Real Algebraic Numbers: Complexity Analysis and Experimentation

Ioannis Z. Emiris[1], Bernard Mourrain[2], and Elias P. Tsigaridas[1]

[1] Department of Informatics and Telecommunications
National Kapodistrian University of Athens, HELLAS
{emiris,et}@di.uoa.gr
[2] GALAAD, INRIA
Sophia-Antipolis, FRANCE
mourrain@sophia.inria.fr

Abstract. We present algorithmic, complexity and implementation results concerning real root isolation of a polynomial of degree d, with integer coefficients of bit size $\leq \tau$, using Sturm (-Habicht) sequences and the Bernstein subdivision solver. In particular, we unify and simplify the analysis of both methods and we give an asymptotic complexity bound of $\widetilde{\mathcal{O}}_B(d^4\tau^2)$. This matches the best known bounds for binary subdivision solvers. Moreover, we generalize this to cover the non square-free polynomials and show that within the same complexity we can also compute the multiplicities of the roots. We also consider algorithms for sign evaluation, comparison of real algebraic numbers and simultaneous inequalities, and we improve the known bounds at least by a factor of d.
Finally, we present our `C++` implementation in SYNAPS and some preliminary experiments on various data sets.

1 Introduction

The representation and manipulation of shapes is important in many applications, e.g. CAGD, non linear computational geometry, robotics and molecular biology. The usual underlying models for these shapes are parametrized patches of rational surfaces, BSplines, natural quadrics and implicit algebraic curves or surfaces. Geometric processing on these objects, e.g. computing boundary representations [8], arrangements [18,38], Voronoi diagram of curved objects [21], requires the intensive use of polynomial solvers and computations with algebraic numbers. In such applications, a geometric model may involve several thousands of algebraic primitives. Their manipulations involve the computation of intersection points, of predicates on these intersection points (such as the comparison of coordinates), of the sign of polynomial expressions at these points (such as the sign of a polynomial which defines the boundary of an object). The coordinates of these intersection points, which are the solutions of polynomial equations, are real algebraic numbers that we need to manipulate efficiently.

The objective of this paper is to give an overview of effective computations with real algebraic numbers, which unify, simplify and improve previous approaches. We will tackle both complexity analysis and practical issues. We consider two algorithms for real root isolation of univariate integer polynomials, one based on Sturm sequences and one based on Descartes' rule of sign and we will put both under the general concept of a subdivision solver. We will also analyse algorithms for sign evaluation, comparison of real algebraic numbers and the problem of simultaneous inequalities.

Our aim is to provide better insights on these algorithms and better bounds on their complexity. For the analysis we consider the bit complexity model which is more realistic than the arithmetic one in the problems we are interested in. Our algorithms are essentially output sensitive, since they depend not only on the input bit size, but also on the actual separation bound, as we will see.

Notation. In what follows $\mathcal{O}_B$ means bit complexity and the $\widetilde{\mathcal{O}}_B$-notation means that we are ignoring logarithmic factors. For a polynomial $f \in \mathbb{Z}[X]$, $\mathsf{deg}(f)$ denotes its degree. By $\mathcal{L}(f)$ we denote an upper bound on the bit size of the coefficients of f (including a bit for the sign). For $\mathsf{a} \in \mathbb{Q}$, $\mathcal{L}(\mathsf{a})$ is the maximum bit size of the numerator and the denominator. Let $\mathsf{M}(\tau)$ denote the bit complexity of multiplying two integers of bit size at most τ and $\mathsf{M}(d, \tau)$ denote the bit complexity of multiplying two univariate polynomials of degrees bounded by d and coefficient bit size at most τ. Using FFT, $\mathsf{M}(\tau) = \mathcal{O}_B(\tau \log^{c_1} \tau)$ and $\mathsf{M}(d, \tau) = \mathcal{O}_B(d\tau \log^{c_2}(d\tau))$ for suitable constants c_1, c_2.

Prior works. Various algorithms exist for polynomial real root isolation, but most of them focus on square-free polynomials. There is a huge bibliography on the problem and the references cited in this paper are only the tip of the iceberg of the existing bibliography.

Collins and Akritas [10] introduced a subdivision-based real root isolation algorithm that relies on Descartes' rule of sign (we call it *Descartes solver* from now on) and derived a bound of $\widetilde{\mathcal{O}}_B(d^6\tau^2)$. Johnson [27] improved the bound to $\widetilde{\mathcal{O}}_B(d^5\tau^2)$ without using fast Taylor shifts [48], whereas a gap in his proof was corrected by Krandick [29]. The latter also presented a different way of traversing the subdivision tree. Rouillier and Zimmermann (c.f. [43] and references therein) presented a unified approach with optimal memory management for various variants of the Descartes solver.

An algorithm (we call it *Bernstein solver* from now on) that is based on a combination of Descartes' rule and on the properties of the Bernstein basis was first introduced by Lane and Riesenfeld [31] and a bound on its complexity was first obtained by Mourrain et al [39]. The interested reader may also refer to Mourrain et al [37] for a variant with optimal memory management and the connection to Descartes solver. In the same context, Eigenwillig et al [15] proposed a randomized algorithm for square-free polynomials with bit stream coefficients. The complexity of all these algorithms was bounded by $\widetilde{\mathcal{O}}_B(d^6\tau^2)$. Recently, it was proven that the number of steps of both Descartes and Bernstein solver is $\widetilde{\mathcal{O}}_B(d\tau)$ [16,46], which is the crucial step for obtaining a $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ bound for both solvers [16], provided the polynomials are square-free.

Another algorithm, also based on Descartes' rule of sign but which does not rely on subdivision schemes, is the CF algorithm [1,47]. It exploits the continued fraction expansion of the real roots of a polynomial in order to isolate them. The expected bit complexity of this algorithm is $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ [47].

If we restrict ourselves to real root isolation using Sturm (or Sturm-Habicht) sequences (we call it *Sturm solver* from now on) the first complete complexity analysis is probably due to Heindel [26]; see also [11], who obtained a complexity of $\widetilde{\mathcal{O}}_B(d^7\tau^3)$. Du et al [14], giving an amortized-like argument for the number of subdivisions, obtained a complexity of $\widetilde{\mathcal{O}}_B(d^4\tau^2)$, for square-free polynomials.

Another family of solvers (that we call numerical), compute an approximation of all the roots (real and complex) of a polynomial up to a desired accuracy (see e.g. [45,40]). They are based on the construction of balanced splitting circles in the complex plane and achieve the quasi-optimal complexity bound $\widetilde{\mathcal{O}}_B(d^3\tau)$, if we want to isolate the roots. However, performance in practice does not always agree with that predicted by asymptotic analysis. Let us also mention the Aberth solver [5,6], which has unknown (bit) complexity, but is very efficient in practice.

For sign evaluation and comparison as well as computations with real algebraic numbers the reader may refer to [42]. In [19] for degree ≤ 4, it is proved that these operations can be performed in $\mathcal{O}(1)$, or $\widetilde{\mathcal{O}}_B(\tau)$. For the problem of simultaneous inequalities (SI), we are interested in computing the (number of) real roots of a polynomial f, such that n other polynomials achieve specific sign conditions, where the degree of all the polynomials is bounded by d and their bit size by τ. Ben-Or, Kozen and Reif [3] presented the BKR algorithm for SI and Canny [7] improved it in the univariate case (by a factor) achieving $\mathcal{O}(n(m\,d\log(m)\log^2(d)+m^{2.376}))$ arithmetic complexity, where m is the number of real roots of f. Coste and Roy [12] introduced Thom's encoding for the real roots of a polynomial and SI in this encoding (see also [44]). Their approach is purely symbolic and works over arbitrary real closed fields. They state a complexity of $\widetilde{\mathcal{O}}_B(N^8)$, using fast multiplication algorithms but not fast computations and evaluation of polynomial sequences, where $N \geq n, d, \tau$. Basu, Pollack and Roy [2] presented an algorithm for SI where the real algebraic numbers are in isolating interval representation, with complexity $\widetilde{\mathcal{O}}_B(nd^6\tau^2)$ or $\widetilde{\mathcal{O}}_B(N^9)$, that uses repeated refinements of the isolating intervals and does not assume fast multiplication algorithms.

Results. For the problem of real root isolation of a univariate polynomial, we consider the general concept of a subdivision solver, Fig. 1, that mimics the binary search algorithm. The Sturm and Bernstein solvers count differently the number of real roots of a polynomial in an interval; see Cor. 2.1 and Prop. 3.1, respectively. Moreover, the Sturm solver counts exactly the number of real roots while the Bernstein solver provides an overestimation. However, exploiting the fact that Descartes' rule of sign (Prop. 3.1) can not overestimate the number of real roots by more than the degree of the polynomial, both solvers can be put under the general concept of the subdivision solver of Fig. 1. With exactly the same arguments we can prove that they perform the same number of steps.

The analysis that we present, Prop. 5.2, unifies and simplifies significantly the previous approaches and applies as is to the Descartes solver, as well.

For the Sturm solver we present an algorithm with complexity $\widetilde{\mathcal{O}}_B(d^4\tau^2)$, that improves the result of Du et al [14], see also [13], by extending it to non square-free polynomials. We simplify significantly the proof (Th. 5.1) and unify it with the Bernstein approach. We also show that computing the multiplicities of the roots can be achieved within the same complexity bound.

For the Bernstein solver, we simplify the proof of Eigenwillig et al [16,46] for the number of subdivisions by considering the subdivision tree at an earlier level and by using Th. 4.1 exactly as stated in [27,30]. Thus, we arrive at the same bound for the Bernstein subdivision method (Th. 5.1) as in [16], but for polynomials which are not necessarily square-free.

Real root isolation is an important ingredient for the construction of algebraic numbers represented in isolating interval representation, see Def. 6.1. We analyze the complexity of comparison, sign evaluation and simultaneous inequalities (Sec. 6). Even though the algorithms for these operations are not new [2,19,42,51], the results from real solving and optimal algorithms for polynomial remainder sequences, allow us to improve the complexity of all the algorithms, at least by a factor of d (Cor. 6.1, 6.2). For SI we prove a bound (Cor. 6.3) of $\widetilde{\mathcal{O}}_B(d^4\tau \max\{n,\tau\})$.

These algebraic operations ought to have efficient and generic implementations so that they can be used by other scientific communities. We present a package of SYNAPS [36] that provides these functionalities on real algebraic numbers and exploits various algorithmic and implementation techniques. Experimental results (Sec. 7) illustrate the advantages of the software and our implementation of various algorithms for real root isolation.

Our results extend directly to the bivariate case, i.e. real solving a polynomial system, sign evaluation of a bivariate polynomial evaluated over two algebraic numbers and SI in two variables. However, due to reasons of space, we cannot present these results here. The reader may refer to [22,20].

Outline. In Sec. 2, we recall the main ingredients of the Sturm solver and analyse them in detail. Sec. 3 presents the ingredients of the Bernstein solver and their complexity. In Sec. 4, we present the general scheme for two subdivision algorithms based on Sturm-Habicht sequences and on the Bernstein basis representation, for real root isolation and computation of the multiplicities. The following section is devoted to the complexity analysis of both methods. Sec. 6 is devoted to operations with real algebraic numbers, i.e. comparison, sign evaluation and SI. Sec. 7 illustrates our implementation in SYNAPS and experiments concerning real root isolation on various data sets. Finally, we sketch our current and future work in Sec. 8.

2 Preliminaries for Sturm–Habicht Sequences

We recall here the main ingredients related to Sturm(-Habicht) sequences computations and their bit complexity.

Let $f = \sum_{k=0}^{p} f_k x^k, g = \sum_{k=0}^{q} g_k x^k \in \mathbb{Z}[x]$ where $\texttt{deg}(f) = p \geq q = \texttt{deg}(g)$ and $\mathcal{L}(f) = \mathcal{L}(g) = \tau$. We denote by $\texttt{rem}\,(f,g)$ and $\texttt{quo}\,(f,g)$ the remainder and the quotient, respectively, of the Euclidean division of f by g, in $\mathbb{Q}[x]$.

Definition 2.1. [32] *The* signed polynomial remainder sequence *of f and g, denoted by* $\mathbf{SPRS}\,(f,g)$*, is the polynomial sequence*

$$R_0 = f, R_1 = g, R_2 = -\,\texttt{rem}\,(f,g)\,, \ldots, R_k = -\,\texttt{rem}\,(R_{k-2}, R_{k-1})\,,$$

with k the minimum index such that $\texttt{rem}\,(R_{k-1}, R_k) = 0$. The quotient sequence *of f and g is the polynomial sequence $\{Q_i\}_{0 \leq i \leq k-1}$, where $Q_i = \texttt{quo}\,(R_i, R_{i+1})$ and the* quotient boot *is $(Q_0, Q_1, \ldots, Q_{k-1}, R_k)$.*

There is a huge bibliography on signed polynomial remainder sequences (c.f. [2,49,51] and references therein). Gathen and Lucking [50] presented a unified approach to subresultants, while El Kahoui [17] studied the subresultants in arbitrary commutative rings. For the Sturm-Habicht (or Sylvester-Habicht) sequences the reader may refer to González-Vega et al [24], see also [2,32,33].

In this paper we consider the Sturm-Habicht sequence of f and g, i.e. $\mathbf{StHa}(f,g)$, which contains polynomials proportional to the polynomials in $\mathbf{SPRS}\,(f,g)$. Sturm-Habicht sequences achieve better bounds on the bit size of the coefficients.

Let M_j be the matrix which has as rows the coefficient vectors of the polynomials fx^{q-1-j}, fx^{q-2-j}, $\ldots, fx, f,\ g, gx, \ldots, gx^{p-2-j}, gx^{p-1-j}$ with respect to the monomial basis $x^{p+q-1-j}$, $x^{p+q-2-j}, \ldots, x, 1$. The dimension of M_j is $(p+q-2j) \times (p+q-j)$. For $l = 0, \ldots, p+q-1-j$ let M_j^l be the square matrix of dimension $(p+q-2j) \times (p+q-2j)$ obtained by taking the first $p+q-1-2j$ columns and the $l + (p+q-2j)$ column of M_j.

Definition 2.2. *The Sturm-Habicht sequence of f and g, is the sequence*

$$\mathbf{StHa}(f,g) = (H_p = H_p(f,g), \ldots, H_0 = H_0(f,g)),$$

where $H_p = f$, $H_{p-1} = g$, $H_j = \delta_j \sum_{l=0}^{j} \det\,(M_j^l) x^l$ and $\delta_j = (-1)^{(p+q-j)(p+q-j-1)/2}$. The sequence of principal Sturm-Habicht coefficients $(h_p = h_p(f,g), \ldots, h_0(f,g))$ is defined as $h_p = 1$ and h_j is the coefficient of x^j in the polynomial H_j, for $0 \leq j \leq p$. When $h_j = 0$ for some j then the sequence is called defective, otherwise non-defective.

If $\mathbf{StHa}(f,g)$ is non-defective then it coincides up to sign with the classical subresultant sequence introduced by Collins [9], see also [51]. However, in the defective case, one can have better control on the bit size of the coefficients in the sequence, see e.g. [32,33].

Theorem 2.1. [2,41,33,49] *There is an algorithm that computes* $\mathbf{StHa}(f, g)$ *in* $\mathcal{O}_B(pq\,\mathsf{M}\,(p\tau))$*, or* $\widetilde{\mathcal{O}}_B(p^2 q\tau)$*. Moreover,* $\mathcal{L}\,(H_j(f, g)) = \mathcal{O}(p\tau)$*.*

Let the quotient boot that corresponds to $\mathbf{StHa}(f, g)$, be $\mathbf{StHaQ}(f, g) = (Q_0, Q_1, \ldots, Q_{k-1}, H_k)$. The number of coefficients in $\mathbf{StHaQ}(f, g)$ is $\mathcal{O}(q)$ and their bit size is $\mathcal{O}(p\tau)$ [2,41].

Theorem 2.2. [2,32,41,49] *The quotient boot, the resultant and the* gcd *of* f *and* g*, can be computed in* $\mathcal{O}_B(q \log(q)\,\mathsf{M}\,(p\tau))$ *or* $\widetilde{\mathcal{O}}_B(p\,q\,\tau)$*.*

Theorem 2.3. [32,41] *There is an algorithm that computes the evaluation of* $\mathbf{StHa}(f, g)$ *over a number* a*, where* $\mathsf{a} \in \mathbb{Q} \cup \{\pm\infty\}$ *and* $\mathcal{L}\,(\mathsf{a}) = \sigma$ *with complexity* $\mathcal{O}_B(q \log(q)\,\mathsf{M}\,(\max\{p\tau, q\sigma\}))$ *or* $\mathcal{O}_B(q\mathsf{M}\,(\max\,(p\tau, q\sigma)))$ *if* $\mathbf{StHaQ}(f, g)$ *is already computed. In both cases the complexity is* $\widetilde{\mathcal{O}}_B(q \max\{p\tau, q\sigma\})$*.*

In many cases, e.g. real root isolation, sign evaluation, comparison of real algebraic numbers, we need the evaluation of $\mathbf{StHa}(f, f^{'})$ over a rational number of bit size $\mathcal{O}(p\tau)$. If we perform the evaluation by Horner's rule then for every polynomial in sequence, we must perform $\Omega(p)$ multiplications between numbers of bit size $\mathcal{O}(p\tau)$ and $\mathcal{O}(p^2\tau)$, thus the overall complexity is $\mathcal{O}_B(p^3\mathsf{M}\,(p\tau))$.

However, when we compute the complete $\mathbf{StHa}(f, f^{'})$ in $\mathcal{O}_B(p^2\mathsf{M}\,(p\tau))$ (Th. 2.1), the quotient boot is computed implicitly [41,2]. Thus, we can use the quotient boot in order to perform the evaluation even if we have already computed all the polynomials in the sequence. Notice also that the computation should be started by the last element of the quotient boot so as to avoid the costly evaluation of the first two polynomials in the sequence using Horner's scheme.

Even though this approach is optimal, it involves big constants in its complexity, thus it is not efficient in practice when the length of the sequence is not sufficiently big or when the sequence is defective, see e.g. [13]. Moreover, special techniques should be used for its implementation to avoid costly operations with rational numbers. So, as it is always the case with optimal algebraic algorithms, the implementation is far from a trivial task.

Theorem 2.4. [2] *The square-free part of* f*, i.e.* f_{red}*, can be computed from* $\mathbf{StHa}(f, f^{'})$*, in* $\mathcal{O}_B(p \log(p)\,\mathsf{M}\,(p\tau))$ *or* $\widetilde{\mathcal{O}}_B(p^2\tau)$*, and* $\mathcal{L}\,(f_{red}) = \mathcal{O}(p + \tau)$*.*

Let $W_{(f,g)}(\mathsf{a})$ denote the number of modified sign changes of the evaluation of $\mathbf{StHa}(f, g)$ over a. Notice that $W_{(f,g)}(\mathsf{a})$ does not refer to the usual counting of sign variations, since special care should be taken for the presence of consecutive zeros [2,24,32].

Theorem 2.5. [2,51,42] *Let* $f, g \in \mathbb{Z}[x]$ *be relatively prime polynomials, where* f *is square-free and* $f^{'}$ *is the derivative of* f *and its leading coefficient* $f_d > 0$*. If* $\mathsf{a} < \mathsf{b}$ *are both non-roots of* f *and* γ *ranges over the roots of* f *in* (a, b)*, then* $W_{(f,g)}(\mathsf{a}) - W_{(f,g)}(\mathsf{b}) = \sum_{\gamma} \operatorname{sign}\,(f^{'}(\gamma) g(\gamma))$*.*

Corollary 2.1. *If* $g = f^{'}$ *then* $\mathbf{StHa}(f, f^{'})$ *is a Sturm sequence and Th. 2.5 counts the number of distinct real roots of* f *in* (a, b)*.*

For the Sturm solver $V(f, [\mathsf{a}, \mathsf{b}])$ will denote $V(f, [\mathsf{a}, \mathsf{b}]) = W_{(f,f')}(\mathsf{a}) - W_{(f,f')}(\mathsf{b})$.

3 Preliminaries for the Bernstein Basis Representation

In this section we present the main ingredients needed for the representation of polynomials in the Bernstein basis.

Let $\mathbb{R}[x]_d$ be the set of real polynomials of degree d. For $a < b \in \mathbb{R}$, we denote by $B_d^i(x; a, b) = \binom{d}{i} \frac{(x-a)^i (b-x)^{d-i}}{(b-a)^d}$ $(i = 0, \ldots, d)$ the Bernstein basis of $\mathbb{R}[x]_d$ on an interval $[a, b]$.

For any polynomial $f = \sum_{i=0}^{d} b_i B_d^i(x; a, b) \in \mathbb{R}[x]_d$ represented in the Bernstein basis, the coefficients $\mathbf{b} = (b_i)_{i=0,\ldots,d}$ are called *control coefficients* of f. We denote by $V(f, [a, b]) \equiv V(\mathbf{b})$, the number of sign changes in this sequence $\mathbf{b}$ (after removing zero coefficients).

The following theorem, which is a direct consequence of Descartes' rule, allows us to bound the number of real roots of f on the interval $[a, b]$

Proposition 3.1. [2] *The number N of real roots of f on (a, b) is bounded by $V(f, [a, b])$. Moreover $N \equiv V(f, [a, b]) \mod 2$.*

Given a polynomial f represented in the Bernstein basis on an interval $[a, b]$, de Casteljau's algorithm (see e.g. [2,37]), allows us to compute its representation in the Bernstein bases on the two sub-intervals, $I_L = [a, (1-t)a + tb]$ and $I_R = [(1-t)a + tb, b]$, where $0 \le t \le 1$. Namely, $\mathbf{b}_L = (b_0^i)_{i=0,\ldots,d}$ (resp. $\mathbf{b}_R = (b_i^{d-i})_{i=0,\ldots,d}$) are the control coefficients of f on I_L (resp. I_R), where $b_i^0 = b_i, i = 0, \ldots, d$, and

$$b_i^r = (1-t)\, b_i^{r-1} + t\, b_{i+1}^{r-1}(t), \quad 0 \le i \le d - r,\ 0 \le r \le d. \tag{1}$$

In order to analyse the complexity of the de Casteljau algorithm we recall some polynomial transformations related to the Bernstein representation, see [37] for more details. Let $\mathbb{R}[x, y]_{[d]}$ be the set of homogeneous polynomials of degree d in (x, y). For any $f \in \mathbb{R}[x]_d$, we denote by $\overline{f}$ the homogenisation of f in degree d. For $\lambda \neq 0, \mu \in \mathbb{R}$, consider the following maps, $\mathbb{R}^2 \to \mathbb{R}^2$:

- $\rho : (x, y) \mapsto (y, x)$,
- $H_\lambda : (x, y) \mapsto (\lambda x, y)$, $H'_\lambda : (x, y) \mapsto (x, \lambda y)$,
- $T_\mu : (x, y) \mapsto (x - \mu y, y)$, $T'_\mu : (x, y) \mapsto (x, y - \mu x)$.

The composition of the previous maps with $\overline{f}$ induces invertible transformations on the set of polynomials of degree d. The corresponding maps for non-homogeneous polynomials, which we denote using the same names, are: $\forall f \in \mathbb{R}[x]_d$, $\rho(f) = x^d f(1/x)$, $H_\lambda(f) = f(\lambda x)$, $H'_\lambda(f) = f(\lambda^{-1} x)$, $T_\mu(f) = f(x - \mu)$ and $T'_\mu(f) = (1 - \mu\, x)^d f(\frac{x}{1-\mu\, x})$.

For any polynomial, $f(x) = \sum_{i=0}^{d} b_i B_d^i(x; a, b)$, we have

$$\rho \circ T_1 \circ \rho \circ H_{b-a} \circ T_{-a}(f) = \sum_{i=0}^{d} \binom{d}{i} b_i x^i.$$

Now consider another interval $[c, e]$. The representation of f in the Bernstein basis on $[c, e]$ is $f(x) = \sum_{i=0}^{d} b'_i B_d^i(x; c, e)$. The map which transforms f from its Bernstein representation on $[a,b]$ to its Bernstein representation on $[c,e]$, i.e. from $\sum_{i=0}^{d}\binom{d}{i}b_i x^i$ to $\sum_{i=0}^{d}\binom{d}{i}b'_i x^i$ is

$$\rho\circ T_1\circ\rho\circ H_{e-c}\circ T_{-c}\circ T_a\circ H_{\frac{1}{b-a}}\circ\rho\circ T_{-1}\circ\rho = T'_1\circ H_{e-c}\circ T_{a-c}\circ H_{\frac{1}{b-a}}\circ T'_{-1}. \quad (2)$$

If we consider $[a,b]=[0,1]$ and $[c,e]=[0,\frac{1}{2}]$ then map (2) becomes: $\rho\circ T_{-1}\circ\rho\circ H_{\frac{1}{2}}\circ\rho\circ T_1\circ\rho$. And after simplifications, we obtain

$$\Delta_L : \overline{f}\mapsto\overline{f}(x+\frac{y}{2},\frac{y}{2}) = \overline{f}\circ T_{-1}\circ H'_{\frac{1}{2}}. \quad (3)$$

If we consider the symmetric case, i.e. $[a,b]=[0,1]$ and $[c,e]=[\frac{1}{2},1]$, then map (2) becomes: $\rho\circ T_{-1}\circ\rho\circ H_{\frac{1}{2}}\circ T_{-\frac{1}{2}}\circ\rho\circ T_1\circ\rho$. It corresponds to the following map on the homogeneous polynomials:

$$\Delta_R : \overline{f}\mapsto\overline{f}(\frac{x}{2},\frac{x}{2}+y) = \overline{f}\circ T'_{-1}\circ H_{\frac{1}{2}}.$$

In both cases, multiplication by 2^d yields the maps

$$\overline{\Delta}_R : \overline{f}\mapsto\overline{f}(x,x+2\,y) \quad \text{and} \quad \overline{\Delta}_L : \overline{f}\mapsto\overline{f}(2\,x+y,y), \quad (4)$$

that operate operate on polynomials with integer coefficients.

Proposition 3.2. *Let $(b_i)_{i=0,\dots,d}\in\mathbb{Z}^{d+1}$ be the coefficients of a polynomial f in the Bernstein basis on the interval $[a,b]$, and let ν be a bound on the bit size of coefficients. The complexity of computing the Bernstein coefficients of f on the two sub-intervals $[a,\frac{a+b}{2}]$ and $[\frac{a+b}{2},b]$ is bounded by $\widetilde{\mathcal{O}}_B(d(d+\nu))$ and their bit size is bounded by $d+\nu$.*

Proof. Using the de Casteljau scheme, Eq. (1) using $t=\frac{1}{2}$, we prove by induction that the coefficients $b_i^r = \frac{(b_i^{r-1}+b_{i+1}^{r-1})}{2}$ are of the form $\frac{\overline{b}_i^r}{2^i}$, where $\overline{b}_i^r\in\mathbb{Z}$ is of bit size $\le\nu+r$. Reducing to the same denominator 2^d, we obtain integer coefficients of bit size $\le\nu+d$.

We denote by ν' the bit size of the coefficients $(\binom{d}{i}b_i)_{i=0,\dots,d}$ where $(b_i)_{i=0,\dots,d}$ are the coefficients of f in the Bernstein basis $(B_d^i(x;a,b))_{i=0,\dots,d}$. Notice that $\nu'\le\nu+d$.

In order to compute the coefficients of f on $[a,\frac{a+b}{2}]$ and $[\frac{a+b}{2},b]$, we apply the same operations as when we compute the coefficients of a polynomial in the Bernstein basis on $[0,\frac{1}{2}]$ and $[\frac{1}{2},1]$, given its coefficients in the Bernstein basis on $[0,1]$.

According to (3), applying de Casteljau's algorithm corresponds first to multiply by the binomial coefficients, then to shift, $y\mapsto x+y$, then to scale one variable of the homogeneous polynomial $\overline{f}$ by $\frac{1}{2}$, and finally to divide by the binomial coefficients[3].

[3] There is no need for the division step if we have to apply repeatedly the shift operation.

Since the bit size of the binomial coefficients is bounded by d (their sum is 2^d), multiplying the b_i by them costs at most $\widetilde{\mathcal{O}}_B(d(\nu + d))$. Shifting by 1 a polynomial of degree d with coefficients of bit size $\leq \nu + d$ requires $\widetilde{\mathcal{O}}_B(d(d + \nu))$ bit operations [49,48] and the resulting polynomial has (coefficient) bit size $\mathcal{O}(\nu + d)$. Consequently, scaling the variable of the (resulting) polynomial by $\frac{1}{2}$ and computing the quotient by the binomial coefficients costs $\widetilde{\mathcal{O}}_B(d(\nu + d))$.

Therefore, the complexity of computing the Bernstein coefficients of f on the sub-interval $[a, \frac{a+b}{2}]$ is bounded by $\widetilde{\mathcal{O}}_B(d(\nu + d))$. By symmetry, inverting the order of the coefficients of f, we obtain the same bound for the coefficients of f on $[\frac{a+b}{2}, b]$, which ends the proof. $\square$

4 Subdivision Solver

Let $f = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x]$, such that $\mathtt{deg}(f) = d$ and $\mathcal{L}(f) = \tau$, and let f_{red} be its square-free part. We want to isolate the real roots of f, i.e. to compute intervals with rational endpoints that contain one and only one root of f, as well as the multiplicity of every real root.

In Fig. 1, we present the general scheme of the subdivision solver that we consider, augmented appropriately so that it also outputs the multiplicities. It uses an external function $V(f, I)$, which bounds the number of roots of f in the interval I. A real root isolation algorithm can be put under the subdivision solver concept of Fig. 1 if it can provide the $V(f, I)$ function. In the case of the Sturm solver, $V(f, I)$ returns the exact number (counted without multiplicities) of the real roots of f in I (Cor. 2.1). In the case of Bernstein solver, $V(f, I)$ is equal to the number of real roots of f in I (counted with multiplicities) modulo 2 (Prop. 3.1).

Separation bounds. An important quantity for the analysis of the subdivision solvers is a bound on the minimal distance, $\mathrm{sep}(f)$, between the roots of a univariate polynomial f (also called *separation bound*), or more generally on the product of distances between roots. We recall here classical results, slightly adapted to our context. For the separation bound it is known, e.g. [2,34,51], that $\mathrm{sep}(f) \geq d^{-\frac{d+2}{2}} (d+1)^{\frac{1-d}{2}} 2^{\tau(1-d)}$, thus $\log(\mathrm{sep}(f)) = \mathcal{O}(d\tau)$. The latter provides a bound on the bit size of the endpoints of the isolating intervals.

Recall, that Mahler's measure, see e.g. [34,51,2], of a polynomial f is $\mathcal{M}(f) = |a_d| \prod_{i=1}^{d} \max\{1, |\gamma_i|\}$, where a_d is the leading coefficient and γ_i are all the roots of f. We know that $\mathcal{M}(f) < 2^\tau \sqrt{d+1}$ [2,34]. Thus, the following inequality [2,34] holds:

$$\mathcal{M}(f_{red}) \leq \mathcal{M}(f) < 2^\tau \sqrt{d+1}. \tag{5}$$

For the minimum distance between two consecutive real roots of a square-free polynomial, Davenport-Mahler bound is known [13] (see also [27,30]). The conditions for this bound to hold were generalized by Du et al [14]. A similar bound, with less strict hypotheses, also appeared in [35]. Using (5) we can provide a bound similar to [27] for non square-free polynomials.

ALGORITHM: Real Root Isolation
Input: A polynomial $f \in \mathbb{Z}[x]$, such that $\mathtt{deg}(f) = d$ and $\mathcal{L}(f) = \tau$. Output: A list of intervals with rational endpoints, which contain one and only one real root of f and the multiplicity of every real root.
1. Compute the square-free part of f, i.e. f_{red} 2. Compute an interval $I_0 = (-B, B)$ with rational endpoints that contains all the real roots. Initialize a queue Q with I_0. 3. While Q is not empty do *a*) Pop an interval I from Q and compute $v := V(f, I)$. *b*) If $v = 0$, discard I. *c*) If $v = 1$, output I. *d*) If $v \geq 2$, split I into I_L and I_R and push them to Q. 4. Determine the multiplicities of the real roots, using the square-free factorization of f.

Fig. 1. Real root isolation subdivision algorithm

Theorem 4.1 (Davenport-Mahler bound revisited). *Let $A = \{\alpha_1, \ldots, \alpha_k\}$ and $B = \{\beta_1, \ldots, \beta_k\}$ be subsets of* distinct *(complex) roots of f (not necessarily square-free) such that $\beta_i \notin \{\alpha_1, \ldots, \alpha_i\}$ and $|\beta_i| \leq |\alpha_i|$, for all $i \in \{1, \ldots, k\}$. Then*

$$\prod_{i=1}^{k} |\alpha_i - \beta_i| \geq \mathcal{M}(f)^{-d+1} d^{-\frac{d}{2}} (\frac{\sqrt{3}}{d})^k.$$

The bound also holds when $\alpha_1 > \beta_1 = \alpha_2 > \beta_2 = \ldots \alpha_k > \beta_k := \alpha_{k+1}$, are distinct real roots of f.

Proof. Use [27] and (5). □

5 Complexity Analysis of Real Root Isolation

In this section, we bound the number of bit operations for isolating the real roots of a polynomial using the Sturm and the Bernstein solver. In what follows we present in detail the complexity of each step of the subdivision algorithm (see Fig. 1).

We consider the tree associated with a run of the subdivision algorithm on a polynomial f. Each node represents an interval. The root of the tree corresponds to the initial interval $I_0 = [\mathsf{a}, \mathsf{b}]$. The algorithm splits every interval which is not a leaf of the tree to two equal sub-intervals. The depth of a node of the tree (associated with an interval I) is $\log_2(|I_0|/|I|)$. This is also the number of subdivisions performed to obtain the sub-interval I from I_0. The number of steps (subdivisions) that the algorithm performs equals the total number of nodes of the subdivision tree, or in other words equals the number of intervals (sub-intervals of I_0) that are tested. Notice that the arguments are independent of the subdivision solver, Sturm or Bernstein in this paper, that we use.

5.1 Square-Free Factorisation [step 1]

The computation of f_{red} can be done in $\widetilde{\mathcal{O}}_B(d^2\tau)$ (Th. 2.4). Notice that $\mathcal{L}(f_{red}) = \mathcal{O}(d+\tau)$. In order to simplify notation, we assume that $d = \mathcal{O}(\tau)$, thus $\mathcal{L}(f_{red}) = \mathcal{O}(\tau)$. Notice also that after this computation, the Sturm-Habicht sequence $\mathbf{StHa}(f)$ is available. We do not need the complete sequence but only the quotient boot, thus this computation can be done in $\widetilde{\mathcal{O}}_B(d^2\tau)$ (Th. 2.2). However, we may also assume that the complete sequence is computed, with complexity $\widetilde{\mathcal{O}}_B(d^3\tau)$ (Th. 2.1), since this step is not the bottleneck of the algorithm.

5.2 Root Bounds and Initialization [step 2]

The Cauchy bound states that if α is a real root of f then $|\alpha| \leq B = 1 + \max\left(\left|\frac{a_{d-1}}{a_d}\right|, \ldots, \left|\frac{a_0}{a_d}\right|\right) \leq 2^\tau$. Various upper bounds are known for the absolute value of the real roots [2,51,49]. However, asymptotically the bit size of all the bounds is the same, i.e. $B \leq 2^\tau$. Thus, we can take $I_0 = [\mathsf{a}, \mathsf{b}]$, with $\mathsf{a} \leq -2^\tau$ and $\mathsf{b} \geq 2^\tau$.

For the Sturm solver, before starting the main loop, we may have to compute the Sturm-Habicht sequence of f, which costs $\widetilde{\mathcal{O}}_B(d^3\tau)$ (Th. 2.1).

For the Bernstein solver, we have to represent f_{red} in the Bernstein basis on $[\mathsf{a}, \mathsf{b}]$. This can be done in $\mathcal{O}(d^2)$ arithmetic operations and it produces coefficients of size $\mathcal{O}(d(d+\tau))$. The cost of this transformation is bounded by $\widetilde{\mathcal{O}}_B(d^3(d+\tau))$.

In both methods, the initialization step can be done in $\widetilde{\mathcal{O}}_B(d^3(d+\tau))$.

5.3 Computing $V(f, I)$ and Splitting [steps 3.a-d]

Suppose that the algorithm is at depth h of the subdivision tree. The tested interval, say I, has endpoints of bit size bounded by $\tau + h$, since each subdivision step increases the bit size by one.

Using the Sturm solver, we compute $V(f, I)$, Cor. 2.1, by evaluating $\mathbf{StHa}(f)$ over the endpoints of I. The cost of the evaluation is $\widetilde{\mathcal{O}}_B(d^2(\tau+h))$ (Th. 2.3). Then we split I, i.e. compute the middle point of it, in $\mathcal{O}_B(\tau+h)$.

Using the Bernstein solver, we compute $V(f, I)$ by counting the number of sign variations in the control coefficients of f in I. This can be done in $\mathcal{O}(d)$ operations. We denote by $\tau_0 = \mathcal{O}(d(d+\tau))$ (Sec. 5.2) a bound on the bit size of the coefficients of f in the Bernstein basis on the interval I_0. By Prop. 3.2, since we performed h subdivisions so far, starting from a polynomial with coefficients of bit size τ_0, the coefficients of f on I are of bit size $\tau_0 + dh$ and the complexity of the splitting operation is in $\widetilde{\mathcal{O}}_B(d(d+\tau_0+d\,h)) = \widetilde{\mathcal{O}}_B(d^2(d+\tau+h))$ (Sec. 5.2).

Consequently, for both solvers, steps 3.a-d can be performed with complexity $\widetilde{\mathcal{O}}_B(d^2(d+\tau+h))$.

5.4 Subdivision Tree Analysis [step 3]

In this section, we analyse the total number of subdivisions.

A bound on the number of the subdivision steps was derived in [30, Th. 5.5, 5.6], where in Rem. 5.7 the authors state: "The theorem (5.6) implies the dominance relations $hk \preceq n \log(nd)$ and $h \preceq n \log(nd)$ which can be used in an asymptotic analysis of Algorithm 1 when the ring S of the coefficients is $\mathbb{Z}$", where k is the number of internal nodes of depth h in the recursion tree of the subdivision algorithm based on Descartes' rule, n is the degree and d is the Euclidean norm of the polynomial. In [46, Th. 5], a $\mathcal{O}(d\tau + d \log d)$ bound is derived and, later on, [16] proved optimality under the mild assumption that $\tau = \Omega(\log d)$.

Our arguments for bounding the number of the subdivision steps are a combination and/or simplification of the arguments in [30,14,46]. Our proof (prop. 5.2) is simpler than the one in [16,46] since the handling of the subdivision tree stops at an earlier level and we use Th. 4.1 (as stated in [27] and [30]) without any modifications. We also simplify substantially the proof of [14], for Sturm solver.

We denote by $\mathcal{I}$ the set of intervals which are the parents of two leaves in the subdivision tree in Sturm (resp. Bernstein) solver. By construction, for $I \in \mathcal{I}$, $V(f, I) \geq 2$ but for the two sub-intervals I_L, I_R of I, $V(f, I_L)$ and $V(f, I_R)$ are in $\{0, 1\}$, since these intervals are leaves of the subdivision tree. Moreover, for the Sturm solver, it holds that $V(f, I) = 2$ and $V(f, I_L) = V(f, I_R) = 1$.

Notice that $|\mathcal{I}|$ is less than $V(f, I_0)$, since at each subdivision the sum of the variations of f on all the intervals cannot increase, for both methods (see [39,37] for the Bernstein solver). In particular, we have $|\mathcal{I}| \leq d$.

Let α_I (or β_I) be a, possibly complex, root of f whose real part is in I.

Proposition 5.1. *If $I \in \mathcal{I}$ then there exist two distinct (complex) roots $\alpha_I \neq \beta_I$ of f such that $|\alpha_I - \beta_I| < 2|I|$.*

Proof. Consider an interval $I \in \mathcal{I}$ which contains two leaves I_L, I_R of the subdivision tree. We have the following possibilities for the sign variation of f on the two sub-intervals I_L, I_R:

$(1, 1)$: for both solvers, there are two distinct real roots $\alpha \in I_L, \beta \in I_R$ in I and $|\alpha - \beta| \leq |I|$. This is the only case, which can happen in Sturm's solver.

$(0, 0)$: this may happen only in the Bernstein solver. Since the sign variations are $V(f, I) \geq 2$, the first circle theorem [37,2,30] implies that there exist two complex conjugate roots $\beta, \overline{\beta}$ in the disc $D(m(I), \frac{|I|}{2})$. Therefore, $|\beta - \overline{\beta}| \leq |I|$.

$(1, 0)$ **or** $(0, 1)$: this may also happen only in the Bernstein solver. There is a real root α in I. Since $V(f, I) \geq 2$, the second circle theorem [37,2,30] implies that there exists two complex conjugate roots $\beta, \overline{\beta}$ in the union of the discs $D(m(I) \pm \frac{1}{2\sqrt{3}}\mathbf{i}|I|, \frac{1}{\sqrt{3}}|I|)$, which is contained in a disc of diameter $2\,|I|$. Therefore $|\beta - \alpha| < 2|I|$.

Thus the proposition holds. □

In addition, we can prove the following result.

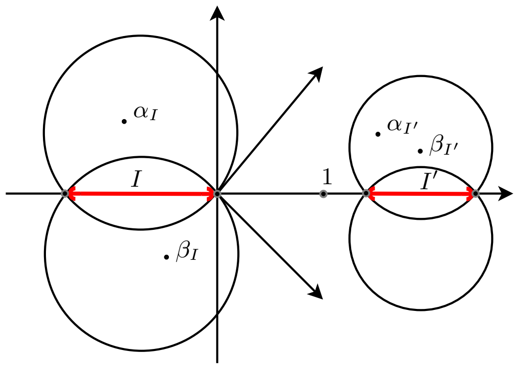

Fig. 2. Two disjoint intervals I and I' and the accociated two circles to each of them.

Lemma 5.1. *Let* $\{\alpha_I, \beta_I\} \cap \{\alpha_{I'}, \beta_{I'}\} \neq \emptyset$*, then* $I \cap I' \neq \emptyset$.

Proof. For the Sturm solver, this property is clear since $\alpha_I, \beta_I \in I$.

Let us consider the Bernstein subdivision method. We suppose that $I \cap I' = \emptyset$. Without loss of generality, we can assume in the proof that $I \neq I'$, $|I| \geq |I'|$, and that $\forall x \in I, \forall y \in I', x \leq y$.

Since the intervals are obtained by binary subdivision, we can assume that the distance between I and I' is at least $|I'|$. Using scaling and translation, we can assume that the right endpoint of I is 0 and that $I' = [1+u, 2+u]$ ($u \geq 0$). The tangents to the larger circles, containing I and the roots α_I and β_I at $(0,0)$, are $\frac{\sqrt{3}}{2}x \pm \frac{y}{2} = 0$ (see Fig. 2). We denote by R_I the union of the corresponding discs, so that $\alpha_I, \beta_I \in R_I$.

The center of the discs, whose union $R_{I'}$ contains the roots $\alpha_{I'}$ and $\beta_{I'}$, are $(\frac{3}{2} + u, \pm\frac{\sqrt{3}}{6})$ and their radius is $\frac{\sqrt{3}}{3}$ (see Fig. 2). A direct computation of the distance between these centers and the two tangent lines shows that $R_I \cap R_{I'} = \emptyset$. Consequently, if $I \cap I' = \emptyset$, then we conclude that $\{\alpha_I, \beta_I\} \cap \{\alpha_{I'}, \beta_{I'}\} = \emptyset$. $\square$

Let us number the intervals of $\mathcal{I}$ in increasing order and denote by $\mathcal{I}'$ the subset with an even index and by $\mathcal{I}''$ the subset with an odd index. By Lem. 5.1, the pairs $\{\alpha_I, \beta_I\}$ for $I \in \mathcal{I}'$ (resp. $\mathcal{I}''$) are disjoint. Thus, by Th. 4.1 (exchanging the role of α_I and β_I if necessary), we have

$$\prod_{I \in \mathcal{J}} |\alpha_I - \beta_I| \geq \mathcal{M}(f)^{-d+1} d^{-\frac{d}{2} - |\mathcal{J}|} \sqrt{3}^{|\mathcal{J}|}, \tag{6}$$

for $\mathcal{J} = \mathcal{I}'$ or $\mathcal{J} = \mathcal{I}''$. This is the key argument for the following result:

Proposition 5.2. *The number of subdivisions in both methods is in* $\mathcal{O}\left(d\tau + d\log(d)\right)$.

Proof. The number N of subdivisions equals the number of internal nodes in the subdivision tree. It is less than the sum of the depth of I, for $I \in \mathcal{I}$:

$$\begin{aligned} N &\leq \sum_{I\in\mathcal{I}} \log\frac{|\mathsf{b}-\mathsf{a}|}{|I|} \\ &\leq |\mathcal{I}|\log|\mathsf{b}-\mathsf{a}| - \sum_{I\in\mathcal{I}} \log|I| \\ &\leq |\mathcal{I}|\log|\mathsf{b}-\mathsf{a}| + |\mathcal{I}| - \sum_{I\in\mathcal{I}} \log|\alpha_I - \beta_I| \text{ (Prop. 5.1).} \end{aligned}$$

By (6), we have $-\sum_{I\in\mathcal{I}'} \log|\alpha_I - \beta_I| \leq (d-1)\log(\mathcal{M}(f)) + (\frac{d}{2} + |\mathcal{I}'|)\log d - |\mathcal{I}'|\log\sqrt{3}$. A similar bound applies for $\mathcal{I}''$.

As we can take $\mathsf{a} = -2^\tau, \mathsf{b} = 2^\tau$ (by Cauchy bound), $\log\mathcal{M}(f) \leq \tau + \frac{1}{2}\log(d+1)$ (Eq. 5) and $|\mathcal{I}'| + |\mathcal{I}''| = |\mathcal{I}| \leq d$, the number of internal nodes, N, in the subdivision tree is bounded by

$$\begin{aligned} N &< |\mathcal{I}| + |\mathcal{I}|\log|\mathsf{b}-\mathsf{a}| - \sum_{I\in\mathcal{I}} \log|\alpha_I - \beta_I| \\ &\leq d + d(\tau+1) + (d-1)(2\tau + \log(d+1)) + 2\,d\log d \\ &= \mathcal{O}\left(d\tau + d\log d\right). \end{aligned}$$

□

Remark 5.1. The constant in this bound on the number of subdivisions can be divided by 2, in Sturm method, by applying directly Th. 4.1 to α_I, β_I for $I \in \mathcal{I}$.

5.5 Multiplicities [step 4]

In order to compute the multiplicities of the roots, we compute the square-free factorization, i.e. a sequence of square-free coprime polynomials $(g_1, g_2, \ldots, g_m)$ with $f = g_1 g_2^2 \cdots g_m^m$ and $g_m \neq 1$. The algorithm of Yun [49] computes the square-free factorization in $\widetilde{\mathcal{O}}_B(d^2\tau)$. To be more specific the cost is twice the cost of the computation of $\mathbf{StHa}(f, f')$ [23]. Moreover, $\mathtt{deg}(g_i) = \delta_i \leq d$ and $\mathcal{L}(g_i) = O(d\tau)$ by Mignotte's bound [34], where $1 \leq i \leq m$.

At every isolating interval, one and only one g_i must have opposite signs at its endpoints, since g_i are square-free and pairwise coprime. If a g_i changes sign at an interval then the multiplicity of the real root that the interval contains is i. Each g_i can be evaluated over an isolating point in $\widetilde{\mathcal{O}}_B(\delta_i^2 d\tau)$, using Horner's rule. We can evaluate it over all the isolating points (there are at most $d+1$), in $\widetilde{\mathcal{O}}_B(\delta_i d^2\tau)$ [49,51]. Since $\sum_{i=1}^{m} \delta_i \leq d$ the overall cost is $\widetilde{\mathcal{O}}_B(d^3\tau)$.

5.6 Complexity of Real Root Isolation

In this section we prove that the bit complexity bound for the two subdivision solvers is $\widetilde{\mathcal{O}}_B(d^4\tau^2)$:

Theorem 5.1. *Let $f \in \mathbb{Z}[x]$, with $\mathtt{deg}(f) = d$ and $\mathcal{L}(f) = \tau$, not necessarily square-free. We can isolate the real roots of f and determine their multiplicities using Sturm or Bernstein solver in $\widetilde{\mathcal{O}}_B(d^4\tau^2)$. Moreover, the endpoints of the isolating intervals have bit size bounded by $\mathcal{O}(d\,\tau)$.*

Proof. In order to isolate the real roots of f, we first compute its square-free part (step 1). This can be done in $\widetilde{\mathcal{O}}_B(d^2\tau)$ arithmetic operations and yields a polynomial f_{red}, which coefficients are of bit size bounded by $\mathcal{O}(d+\tau)$ (see Sec. 5.1). This step is not necessary for the Sturm solver.

The initialisation step costs $\widetilde{\mathcal{O}}_B(d^3(d+\tau))$ (Sec. 5.2).

Then we run the main loop of the subdivision algorithm. The cost of a subdivision step at level h is $\widetilde{\mathcal{O}}_B(d^2(d+\tau+h))$ (Sec. 5.3).

By Prop. 5.2, the number of subdivisions and the depth h of any node of the subdivision tree is $\widetilde{\mathcal{O}}(d\,\tau)$. Therefore, the overall complexity of both subdivision solvers is $\widetilde{\mathcal{O}}_B(d^2(d+\tau+d\,\tau)\,d\,\tau) = \widetilde{\mathcal{O}}_B(d^4\tau^2)$. □

Remark 5.2. In Sec. 5.1 we assumed that $d = \mathcal{O}(\tau)$. If we drop this assumption then the complexity of real root isolation is $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau + d^4\tau^2)$. If $N = \max\{d, \tau\}$ then in both cases the complexity bound is $\widetilde{\mathcal{O}}_B(N^6)$.

6 Real Algebraic Numbers

The real algebraic numbers, i.e. those real numbers that satisfy a polynomial equation with integer coefficients, form a real closed field denoted by $\mathbb{R}_{alg} = \overline{\mathbb{Q}}$. From all integer polynomials that have an algebraic number α as root, the primitive one (the gcd of the coefficients is 1) with the minimum degree is called *minimal*. The minimal polynomial is unique (up to a sign), primitive and irreducible [51]. Since we use Sturm-Habicht sequences, it suffices to deal with algebraic numbers, as roots of any square-free polynomial and not as roots of their minimal ones. In order to represent a real algebraic number we choose the *isolating interval representation*.

Definition 6.1. *The isolating-interval representation of real algebraic number $\alpha \in \mathbb{R}_{alg}$ is $\alpha \cong (f(x), I)$, where $f(x) \in \mathbb{Z}[x]$ is square-free and $f(\alpha) = 0$, $I = [\mathsf{a}, \mathsf{b}]$, $\mathsf{a}, \mathsf{b}, \in \mathbb{Q}$, $\alpha \in I$ and f has no other root in I.*

Using the results of Sec. 2 and 3 we can compute the isolating interval representation of all the real roots of a polynomial f, with $\mathtt{deg}(f) = d$ and $\mathcal{L}(f) = \tau$, in $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ and the endpoints of the isolating intervals have bit size $\mathcal{O}(d\tau)$.

Comparison and sign evaluation. We can use Sturm-Habicht sequences in order to find the sign of a univariate polynomial, evaluated over a real algebraic number and to compare two algebraic numbers. We improve existing bounds by one factor.

Corollary 6.1. *Let $g(x) \in \mathbb{Z}[x]$, where $\mathtt{deg}(g) \leq d$ and $\mathcal{L}(g) = \tau$, and a real algebraic number $\alpha \cong (f, [\mathsf{a}, \mathsf{b}])$. We can compute $\mathrm{sign}(g(\alpha))$ in $\widetilde{\mathcal{O}}_B(d^3\tau)$.*

Proof. By Th. 2.5

$$W_{(f,g)}(\mathsf{a}) - W_{(f,g)}(\mathsf{b}) = \mathrm{sign}(g(\alpha)) \cdot \mathrm{sign}(f'(\alpha)),$$

and thus

$$\mathrm{sign}(g(\alpha)) = \big(W_{(f,g)}(\mathsf{a}) - W_{(f,g)}(\mathsf{b})\big) \cdot \mathrm{sign}(f^{'}(\alpha)).$$

Thus we need to perform two evaluations of $\mathbf{StHa}(f, g)$ over the endpoints of the isolating interval of α. The complexity of each is $\widetilde{\mathcal{O}}_B(d^3\tau)$ (Th. 2.3). The sign of $f^{'}(\alpha)$ can be computed as

$$\mathrm{sign}(f^{'}(\alpha)) = \mathrm{sign}(f(\mathsf{b})) - \mathrm{sign}(f(\mathsf{a})).$$

Notice that $f(\mathsf{a})f(\mathsf{b}) < 0$, since γ is the unique real root of f in $[\mathsf{a}, \mathsf{b}]$. It reasonable to assume that the signs of f over the endpoints of the isolating interval are known. If this is not the case then we can evaluate f over them, using Horner's rule, with complexity $\widetilde{\mathcal{O}}_B(d^3\tau)$, since we need d multiplications of numbers of bit size $\mathcal{O}(d^2\tau)$.

We conclude that the overall complexity of the operation is $\widetilde{\mathcal{O}}_B(d^3\tau)$. □

Corollary 6.2. *We can compare two real algebraic numbers in isolating interval representation in* $\widetilde{\mathcal{O}}_B(d^3\tau)$.

Proof. Let two algebraic numbers $\gamma_1 \cong (f_1(x), I_1)$ and $\gamma_2 \cong (f_2(x), I_2)$ where $I_1 = [\mathsf{a}_1, \mathsf{b}_1]$, $I_2 = [\mathsf{a}_2, \mathsf{b}_2]$. Let $J = I_1 \cap I_2$. When $J = \emptyset$, or only one of γ_1 and γ_2 belong to J, we can easily order the 2 algebraic numbers. If $\gamma_1, \gamma_2 \in J$, then $\gamma_1 \geq \gamma_2 \Leftrightarrow f_2(\gamma_1) \cdot f_2^{'}(\gamma_2) \geq 0$. We obtain the sign of $f_2(\gamma_2)$, using Cor. 6.1, thus the complexity of comparison is $\widetilde{\mathcal{O}}_B(d^3\tau)$. □

Simultaneous inequalities Let f, $A_1, \ldots, A_{n_1}$, $B_1, \ldots, B_{n_2}$, $C_1, \ldots, C_{n_3} \in \mathbb{Z}[x]$, with degree bounded by d and coefficient bit size bounded by τ. We wish to compute the number of and the real roots, γ, of f such that $A_i(\gamma) > 0$, $B_j(\gamma) < 0$ and $C_k(\gamma) = 0$ and $1 \leq i \leq n_1, 1 \leq j \leq n_2, 1 \leq k \leq n_3$. Let $n = n_1 + n_2 + n_3$.

Corollary 6.3. *There is an algorithm that solves the problem of simultaneous inequalities (SI) in* $\widetilde{\mathcal{O}}_B(d^4\tau \max\{n, \tau\})$.

Proof. First we compute the isolating interval representation of all the real roots of f in $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ (Th. 5.1). There are at most d. For every real root γ of f, for every polynomial A_i, B_j, C_k we compute sign $(A_i(\gamma))$, sign $(B_j(\gamma))$ and sign $(C_k(\gamma))$. Sign determination costs $\widetilde{\mathcal{O}}_B(d^3\tau)$ (Cor. 6.1) and in the worst case we must compute n of them. Thus the overall cost is $\widetilde{\mathcal{O}}_B(\max\{nd^4\tau, d^4\tau^2\})$.

This improves the known bounds by one or two factors in the bit complexity model.

7 Implementation and Experiments

In this section, we describe the package for algebraic numbers available in the library SYNAPS [36]. The purpose of this package is to provide a set of tools, for the manipulation of algebraic numbers, needed in applications such as Geometric modeling, and non linear computational geometry. In the problems encountered in these domains, the degree of the involved polynomials is not necessarily very high (≤ 100), but geometric operations require an intensive use of algebraic solvers.

For this reason, in this section we focus on univariate equations of *small degree* in opposition with the first sections, but the input bit size may be beyond machine precision. We analyse the behavior of our solvers, in this range of problems which appear in our geometric applications and for which the asymptotic bounds may not be pertinent indicators. We do not consider large degree problems, where memory management issues might influence the solving strategy.

In SYNAPS, there are several solver classes; their interface is as follows:

```
template < class T > struct SOLVER {
  typedef NumberTraits<T>::RT   RT;
  typedef NumberTraits<T>::FT   FT;
  typedef NumberTraits<T>::FIT  FIT;

  typedef UPolDse<T>            Poly;
  typedef root_of<T, Poly>      RO_t;
   ... };
```

where RT is the ring number type (typically $\mathbb{Z}$), FT is the field number type (typically $\mathbb{Q}$), FIT is the interval type, Poly is the univariate polynomial, RO_t is the type for real algebraic numbers, etc.

Algebraic numbers are of the form:

```
template <class T, class UPOL=UPolDse<T> >
struct root_of {
  NumberTraits<T>::Interval_t interval_;
  UPOL polynomial_;
 ... };
```

parametrized by the type of coefficients and univariate polynomials. This allows flexibility and an easy parametrisation of the code.

In order to construct a real algebraic number the user may select from several different univariate solvers, that we are going to describe hereafter. The other functionalities that we provide are the comparison, `bool compare(const RO_t& a, const RO_t& b)` and the evaluation of signs `int sign_at(const Poly& P, const RO_t& a)`, based on interval evaluation and if necessary on the computation of Sturm-Habicht sequences. This involves several additional functions for computing subresultant sequences with various methods (Euclidean, Subresultants, Sturm-Habicht, etc), for computing the GCD, the square-free part, etc. We also provide the four operations, i.e. $\{+, -, *, /\}$, of RO_t with RT's (integer type) and FT's (rational type).

Perhaps the most important operation is the construction of real algebraic numbers, i.e. real root isolation of univariate polynomials. Several subdivision

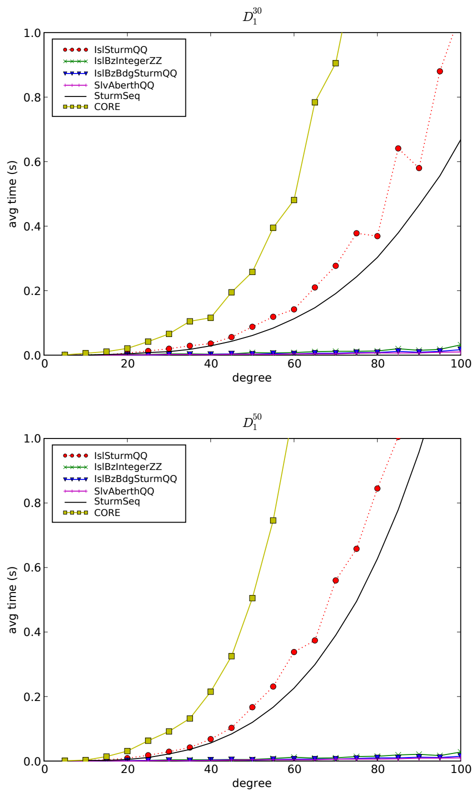

Fig. 3. Random polynomials of bit size 30 (top) and 50 (bottom) bits.

solvers have been tested for the construction of these algebraic numbers. We report here on the following solvers:

(S_1) `solve( f,IslSturm<ZZ>());`
(S_2) `solve( f,IslBzInteger<QQ>());`
(S_3) `solve( f,IslBzBdgSturm<QQ>());`

These solvers take as input, polynomials with integer or rational coefficients and output intervals with rational endpoints. All use the same initial interval.

S_1 (`IslSturmQQ` in the plots) is based on the construction of the Sturm-Habicht sequence and subdivisions, using rational numbers or large integers provided by the library GMP.

S_2 (`IslBzIntegerZZ` in the plots) is an implementation of the Bernstein subdivision solver, using integer coefficients. The polynomial is converted to the Bernstein representation on the initial interval, using rational arithmetic. Then, the coefficients are reduced to the same denominator, and the numerators are

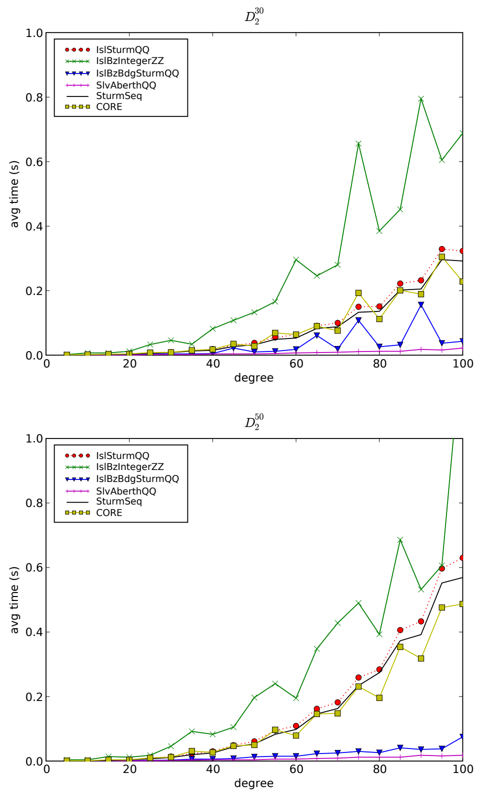

Fig. 4. Random polynomials with multiple roots of bit size 30 (top) and 50 (bottom) bits.

taken. Finally, the integer version of de Casteljau's algorithm, see Eq. (4), is applied at each subdivision step.

S_3 (`IslBzBdgSturmQQ` in the plots) is a combination of two solvers. In a first part, the polynomial is converted to the Bernstein representation on the initial interval, using rational arithmetic and its coefficients are rounded to `double` intervals. The Bernstein subdivision solver is applied on this interval representation and stops when it certifies the isolation of a root or when it is not possible to decide the existence and uniqueness of a root from the "sign" $(-, +, ?)$ of the interval coefficients. In this case, in order to complete the isolation process, the solver S_2 is used on the intervals which are suspect.

We also compare with the time needed for computing the square-free factorization of the tested polynomial (`SturmSeq` in the plots). Our implementation is based on Yun's algorithm and Sturm-Habicht sequences.

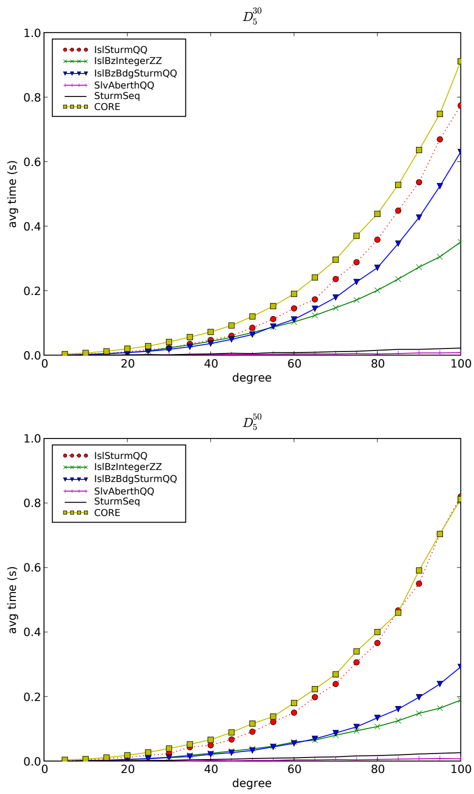

Fig. 5. Mignotte polynomials of bit size 30 (top) and 50 (bottom) bits.

We test against CORE [28] (`CORE` in the plots), and MPSOLVE, a numerical solver based on Aberth's method [5] and implemented by G. Fiorentino and D. Bini [6] (`SlvAberthQQ` in the plots), that are open source tools with real solving capabilities. CORE implements the Sturm solver. In order to isolate the real roots of a polynomial first it computes its square-free part and then performs isolation. MPSOLVE implements an iterative method for approximating the roots of a polynomial and the implementation is based on multiprecision floats. We set the output precision of the algorithm to 30 digits. The code of MPSOLVE is integrated in SYNAPS and called similarly to the other solver (S_1, S_2 and S_3), i.e `solve( f, Aberth<RR>())`.

Other libraries such as [25], or EXACUS with `Leda` [4], or RS [43], have not been tested, due to accessibility obstacles. Namely LEDA is a commercial software and RS, neither open source, is accessible as a binary code through its MAPLE (v. 9.5) interface, which we did not have at the time of the experiments. For experiments against these libraries and the package of Rioboo [42] in AXIOM, for degree ≤ 4, the reader may refer to [19].

Our data are polynomials of degree $d \in \{5, \ldots, 100\}$ and coefficient bit size $\tau \in \{10, 20, 30, 40, 50\}$ with various attributes. Namely D_1^τ denotes random polynomials with few real roots and D_2^τ random polynomials with multiple real roots. D_3^τ denotes polynomials with d (multiple) integer real roots and D_4^τ polynomials with d (multiple) rational real roots. D_5^τ denotes Mignotte polynomials, i.e. $a\left(x^d - 2(Kx-1)^2\right)$, where $K \in [5..30]$, D_6^τ polynomials that are the product of two Mignotte polynomials and D_7^τ Mignotte polynomials with multiple roots.

For reasons of space, we present timings, which are the average of 100 runs, only for D_1^{30}, D_1^{50}, D_2^{30}, D_2^{50}, D_5^{30}, D_5^{50}, D_7^{30} and D_7^{50}. The experiments were performed on an Pentium (2.6 GHz), using g++ 3.4.4 (Suse 10). We have to emphasize that we do not consider experimentation as a competition, but rather as a starting point for improving existing implementations.

For polynomials with few, distinct and well separated real roots, this is the case for D_1 and D_2 (see Fig. 3 and 4), S_1 is clearly the worst choice, since the huge time for the computation of the Sturm sequence dominates the time for its evaluation. In such data sets, S_2 or even approximate solvers are the solvers of choice, since not much output precision is needed in order to isolate the roots.

When there are roots that are very close and/or there are multiple roots, this is the case for D_5 and D_7 (see Fig. 5 and 6), then the computation time of the Sturm-Habicht sequence is negligible (for the experiments that we performed). In such cases a combined solver is the solver of choice, since it isolates the well separated roots and also provides good initial intervals for S_2, if needed. Special notice should be given to the bad behavior of the Bernstein solver (S_2) in the presence of multiple roots. The expected similar asymptotic behavior of Sturm and Bernstein subdivision solvers can be guessed on these experimentations, though the degree is not very high ($\leq$ 100). This applies in the worst case (Mignotte-like polynomials), whereas for random polynomials, the asymptotics of the two solvers seem to be different.

We have to mention that `CORE` does not compute the multiplicities of the real roots. In addition, for the Aberth solver (`SlvAberthQQ` in the plots), even though we specified its parameters in order to search for real roots only and detect multiplicities, since it is a numerical solver it must be given an output precision. In order to be sure in advance that we isolate all real roots, the output precision should be equal to the (theoretical) separation bound. In almost all cases, MPSOLVE is the fastest implementation.

For the exact solvers, we consider solver S_3, which is a combination of solvers, as the most promising option. It is the fastest on random instances and comparable to S_2 on all other instances. However, more theoretical work is needed so that we can provide some guarantee for the approximations. Another important issue is the implementation of the square-free factorization, which seems to be the bottleneck for all the exact algorithms. As we mentioned before, our implementation depends on the arithmetic with integers of arbitrary precision, provided by GMP. This implementation approach does not seem the right choice

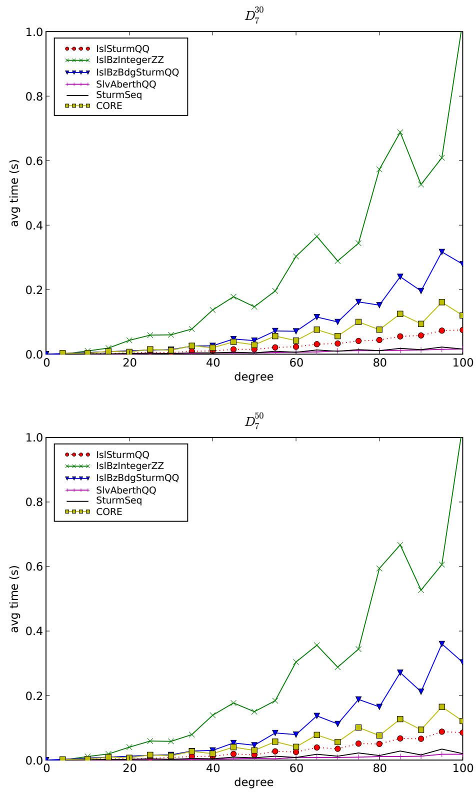

Fig. 6. Mignotte-like polynomials with multiple roots of bit size 30 (top) and 50 (bottom) bits.

for the square-free factorization algorithms. We believe that an implementation based on machine arithmetic combined with modular techniques will give much better results.

In some geometric problems, it is more important to have controlled approximation of the roots than to isolated them. This is the case in the following example where we want to draw a curve defined by an implicit equation. In this specific problem, the polynomial $f(x, y)$ is of degree 43 in each variable with coefficients of bit size 50 [8]. In order to get a picture of the implicit curve in the box $[a, b] \times [c, d]$, we solve the univariate polynomials $f(a + k\,\frac{(b-a)}{N}, y)_{k=0,\ldots,N-1}$ ($N = 200$) and then exchange the role of x and y. The subdivision is stopped, when the precision of 10^{-4} is reached, without checking the existence and uniqueness of the roots in the computed intervals.

Two types of solvers have been tested:

- The first one (SlvBzStd<double>) is a direct implementation of the Bernstein solver with double arithmetic. It produces the left part of Fig. 7. We see that in some regions, the solver is more sensible to numerical errors, and behaves almost "like a random generator of points".
- The second solver (SlvBzBdg<QQ>), similar to S_3, uses exact (rational) arithmetic to convert the input polynomial to its Bernstein representation. Then it normalises the coefficients and rounds up and down the rational numbers to the closest double numbers[4]. Then the main subdivision loop is performed on double interval arithmetic, extending the sign count to this context. If all the interval coefficients contain 0, we recompute the representation of the initial polynomial (using exact rational arithmetic) and run again the rounding and subdivision steps with double arithmetic, until we get the required precision. This produces the right part of Fig. 7.

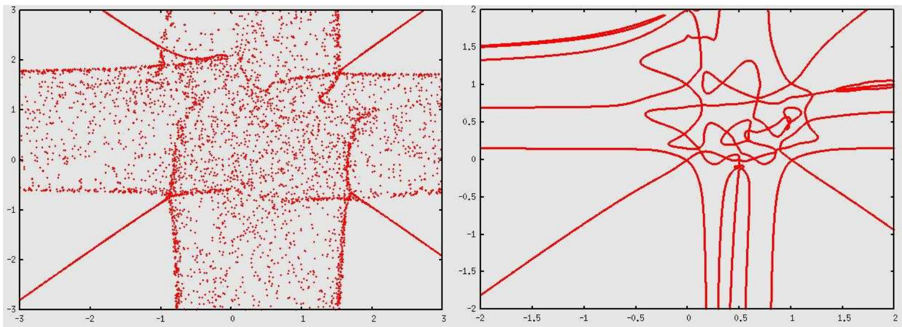

Fig. 7. Left: Approximation using doubles. Right: Approximation using Bernstein solver and interval arithmetic.

The Bernstein solver based on interval arithmetic and using this symbolic-numeric strategy can be applied efficiently (even for input polynomials with large coefficient size) to geometric problems, where (controlled) approximate results are sufficient. Its main advantage is that it exploits the performance of machine precision arithmetic for the main loop of the algorithm and the approximation properties of the Bernstein representation. Notice that the size of the problem is prohibitive for exact subdivision based solvers.

8 Current and Future Work

Our experiments imply that the combination of symbolic and numeric techniques leads to very promising implementations. Along these lines, we plan to

[4] For that purpose, one can use for instance the function get_double of MPFR (http://www.mpfr.org/) with correct rounding mode.

improve the existing implementation of solvers, so that we can approximate with guarantees the roots of a polynomial with exact coefficients. The applications of Bernstein methods to polynomials with approximate coefficients is also under investigation. We are also extending our package in SYNAPS so that it can handle computations in an extension field.

There are a lot of open questions concerning exact algorithms for real root isolation. Just to mention few of them: Is there any class of polynomials, with few real roots, such that the Bernstein solver performs $\mathcal{O}(d\tau)$ subdivision steps but the Sturm solver performs only a constant number? Does the $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ holds for complex root isolation? What is the expected complexity of the exact subdivision solvers?

The most important open problem for a theoretical and hopefully practical point of view is the following: Is there any exact (subdivision based) solver with complexity $\widetilde{\mathcal{O}}_B(d^3\tau)$, similar to the numerical solvers?

Acknowledgments

All authors acknowledge partial support by IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes).

References

1. A. Akritas. An implementation of Vincent's theorem. *Numerische Mathematik*, 36:53–62, 1980.
2. S. Basu, R. Pollack, and M-F.Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2003.
3. M. Ben-Or, D. Kozen, and J. H. Reif. The complexity of elementary algebra and geometry. *J. Comput. Syst. Sci.*, 32:251–264, 1986.
4. E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, L. Kettner, K. Mehlhorn, J. Reichel, S. Schmitt, E. Schömer, and N. Wolpert. EXACUS: Efficient and Exact Algorithms for Curves and Surfaces. In *ESA*, volume 1669 of *LNCS*, pages 155–166. Springer, 2005.
5. D. Bini. Numerical computation of polynomial zeros by means of Aberth's method. *Numerical Algorithms*, 13(3–4):179–200, 1996.
6. D. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, pages 127–173, 2000.
7. J. Canny. Improved algorithms for sign determination and existential quantifier elimination. *The Computer Journal*, 36(5):409–418, 1993.
8. F. Cazals, J.-C. Faugère, M. Pouget, and F. Rouillier. The implicit structure of ridges of a smooth parametric surface. Technical Report 5608, INRIA, 2005.
9. G. Collins. Subresultants and reduced polynomial remainder sequences. *J. ACM*, 14:128–142, 1967.
10. 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.

11. G. Collins and R. Loos. Real zeros of polynomials. In B. Buchberger, G. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 83–94. Springer-Verlag, Wien, 2nd edition, 1982.
12. M. Coste and M. F. Roy. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *J. Symb. Comput.*, 5(1/2):121–129, 1988.
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. 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 81–93, School of Science, Beihang University, Beijing, China, 2005.
15. 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.
16. A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the Descartes method. In *ISSAC '06: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pages 71–78, New York, NY, USA, 2006. ACM Press.
17. M. El Kahoui. An elementary approach to subresultants theory. *J. Symb. Comput.*, 35(3):281–292, 2003.
18. I. Emiris, A. Kakargias, M. Teillaud, E. Tsigaridas, and S. Pion. Towards an open curved kernel. In *Proc. Annual ACM Symp. on Computational Geometry*, pages 438–446, New York, 2004. ACM Press.
19. I. Emiris and E. Tsigaridas. Computing with real algebraic numbers of small degree. In *Proc. ESA*, LNCS, pages 652–663. Springer, 2004.
20. I. Emiris and E. Tsigaridas. Real solving of bivariate polynomial systems. In V. Ganzha, E. Mayr, and E. Vorozhtsov, editors, *Proc. Computer Algebra in Scientific Computing (CASC)*, LNCS, pages 150–161. Springer Verlag, 2005.
21. I. Emiris, E. Tsigaridas, and G. Tzoumas. The predicates for the Voronoi diagram of ellipses. In *Proc. 22th Annual ACM Symp. on Computational Geometry*, pages 227–236, Sedona, USA, 2006.
22. I. Emiris and E. P. Tsigaridas. Computations with one and two algebraic numbers. Technical report, ArXiv, Dec 2005.
23. K. Geddes, S. Czapor, and G. Labahn. *Algorithms of Computer Algebra*. Kluwer Academic Publishers, Boston, 1992.
24. L. González-Vega, H. Lombardi, T. Recio, and M.-F. Roy. Sturm-Habicht Sequence. In *ISSAC*, pages 136–146, 1989.
25. L. Guibas, M. Karavelas, and D. Russel. A computational framework for handling motion. In *Proc. 6th Workshop Algor. Engin. & Experim. (ALENEX)*, pages 129–141, Jan 2004.
26. L. E. Heindel. Integer arithmetic algorithms for polynomial real zero determination. *Journal of the Association for Computing Machinery*, 18(4):533–548, Oct. 1971.
27. J. Johnson. Algorithms for polynomial real root isolation. In B. Caviness and J. Johnson, editors, *Quantifier elimination and cylindrical algebraic decomposition*, pages 269–299. Springer, 1998.
28. V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A CORE library for robust numeric and geometric computation. In *15th ACM Symp. on Computational Geometry*, 1999.

29. W. Krandick. Isolierung reeller Nullstellen von Polynomen. In J. Herzberger, editor, *Wissenschaftliches Rechnen*, pages 105–154. Akademie-Verlag, Berlin, 1995.
30. W. Krandick and K. Mehlhorn. New bounds for the Descartes method. *JSC*, 41(1):49–66, Jan 2006.
31. J. M. Lane and R. F. Riesenfeld. Bounds on a polynomial. *BIT*, 21:112–117, 1981.
32. T. Lickteig and M.-F. Roy. Sylvester-Habicht Sequences and Fast Cauchy Index Computation. *J. Symb. Comput.*, 31(3):315–341, 2001.
33. H. Lombardi, M.-F. Roy, and M. Safey El Din. New Structure Theorem for Subresultants. *J. Symb. Comput.*, 29(4-5):663–689, 2000.
34. M. Mignotte. *Mathematics for Computer Algebra*. Springer-Verlag, 1992.
35. M. Mignotte. On the Distance Between the Roots of a Polynomial. *Appl. Algebra Eng. Commun. Comput.*, 6(6):327–332, 1995.
36. B. Mourrain, J. P. Pavone, P. Trébuchet, and E. Tsigaridas. SYNAPS, a library for symbolic-numeric computation. In *8th Int. Symposium on Effective Methods in Algebraic Geometry, MEGA*, Sardinia, Italy, May 2005. Software presentation.
37. B. Mourrain, F. Rouillier, and M.-F. Roy. *Bernstein's basis and real root isolation*, pages 459–478. Mathematical Sciences Research Institute Publications. Cambridge University Press, 2005.
38. B. Mourrain, J. Técourt, and M. Teillaud. On the computation of an arrangement of quadrics in 3d. *Comput. Geom.*, 30(2):145–164, 2005.
39. 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.
40. V. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and rootfinding. *J. Symbolic Computation*, 33(5):701–733, 2002.
41. D. Reischert. Asymptotically fast computation of subresultants. In *ISSAC*, pages 233–240, 1997.
42. R. Rioboo. Towards faster real algebraic numbers. In *Proc. ACM Intern. Symp. on Symbolic & Algebraic Comput.*, pages 221–228, Lille, France, 2002.
43. F. Rouillier and Z. Zimmermann. Efficient isolation of polynomial's real roots. *J. of Computational and Applied Mathematics*, 162(1):33–50, 2004.
44. M.-F. Roy and A. Szpirglas. Complexity of the Computation on Real Algebraic Numbers. *J. Symb. Comput.*, 10(1):39–52, 1990.
45. A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany, 1982.
46. V. Sharma and C. Yap. Sharp Amortized Bounds for Descartes and de Casteljau's Methods for Real Root Isolation. (unpublished manuscript), Oct 2005.
47. E. P. Tsigaridas and I. Z. Emiris. Univariate polynomial real root isolation: Continued fractions revisited. In Y. Azar and T. Erlebach, editors, *In Proc. 14th European Symposium of Algorithms (ESA)*, volume 4168 of *LNCS*, pages 817–828, Zurich, Switzerland, 2006. Springer Verlag.
48. J. von zur Gathen and J. Gerhard. Fast Algorithms for Taylor Shifts and Certain Difference Equations. In *ISSAC*, pages 40–47, 1997.
49. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge Univ. Press, Cambridge, U.K., 2nd edition, 2003.
50. J. von zur Gathen and T. Lücking. Subresultants revisited. *Theor. Comput. Sci.*, 1-3(297):199–239, 2003.
51. C. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.