ident$	0.001	0	0	0
$n=50$	Res	0	0	0.008	0.184
	$Isol$	0.038	0.411	9.64	689
	$Ident$	0	0.001	0.001	0.003
$n=100$	Res	0	0	0.013	0.263
	$Isol$	0.870	9.81	550	28007
	$Ident$	0.001	0	0.002	0.012
$n=200$	Res	0	0	0.007	?
	$Isol$	13.0	364	21907	?
	$Ident$	0.001	0.002	0.029	?

Tables 9-12 provide more detailed experimental results for the *BMD* algorithm. Rows marked *Total time*, *No. of attempts*, *Isol. time*, and *Isol. accuracy* give, respectively, the total run time of the *BMD* algorithm, the number of times the DCM^L algorithm was run, the run time of the final DCM^L algorithm call which successfully isolated the roots, and the accuracy used in the final DCM^L algorithm call. One can observe that the approximation accuracy required for root isolation depends only on n and on the geometry of roots. For a fixed n the geometry of roots of random polynomials and Laguerre polynomials does not depend on m, and indeed the required approximation accuracy does not change with m. For Wilkinson polynomials the root separation increases with m, and hence the required approximation accuracy decreases with m. For Mignotte polynomials the root separation decreases with m, and hence the required approximation accuracy increases with m.

Table 9. *BMD*: Randomly generated polynomials

	Step	$m = 2$	$m = 3$	$m = 5$	$m = 10$	$m = 20$
$n = 10$	*Total time*	0.002	0.002	0.004	0.007	0.017
	No. of attempts	1	1	1	1	1
	Isol. time	0.002	0.002	0.004	0.007	0.017
	Isol. accuracy	64	64	64	64	64
$n = 20$	*Total time*	0.006	0.006	0.008	0.014	0.034
	No. of attempts	1	1	1	1	1
	Isol. time	0.006	0.006	0.008	0.014	0.034
	Isol. accuracy	128	128	128	128	128
$n = 50$	*Total time*	0.022	0.024	0.031	0.044	0.093
	No. of attempts	1	1	1	1	1
	Isol. time	0.022	0.024	0.031	0.044	0.093
	Isol. accuracy	256	256	256	256	256
$n = 100$	*Total time*	0.092	0.110	0.081	0.141	0.241
	No. of attempts	1	1	1	1	1
	Isol. time	0.092	0.110	0.081	0.141	0.241
	Isol. accuracy	512	512	512	512	512
$n = 200$	*Total time*	0.348	0.362	0.508	0.440	1.02
	No. of attempts	1	1	1	1	1
	Isol. time	0.348	0.362	0.508	0.440	1.02
	Isol. accuracy	1024	1024	1024	1024	1024

Table 10. *BMD*: Generalized Laguerre polynomials

	Step	$m = 2$	$m = 3$	$m = 5$	$m = 10$	$m = 20$
$n = 10$	*Total time*	0.007	0.006	0.010	0.008	0.010
	No. of attempts	1	1	1	1	1
	Isol. time	0.007	0.006	0.010	0.008	0.010
	Isol. accuracy	64	64	64	64	64
$n = 20$	*Total time*	0.062	0.066	0.070	0.077	0.090
	No. of attempts	2	2	2	2	2
	Isol. time	0.036	0.036	0.043	0.040	0.042
	Isol. accuracy	256	256	256	256	256
$n = 50$	*Total time*	1.37	1.43	1.54	1.62	1.61
	No. of attempts	2	2	2	2	2
	Isol. time	0.832	0.885	0.993	0.877	1.01
	Isol. accuracy	512	512	512	512	512
$n = 100$	*Total time*	43.5	48.3	46.5	43.7	69.9
	No. of attempts	3	3	3	3	3
	Isol. time	22.3	24.8	21.8	21.8	41.5
	Isol. accuracy	2048	2048	2048	2048	2048
$n = 200$	*Total time*	1757	1753	1685	1678	1678
	No. of attempts	3	3	3	3	3
	Isol. time	876	845	833	838	831
	Isol. accuracy	4096	4096	4096	4096	4096

Table 11. *BMD*: Generalized Wilkinson polynomials

	Step	$m = 2$	$m = 3$	$m = 5$	$m = 10$	$m = 20$
$n = 10$	*Total time*	0.010	0.009	0.009	0.007	0.007
	No. of attempts	2	2	2	2	2
	Isol. time	0.006	0.005	0.009	0.005	0.004
	Isol. accuracy	128	128	128	128	128
$n = 20$	*Total time*	0.028	0.028	0.026	0.026	0.020
	No. of attempts	1	1	1	1	1
	Isol. time	0.028	0.028	0.026	0.026	0.020
	Isol. accuracy	128	128	128	128	128
$n = 50$	*Total time*	1.02	0.890	0.757	0.402	0.483
	No. of attempts	2	2	2	1	1
	Isol. time	0.628	0.543	0.475	0.402	0.483
	Isol. accuracy	512	512	512	256	256
$n = 100$	*Total time*	22.4	18.8	15.0	10.1	4.44
	No. of attempts	2	2	2	2	1
	Isol. time	14.2	11.7	9.33	6.45	4.44
	Isol. accuracy	1024	1024	1024	1024	512
$n = 200$	*Total time*	1189	815	606	407	257
	No. of attempts	3	2	2	2	2
	Isol. time	587	537	381	254	163
	Isol. accuracy	4096	2048	2048	2048	2048

Table 12. *BMD*: Mignotte polynomials

	Step	$m = 3$	$m = 5$	$m = 10$	$m = 20$
$n = 10$	*Total time*	0.005	0.007	0.013	0.028
	No. of attempts	2	2	3	4
	Isol. time	0.003	0.005	0.006	0.012
	Isol. accuracy	128	128	256	512
$n = 20$	*Total time*	0.011	0.016	0.040	0.103
	No. of attempts	2	2	3	4
	Isol. time	0.007	0.009	0.017	0.037
	Isol. accuracy	256	256	512	1024
$n = 50$	*Total time*	0.083	0.126	0.246	0.812
	No. of attempts	2	2	3	4
	Isol. time	0.063	0.088	0.121	0.303
	Isol. accuracy	512	512	1024	2048
$n = 100$	*Total time*	0.485	0.621	1.55	4.67
	No. of attempts	2	2	3	4
	Isol. time	0.227	0.372	0.779	2.44
	Isol. accuracy	1024	1024	2048	4096
$n = 200$	*Total time*	1.17	3.72	10.9	31.0
	No. of attempts	1	2	3	4
	Isol. time	1.17	2.34	5.39	16.0
	Isol. accuracy	1024	2048	4096	8192

Acknowledgments

The authors thank the anonymous reviewers for their detailed comments that improved the presentation and the quality of the results.

ET is partially supported by GeoLMI (ANR 2011 BS03 011 06), HPAC (ANR ANR-11-BS02-013), and an FP7 Marie Curie Career Integration Grant.

References

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

[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. 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.

[4] F. Boulier, C. Chen, F. Lemaire, and M. Moreno Maza. Real root isolation of regular chains. In *Proc. Asian Symposium on Computer Mathematics (ASCM)*, pages 1–15, 2009.

[5] J.-S. Cheng, X.-S. Gao, and C.-K. Yap. Complete numerical isolation of real roots in zero-dimensional triangular systems. *J. Symbolic Computation*, 44:768–785, July 2009.

[6] G. Collins and A. Akritas. Polynomial real root isolation using Descartes' rule of signs. In *SYMSAC '76*, pages 272–275, New York, USA, 1976. ACM Press.

[7] G. E. Collins and R. Loos. Polynomial real root isolation by differentiation. In *Proc. of the 3rd Int'l Symp. on Symbolic and Algebraic Computation*, SYMSAC '76, pages 15–25, New York, NY, USA, 1976. ACM.

[8] 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.

[9] 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).

[10] 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.

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

[12] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. A Descartes Algorithm for Polynomials with Bit-Stream Coefficients. In V. Ganzha, E. Mayr, and E. Vorozhtsov, editors, *CASC*, volume 3718 of *LNCS*, pages 138–149. Springer, 2005.

[13] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the Descartes method. In *Proc. Annual ACM ISSAC*, pages 71–78, New York, USA, 2006.

[14] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real Algebraic Numbers: Complexity Analysis and Experimentation. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, editors, *Reliable Implementations of Real Number Algorithms: Theory and Practice*, volume 5045 of *LNCS*, pages 57–82. Springer Verlag, 2008. (also available in www.inria.fr/rrrt/rr-5897.html).

[15] I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. The DMM bound: Multivariate (aggregate) separation bounds. In S. Watt, editor, *Proc. 35th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 243–250, Munich, Germany, July 2010. ACM.

[16] 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.

[17] J. Johnson and W. Krandick. Polynomial real root isolation using approximate arithmetic. In *Proc. Int'l Symp. on Symbolic and Algebraic Comp. (ISSAC)*, pages 225–232. ACM, 1997.

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

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

[20] T. Lickteig and M.-F. Roy. Sylvester-Habicht Sequences and Fast Cauchy Index Computation. *J. Symb. Comput.*, 31(3):315–341, 2001.

[21] R. Loos. Computing in algebraic extensions. In B. Buchberger, G. E. Collins, R. Loos, and R. Albrecht, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 173–187. Springer-Verlag, 1983.

[22] Z. Lu, B. He, Y. Luo, and L. Pan. An algorithm of real root isolation for polynomial system. In D. Wang and L. Zhi, editors, *Proc. 1st ACM Int'l Work. Symbolic Numeric Computation (SNC)*, pages 94–107, 2005.

[23] K. Mehlhorn and M. Sagraloff. A deterministic algorithm for isolating real roots of a real polynomial. *J. Symbolic Computation*, 46(1):70–90, 2011.

[24] K. Mehlhorn, M. Sagraloff, and P. Wang. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 66(0):34 – 69, 2015.

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

[26] B. Mourrain, M. Vrahatis, and J. Yakoubsohn. On the complexity of isolating real roots and computing with certainty the topological degree. *J. Complexity*, 18(2), 2002.

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

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

[29] V. Pan and E. Tsigaridas. Nearly Optimal Computations with Structured Matrices. In *Proceedings of the 2014 Symposium on Symbolic-Numeric Computation* , pages 21–30, Shanghai, China, July 2014.

[30] V. Pan and E. P. Tsigaridas. Nearly optimal refinement of real roots of a univariate polynomial. *Arxiv*, Feb 2014.

[31] V. Y. Pan and E. Tsigaridas. Nearly optimal computations with structured matrices (journal version). Jan. 2015.

[32] V. Y. Pan and E. P. Tsigaridas. On the boolean complexity of real root refinement. In *Proc. 38th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 299–306, Boston, USA, Jun 2013. ACM.

[33] C. Pascal and E. Schost. Change of order for bivariate triangular sets. In *Proc. 31th ACM Int'l Symp. on Symbolic & Algebraic Comp. (ISSAC)*, pages 277–284, New York, NY, USA, 2006.

[34] R. Rioboo. Towards faster real algebraic numbers. *J. Symb. Comput.*, 36(3-4):513–533, 2003.

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

[36] S. Rump. On the sign of a real algebraic number. In *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 238–241, New York, NY, USA, 1976. ACM Press.

[37] S. M. Rump. Real root isolation for algebraic polynomials. *ACM SIGSAM Bulletin*, 11(2):327–336, 1977.

[38] M. Sagraloff. On the complexity of real root isolation. *CoRR*, abs/1011.0344v1, 2010.

[39] 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.

[40] 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.

[41] 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.

[42] E. P. Tsigaridas and I. Z. Emiris. On the complexity of real root isolation using Continued Fractions. *Theor. Comput. Sci.*, 392:158–173, 2008.

[43] M. van Hoeij and M. Monagan. A modular GCD algorithm over number fields presented with multiple extensions. In *Proc. Annual ACM ISSAC*, pages 109–116, July 2002.

[44] B. Xia and L. Yang. An algorithm for isolating the real solutions of semi-algebraic systems. *J. Symbolic Computation*, 34:461–477, November 2002.

[45] B. Xia and T. Zhang. Real solution isolation using interval arithmetic. *Comput. Math. Appl.*, 52:853–860, September 2006.

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

A A bound for the resultant

Proposition 16. *Let $B = \sum_{i,j} c_{i,j} x^i y^j \in \mathbb{Z}[x,y]$ of degree n with respect to y and of degree η with respect to x, and of bitsize σ. Let $A = \sum_{i=0}^{m} a_i x^i \in \mathbb{Z}[x]$ of degree m and bitsize τ. The resultant of B and A with respect to x is univariate polynomial in y of degree at most mn and bitsize at most $m\sigma + \eta\tau + m\lg(n+1) + (m+\eta)\lg(m+\eta)$ or $\widetilde{\mathcal{O}}(m\sigma + \eta\tau)$.*

Proof: The proof follows closely the proof in [2, Prop. 8.15] that provides a bound for general multivariate polynomials. We can compute the resultant of $B(x,y)$ and $A(x)$ with respect to x from the determinant of the Sylvester matrix, by considering them as univariate polynomial in x, with coefficients that are polynomial in y, which is

$$\begin{pmatrix}
b_\eta & b_{\eta-1} & \dots & b_0 & & & & \\
 & b_\eta & b_{\eta-1} & \dots & b_0 & & & \\
 & & \ddots & \ddots & & \ddots & & \\
 & & & b_\eta & b_{\eta-1} & \dots & b_0 & \\
a_m & a_{m-1} & \dots & a_0 & & & & \\
 & a_m & a_{m-1} & \dots & a_0 & & & \\
 & & \ddots & \ddots & & \ddots & & \\
 & & & a_m & a_{m-1} & \dots & a_0 &
\end{pmatrix}
\begin{matrix}
x^{m-1}B \\ x^{m-2}B \\ \vdots \\ x^0 B \\ x^{\eta-1}A \\ x^{\eta-2}A \\ \vdots \\ x^0 A
\end{matrix}$$

where $b_k = \sum_{i=0}^{n} c_{i,k} y^i$.

The resultant is a factor of the determinant of the Sylvester matrix. The matrix is of size $(\eta+m)\times(\eta+m)$, hence the determinant consists of $(\eta+m)!$ terms. Each term is a product of m univariate polynomials in y, of degree η and bitsize σ, times the product of n numbers, of bitsize τ. The first product results in polynomials in y of degree at most mn and bitsize at most $m\sigma + m\lg(n+1)$; since there are at most $(n+1)^m$ terms with bitsize at most $m\sigma$ each. The second product results in numbers of bitsize at most $\eta\tau$. Hence each term of the determinant is, in the worst case a univariate polynomial in y of degree m and bitsize $m\sigma + \eta\tau + m\lg(n+1)$. We conclude that the resultant is of degree at most mn in y and of bitsize $m\sigma + \eta\tau + m\lg(n+1) + (m+\eta)\lg(m+\eta)$ or $\widetilde{\mathcal{O}}(m\sigma + \eta\tau)$. □