A note on the complexity of univariate root isolation

Ioannis Z. Emiris, Elias P. P. Tsigaridas

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A note on the complexity of univariate root isolation

Ioannis Z. Emiris — Elias P. Tsigaridas

N° ????

Nobember 2006

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

INRIA
SOPHIA ANTIPOLIS

A note on the complexity of univariate root isolation

Ioannis Z. Emiris * , Elias P. Tsigaridas†

Thème SYM — Systèmes symboliques
Projet GEOMETRICA

Rapport de recherche n° ???? — Nobember 2006 — 18 pages

Abstract: This paper presents the average-case bit complexity of subdivision-based univariate solvers, namely those named after Sturm, Descartes, and Bernstein. By real solving we mean real root isolation. We prove bounds of $\widetilde{\mathcal{O}}_B(N^5)$ for all methods, where N bounds the polynomial degree and the coefficient bitsize, whereas their worst-case complexity is in $\widetilde{\mathcal{O}}_B(N^6)$. In the case of the Sturm solver, our bound depends on the number of real roots. Our work is a step towards understanding the practical complexity of real root isolation. This enables a better juxtaposition against numerical solvers, the latter having worst-case complexity in $\widetilde{\mathcal{O}}_B(N^4)$. Our approach extends to complex root isolation, where we offer a simple proof leading to bounds on the worst and average-case complexities of $\widetilde{\mathcal{O}}_B(N^7)$ and $\widetilde{\mathcal{O}}_B(N^6)$ respectively, where the latter is output-sensitive.

Key-words: polynomial equation, Sturm, Descartes' rule, Bernstein basis, subdivision, average case, bit complexity, separation bound

This work is partially supported by the european project ACS (Algorithms for Complex Shapes, IST FET Open 006413) and ARC ARCADIA (http://www.loria.fr/∼petitjea/Arcadia/)

* Department of Informatics and Telecommunications National Kapodistrian University of Athens, HELLAS

† Part of this work was done while the author was a PhD student at the Department of Informatics and Telecommunications of National Kapodistrian University of Athens, HELLAS

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

A note on the complexity of univariate root isolation

Résumé : Pas de résumé

Mots-clés : Pas de motclef

1 Introduction

One of the most important procedures in computer algebra and algebraic algorithms in general is root isolation of univariate polynomials. The goal of this algorithms is to compute isolating intervals in the real case, or boxes in the complex case that isolate the roots of the polynomial and to compute one such interval, or box, for every root.

We restrict ourselves to exact algorithms, i.e algorithms that perform arithmetic with rational numbers of arbitrary size.

The most well known and frequently used algorithms are the subdivision algorithms, either based on Sturm sequences, (STURM) or on Descartes' rule of sign (DESCARTES) or on Descartes' rule of sign and the properties of Bernstein basis representation (BERNSTEIN). The subdivision algorithms mimic the process of the binary search algorithm. They consider an initial interval that contains all the real roots and the repeatedly subdivide it until is certified that zero or one root is contained in the tested interval.

Quite recently it was proven that the complexity of STURM [6] and DESCARTES and BERNSTEIN [8] in the worst case is $\widetilde{\mathcal{O}}_B(d^4\tau)$, where d is the degree of the polynomial and τ the maximum coefficient it size. Moreover, the same proof for the number of steps that the algorithms perform holds for all the solvers, the polynomial need not be square-free and in the same complexity bound we can also compute the multiplicities of the real roots [9].

Another type of exact real root isolation solver is the CF solver [1], which we will not mention in this work and is based on the continued fractions expansion of the real roots. Quite recently it was proven [29], using the metric theory of the continued fractions of the real numbers that the expected complexity of CF is $\widetilde{\mathcal{O}}_B(d^4\tau^2)$, thus matches the current known bounds of the subdivision based algorithms. Moreover, by spreading the roots away the expected complexity bound can be improved to $\widetilde{\mathcal{O}}_B(d^3\tau)$ [28]. This algorithm we will denote it by MCF.

The direct competitors of the exact algorithms are the numerical algorithms that compute an approximation, up to a desired accuracy, of all the complex roots of a polynomial. These algorithms can be turned to isolation algorithm by requiring the accuracy to be equal to the separation bound of the polynomial. The ones with the best complexity up to now, namely $\widetilde{\mathcal{O}}_B(d^3\tau)$ [27, 23], are recursively splitting the polynomial until they compute linear factor that approximate sufficiently the real roots.

Even though the worst-case complexity bounds of the exact algorithms are worse than those of the numerical ones, recent implementations behave in practice in a rather satisfactory manner, e.g. [26, 12, 29, 9]. On the other hand, optimal numerical algorithms are very difficult to implement.

If we want to isolate the complex roots of a polynomial, i.e to compute isolating boxes in the complex plane that contain one and only one complex root, then the exact algorithms that are known are based on Sturm sequences [32, 24].

1.1 Main results and overview

In this paper we study the average complexity subdivision solvers for real and complex root isolation. The complexity bounds that we prove improve the ones of the worst case analysis and are a step towards undestanding the behavior of the exact subdivision based root isolation algorithms and possibly to reduce the gap with the numerical ones.

Table 1 shows the complexities of the solvers. The results for the Continued Fractions (CF and MCF) solvers appear in [29, 28]. All the other results on the expected complexity are new and to the best of our knowledge this is the first time that such results appear for the STURM and the DESCARTES/BERNSTEIN solver. In particular, the former has output-sensitive complexity since it depends on the number of real roots.

Solver	Average case	Worst case
STURM	$\widetilde{\mathcal{O}}_B(rd^2\tau^2)$	$\widetilde{\mathcal{O}}_B(d^4\tau^2)$
DESCARTES	$\widetilde{\mathcal{O}}_B(d^3\tau^2)$	$\widetilde{\mathcal{O}}_B(d^4\tau^2)$
BERNSTEIN	$\widetilde{\mathcal{O}}_B(d^3\tau^2)$	$\widetilde{\mathcal{O}}_B(d^4\tau^2)$
CF	$\widetilde{\mathcal{O}}_B(d^4\tau^2)$	
MCF	$\widetilde{\mathcal{O}}_B(d^3\tau)$	
Numeric		$\widetilde{\mathcal{O}}_B(d^3\tau)$

Table 1: Complexity bounds for univariate real root isolation.

For the problem of complex root isolation we simplify significantly the proof [6] for the number of subdivision that the algorithm perfrom and we also present an average case analysis.

The rest of the paper is structured as follows. The next subsection concentrates on notation. The following section describes the general procedure of a subdivision solver. In Sec. 3 we present the average bit size of the separation bound and we study the average complexity of the STURM and DESCARTES algorithm. In Sec. 4 we study the problem of complex root isolation and in Sec. 5 presents Kronecker's algorithm for completeness, but also in order to settle an open question by Collins and Loos on its average-case behavior. Finaly, we present some open questions.

1.2 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 $A = \sum_{i=1}^{d} a_i X^i \in \mathbb{Z}[X]$, $\mathsf{deg}(A)$ denotes its degree. We consider square-free polynomials except if explicitly stated otherwise. By $\mathcal{L}(A)$ we denote an upper bound on the bit size 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 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. Using FFT, $\mathsf{M}(\tau) = \mathcal{O}_B(\tau \lg^c \tau)$ for a suitable constant c. Δ is the separation bound of A, that is the smallest distance between two (complex) roots of A. Finally $N = \max\{d, \tau\}$.

Algorithm 1: SUBDIVISIONSOLVER$(A, \mathcal{I}_0)$

Input: $A \in \mathbb{Z}[X]$, $\mathcal{I}_0 = [\mathsf{a}, \mathsf{b}]$
Output: A list of isolatins intervals

```
1  INITIALIZATION_SM(A, I_0)
2  L ← ∅
3  Q ← ∅
4  Q ← PUSH(Q, {A, I_0})
5  while Q ≠ ∅ do
6      {f, I} ← POP(Q)
7      V ← COUNT_SM(f, I)
8      switch V do
9          case V = 0 continue
10         case V = 1 L ← ADD(L, I)
11         case V > 1
12             {f_L, I_L}, {f_R, I_R} ← SPLIT_SM(f, I)
13             Q ← PUSH(Q, {f_L, I_L})
14             Q ← PUSH(Q, {f_R, I_R})
15 RETURN L
```

In what follows we consider a polynomial $A \in \mathbb{Z}[X]$ such that $\mathsf{deg}(A) = d$ and $\mathcal{L}(A) = \tau$. The polynomial is square-free except if explicitly stated otherwise.

2 Subdivision Solvers

In this section we will desrcibe the general subdivision algorithm for isolating the real roots of a univariate polynomial. Recall that the subdivision-based algorithms mimic the process of the binary search algorithm.

The pseudo-code of the general subdivision algorithm is presented in Alg. 1. The input is a square-free polynomial $A \in \mathbb{Z}[X]$ and an interval $\mathcal{I}_0$ which contains the real roots of A which we wish to isolate. Usually $\mathcal{I}_0$ contains all real roots of A. The algorithm uses a stack Q that contains pairs of the form $\{f, \mathcal{I}\}$. The semantics of the pair is that we want to isolate the real roots of f contained in the interval $\mathcal{I}$.

In the pseudo-code of Alg. 1 we also use some external functions. To be more specific PUSH$(Q, \{f, \mathcal{I}\})$ inserts the pair $\{f, \mathcal{I}\}$ to the top of the stack Q and POP(Q) returns the pair that is in top of the stack and deletes it from it. Moreover, ADD$(L, \mathcal{I})$ inserts the interval $\mathcal{I}$ to the list L of the isolating intervals.

Finally, there are three sub-algorithms with index SM which have different specialization with respect to the subdivision algorithm applied. The first one INITIALIZATION$_{\text{SM}}$ does all

the necessary pre-processing, $\text{COUNT}_{\text{SM}}(f, \mathcal{I})$ returns the number (or an upper bound) of the real roots of f in $\mathcal{I}$. Finally, $\text{SPLIT}_{\text{SM}}(f, \mathcal{I})$ splits the interval $\mathcal{I}$ to two equal subintervals and possibly modifies polynomial f.

Notice that the complexity of the subdivision algorithm depends on how many times the **while**-loop (Line 5 of Alg. 1) is executed and on the cost of the sub-algorithms $\text{COUNT}_{\text{SM}}(f, \mathcal{I})$ and $\text{SPLIT}_{\text{SM}}(f, \mathcal{I})$. Moreover, at every step of the algorithm, since the tested interval is splitted to two equal sub-intervals, we may assume that the bit size of the endpoints of the intervals is augmented by one bit at every step of the algorithm. To be more specific, if we assume that the endpoints of the initial interval $\mathcal{I}_0$ have bit size τ, then at the h step of the algorithm the bit size of the endpoints of the tested intervals $\mathcal{I} \subseteq \mathcal{J}_0$ is $\tau + h$.

The various subdivision algorithms differ by the way they implement the three sub-algorithms. The most well known ones are STURM, DESCARTES and BERNSTEIN.

3 On the average case complexity

Let $\mathsf{disc}(A)$ be the discriminant and $\mathsf{lc}\,(A)$ the leading coefficient of A. Mahler's measure of a polynomial A is $\mathcal{M}(A) = |\,\mathsf{lc}\,(A)\,| \prod_{i=1}^{d} \max\{1, |\gamma_i|\}$, where γ_i are all the (complex) roots of A [2, 33, 18, 19]. Moreover $\mathcal{M}(A) \leq 2^{\tau}\sqrt{d+1}$. We prove the following theorem, which is based on a theorem by Mignotte [18], see also [9].

Theorem 1 (Davenport-Mahler-Mignotte) *Let $A \in \mathbb{Z}[X]$, with $\deg(A) = d$ and $\mathcal{L}\,(A) = \tau$, where $A(0) \neq 0$. Let Ω be any set of k couples of indices (i, j) such that $1 \leq i < j \leq d$ and let the non-zero (complex) roots of A be $0 < |\gamma_1| \leq |\gamma_2| \leq \cdots \leq |\gamma_d|$. Then*

$$2^k \mathcal{M}(A)^k \geq \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \geq 2^{k - \frac{d(d-1)}{2}}\, \mathcal{M}(A)^{1-d-k}\, \sqrt{\mathsf{disc}(A)}.$$

Proof: Consider the multiset $\overline{\Omega} = \{j | (i, j) \in \Omega\}$, $|\overline{\Omega}| = k$. We use the inequality

$$\forall\, a, b \in \mathbb{C} \quad |a - b| \leq 2 \max\{|a|, |b|\} \tag{1}$$

and the fact [18, 19] that for any root of A, $\frac{1}{\mathcal{M}(A)} \leq |\gamma_i| \leq \mathcal{M}(A)$. In order to prove the left inequality

$$\prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \leq 2^k \prod_{j\in\overline{\Omega}} |\gamma_j| \leq 2^k \max_{j\in\overline{\Omega}} |\gamma_j|^k \leq 2^k \mathcal{M}(A)^k.$$

Recall [33, 18] that $\mathsf{disc}(A) = \mathsf{lc}\,(A)^{2d-2} \prod_{i<j} (\gamma_i - \gamma_j)^2$. For the right inequality we consider the absolute value of the discriminant of A:

$$\begin{aligned} |\mathsf{disc}(A)| &= |\,\mathsf{lc}\,(A)\,|^{2d-2} \prod_{i<j} |\gamma_i - \gamma_j|^2 \\ &= |\,\mathsf{lc}\,(A)\,|^{2d-2} \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j|^2 \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j|^2 \quad \Leftrightarrow \\ \sqrt{|\mathsf{disc}(A)|} &= |\,\mathsf{lc}\,(A)\,|^{d-1} \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j| \quad \Leftrightarrow \\ \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| &= |\,\mathsf{lc}\,(A)\,|^{1-d} \Big(\prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j| \Big)^{-1} \sqrt{|\mathsf{disc}(A)|} \end{aligned} \tag{2}$$

We consider the product $\prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j|$ and we apply $\frac{d(d-1)}{2} - k$ times inequality (1), thus

$$\prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j| \leq 2^{\frac{d(d-1)}{2}-k}\, |\gamma_1|^0 |\gamma_2|^1 \cdots |\gamma_d|^{d-1} \Big(\prod_{j\in\overline{\Omega}} |\gamma_j| \Big)^{-1}.$$

From the definition of Mahler's bound we have

$$|\gamma_1|^0 |\gamma_2|^1 \cdots |\gamma_d|^{d-1} \leq |\mathcal{M}(A)/\,\mathsf{lc}\,(A)\,|^{d-1},$$

and since $\forall i$, $|\gamma_i| \geq \mathcal{M}(A)^{-1}$

$$\prod_{j\in\overline{\Omega}} |\gamma_j| \geq |\gamma_1|^k \geq \mathcal{M}(A)^{-k},$$

we conclude that

$$\prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j| \leq 2^{\frac{d(d-1)}{2}-k}\, \mathcal{M}(A)^{d+k-1}\, |\,\mathsf{lc}\,(A)\,|^{1-d}.$$

Combining the previous inequality with the last equation of (2) we finish the proof of the theorem. □

A similar theorem but with more strict hypotheses on the roots first appeared in [5] and the conditions were generalized in [8], see also [11, 14]. Th. 1 has a factor 2^{d^2} instead of d^d in [5, 8, 11], which plays no role when $d = \mathcal{O}(\tau)$ or when notation with N is used. Possibly a more involved proof may eliminate this factor using the techniques in [20]. Moreover, the loose hypotheses of Th. 1 dramatically simplify the proof for the number of steps of the sudivision-based solvers [8, 9].

Recall that the separation bound is the minimum distance between two complex roots of a polynomial, i.e $\Delta = \min_{i\neq j} |\gamma_i - \gamma_j|$, where γ_i are the (complex) roots of the polynomial

and $1 \leq i, j \leq d$. Let γ_{c_i} be a complex root of A that is closest to γ_i. We denote by Δ_i the quantity

$$\Delta_i = |\gamma_i - \gamma_{c_i}|.$$

A separating point for γ_i and γ_{c_i} is of magnitude at most $\frac{1}{2}\Delta_i$. Our goal is to bound the average bit size of the separation points, i.e the average bit size of $\lg \Delta_i$, which will denote by $\mathbf{A}[\lg \Delta_i]$.

Lemma 2 *The average bit size of a separation point is* $\mathcal{O}(d + \tau)$.

Proof: Our goal is to bound $\mathbf{A}[\lg \Delta_j]$, where $1 \leq j \leq d$ and

$$\mathbf{A}[\lg \Delta_i] = \frac{1}{d} \sum_{i=1}^{d} \lg \Delta_i = \frac{1}{d} \lg \prod_{i=1}^{d} \Delta_i. \tag{3}$$

First we consider the quantity $\prod_{j=1}^{d} \Delta_j$. In order to apply Th. 1 we should rearrange $\prod_{i=1}^{d} |\gamma_i - \gamma_{c_i}|$ so that the requirements on the indices of roots are fulfilled. This can not be achieved when symmetric factors occur, i.e the quantity contains factors of the form $|(\gamma_j - \gamma_{c_j})(\gamma_{c_j} - \gamma_j)|$, for some indices j. In order to avoid this case, we consider the positive integers k_1 and k_2 such that $k_1 + k_2 = d$ and we factorize the quantity as

$$\prod_{j=1}^{d} \Delta_j = \prod_{j=1}^{k_1} \Delta_j \prod_{j=1}^{k_2} \Delta_j, \tag{4}$$

where the factors do not contain symmetric products. Notice that since A is square-free $\mathsf{disc}(A) \geq 1$.

Let $k \leq d$ be such that the hypotheses of Th. 1, for the quantity $\prod_{i=1}^{d} \Delta_i$, are fulfilled. Thus

$$\begin{aligned}
2^k \mathcal{M}(A)^k &\geq \prod_{j=1}^{k} \Delta_j &&\geq 2^{k - \frac{d(d-1)}{2}} \mathcal{M}(A)^{1-d-k} \\
2^{k+k\tau} d^k &\geq \prod_{j=1}^{k} \Delta_j &&\geq 2^{k - \frac{d(d-1)}{2}} 2^{\tau(1-d-k)} d^{1-d-k} \\
k + k\tau + k \lg d &\geq \sum_{j=1}^{k} \lg \Delta_j &&\geq k - \frac{d(d-1)}{2} + \tau(1-d-k) + (1-d-k) \lg d.
\end{aligned}$$

If in the last relation we replace k once by k_1 and once by k_2, then we obtain the relations:

$$\begin{aligned}
k_1 + k_1\tau + k_1 \lg d &\geq \sum_{j=1}^{k_1} \lg \Delta_j &&\geq k_1 - \frac{d(d-1)}{2} + \tau(1-d-k_1) + (1-d-k_1) \lg d, \\
k_2 + k_2\tau + k_2 \lg d &\geq \sum_{j=1}^{k_2} \lg \Delta_j &&\geq k_2 - \frac{d(d-1)}{2} + \tau(1-d-k_2) + (1-d-k_2) \lg d.
\end{aligned}$$

and if we sum them, taking into account that $k_1 + k_2 = d$, then we have

$$d\tau + d(1 + \lg d) \quad \geq \quad \sum_{j=1}^{d} \lg \Delta_j \quad \geq -d^2 - 3d\tau + 2\tau + 2d + (2 - 3d) \lg d.$$

If we consider the last equation and (3) then we conclude that $\mathbf{A}[\lg \Delta_j] = \mathcal{O}(d + \tau)$, under the mild assumption that $\lg d = \mathcal{O}(\tau)$. □

3.1 The Sturm solver

The Sturm subdivision solver STURM counts the number of distinct real roots of A in an interval as follows. It evaluates a signed polynomial sequence of A and its derivate A' over the left endpoint of the interval and counts the number of sign variations. It does the same for the right enpoint and the difference of the sign variations is the number of real roots. Obviously the complexity of the algorithm is the number of steps that it performs time the complexity of each step.

Lemma 3 *The average number of subdivision steps of* STURM *is* $\mathcal{O}(r(d+\tau))$*, where* r *is the number of real roots.*

Proof: Recall that initially all the real roots are contained in the interval $[-2^\tau, 2^\tau]$. Since A has r real roots we must compute r separation points. The magnitude of the separation points is $\frac{1}{2}\Delta_i$, for $1 \leq i \leq r$, and we can compute each of them with $\lceil \lg \frac{2^{\tau+1}}{\Delta_j} \rceil$ subdivision steps, by performing binary search in the initial interval.

Let $\#(T)$ be the total number of subdivisions that we need in order to isolate all the real roots. Or in other words the number of steps that we need in order to compute all the separation points. Then

$$\#(T) = \sum_{j=1}^{r} \left\lceil \lg \frac{2^{\tau+1}}{\Delta_j} \right\rceil \leq 2r + r\tau - \sum_{j=1}^{r} \lg \Delta_j,$$

where the r summand is for the r possible roundings.

By Lem. 3, we have $\mathbf{A}[\lg \Delta_j] = \mathcal{O}(d + \tau)$, hence

$$\#(T) = 2r + r\tau + \sum_{j=1}^{r} \mathcal{O}(d + \tau) = r(\tau + 2) + \mathcal{O}(d + \tau) \sum_{j=1}^{r} 1,$$

and thus $\#(T) = \mathcal{O}(r(d + \tau))$ bounds the average number of subdivision steps. □

A similar proof holds for the worst case analysis [9], but in this case r should be replaced by d.

We need the following theorem:

Theorem 4 [17, 25] *The evaluation of a signed polynomial remainder sequence of A and A', $\mathbf{SR}(A, A')$, over a number a, where $\mathsf{a} \in \mathbb{Q} \cup \{\pm\infty\}$ and $\mathcal{L}(\mathsf{a}) = \sigma$, has bit complexity $\widetilde{\mathcal{O}}_B(d^2 \max\{\tau, \sigma\})$.*

Theorem 5 *The average complexity of* STURM *is $\widetilde{\mathcal{O}}_B(r(d^4 + d^3\tau + d^2\tau^2))$, or $\widetilde{\mathcal{O}}_B(rd^2\tau^2)$ if $d = \mathcal{O}(\tau)$, or $\widetilde{\mathcal{O}}_B(rN^4)$ where $N = \max\{d, \tau\}$.*

Proof: The complexity of the STURM solver is the number of subdivision steps that it performs times the cost of each subdivision. From Lem. 3, the average number of subdivisions is in $\mathcal{O}(r(d + \tau))$. Since the average bit size of the separation points is $\mathcal{O}_B(d + \tau)$ the cost of evaluating the signed polynomial sequence over them is $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$ (Th. 4).

We conclude that the overall complexity is $\widetilde{\mathcal{O}}_B(r(d^4 + d^3\tau + d^2\tau^2)$. □

Remark 6 *If we consider polynomials of degree d with coefficients that are independent standard normals then the expected number of real roots is $r \sim \frac{2}{\pi} \lg d$ [13], see also [7]. In this case the average complexity of* STURM *becomes $\widetilde{\mathcal{O}}_B(N^4)$, thus matching the complexity of numerical solvers [23, 27].*

As another, perhaps more natural, definition of random polynomials we can consider random polynomials of degree d with independent normally distributed coefficients, each with mean zero, but with the variace of the i-th coefficient equal to $\binom{d}{i}$. In this case the expected number of real roots is $r \sim \sqrt{d}$ [7] and the average complexity of STURM *is $\widetilde{\mathcal{O}}_B(N^{4.5})$.*

3.2 The Descartes solver

The Descartes subdivision solver DESCARTES relies on Descartes' rule of sign, which states that the number of sign variations in the coefficient list exceeds the number of positive real roots of A by an even number. In general this rule provides an overestimation on the number of positive real roots. However, when the number of sign variations is zero or one, then we get the exact number of positive real roots. Initially the polynomial is transformed so that its roots are contained in $(0, 1)$. In order to count the number real roots of A in an interval, we transform the polynomial to another polynomial, the real roots of which in $(0, \infty)$ correspond to the real roots of A in the initial interval. If the number of real roots is greater than one then we subdivide it and the we continue the algorithm on each subinterval. Notice that the polynomial transformations needed can be performed by shifting appropriately the variable X. The complexity of the algorithm is the number of intervals that it considers times the time needed for shifting the polynomial by a number whose magnitude is, in the worst case, proportional to the separation bound.

Descartes' rule of sign is independent of the basis that the polynomial is represented. This allows a variant of DESCARTES, which is based on the Bernstein representation of polynomials [16, 22, 21] and is amenable to very efficient implementations [12]. In what follows we will refer only to DESCARTES solver but the results also hold for BERNSTEIN.

Lemma 7 *The average number of subdivision steps of* DESCARTES *is* $\mathcal{O}(d^2 + d\tau)$.

Proof: We consider the subdivision tree, T, that is formed during the execution of the algorithm. Each node of the tree corresponds to an interval that the algorithm tests for roots and a polynomial. The root of the tree corresponds to the initial polynomial and interval. The number of nodes of the tree is the number of steps that the algorithm performs. Since the algorithm terminates all the leaves of the tree correspond to polynomials with either on or zero sign variations.

We consider all the nodes of the tree that are parents of two leaves. The polynomials that correspond to these nodes have ≤ 2 sign variations. We denote the set of these nodes by $\mathcal{J}$ and notice that $|\mathcal{J}| \leq d$, since there are at most d complex roots and at after a subdivision step the number of sign variations can not increase.

If $I \in \mathcal{J}$, then the number of subdivisions that the algorithm performs in order to compute it, is $\lg \frac{2^{\tau+1}}{|I|}$. If we consider the paths from the root of the subdivision tree to all $I \in \mathcal{J}$, then the sum of all the nodes bounds bounds the number of nodes of the the subdivision tree, $\#(T)$. Thus

$$\#(T) \leq \sum_{I \in \mathcal{J}} \lg \frac{2^{\tau+1}}{|I|}.$$

It can be proven [9] that each $I \in \mathcal{J}$ contains two, possible complex roots, such that $|\gamma_i - \gamma_j| \leq 2|I|$. But since $|\gamma_i - \gamma_j| \geq \Delta_i$ we have that $\Delta_i \leq 2|I|$. If we also take into account that $|\mathcal{J}| \leq d$, then

$$\#(T) \leq \sum_{I \in \mathcal{J}} \lg \frac{2^{\tau+1}}{|I|} \leq \sum_{i=1}^{d} \lg \frac{2^{\tau+2}}{\Delta_i} \leq d(\tau+2) - \sum_{i=1}^{d} \lg \Delta_i.$$

By Lem. 3, we have $\mathbf{A}[\lg \Delta_j] = \mathcal{O}(d+\tau)$, hence

$$\#(T) \leq d(d+\tau) + \mathcal{O}(d+\tau) \sum_{i=1}^{d} 1.$$

and thus the average number of subdivision steps is $\mathcal{O}(d^2 + d\tau)$. □

An important observation concerning DESCARTES is that the algorithm depends on all the roots of the polynomial, real and complex, thus we are not able to prove a bound on the number of the subdivision steps that wil depend on the number of the real roots.

We should also mention that for the worst case analysis the proof of Lem. 7 is almost the same. The only difference is that we have to bound the quantity $\sum_{i=1}^{d} \lg \Delta_i$ using Th. 1 instead of Lem. 7. This approach simplifies even further the proof appeared in [9].

To complete the analysis we will need the following theorem:

Theorem 8 (Fast Taylor shift) [30] *Let* $A \in \mathbb{Z}[X]$, *with* $\deg(A) = d$ *and* $\mathcal{L}(A) = \tau$ *and let* $a \in \mathbb{Z}$, *such that* $\mathcal{L}(a) = \sigma$. *Then the cost of computing* $B = A(a+X) \in \mathbb{Z}[X]$ *is* $\mathcal{O}_B(\mathsf{M}\left(d^2 \lg d + d^2\sigma + d\tau\right))$. *Moreover* $\mathcal{L}(B) = \mathcal{O}(\tau + d\sigma)$.

Theorem 9 *The average complexity of* DESCARTES *is* $\widetilde{\mathcal{O}}_B(d^5 + d^4\tau + d^3\tau^2)$, *or* $\widetilde{\mathcal{O}}_B(N^5)$ *where* $N = \max\{d, \tau\}$.

Proof: The complexity of the algorithm is the number of subdivision steps that performs times the cost of each subdivision. The number of subdivisions is $\mathcal{O}(d^2 + d\tau)$. The cost of each subdivision step is dominated by the cost of translating the polynomial. Notice that the bit size of the initial polynomial is $O(d\tau)$ since we transformed so that its roots to be in $(0, 1)$. In the average case we have to translate the polynomial by a number of bit size $\mathcal{O}(d + \tau)$ (Lem. 2). The cost of this computation is $\widetilde{\mathcal{O}}_B(d^3 + d^2\tau)$. Thus the overall cost, in average case is $\widetilde{\mathcal{O}}_B(d^5 + d^4\tau + d^3\tau^2)$. □

4 Complex root isolation

Theorem 1 allows us to prove, with simple arguments, a bound on the number of subdivisions needed to isolate the complex roots of a real or Gaussian polynomial. The algorithm is based on Sturm sequences, and on the computation of the Cauchy index; for a complete description, the reader may refer to [24, 32] and for a more didactic approach to [33].

The main idea of the algorithm is the following. Suppose that we have a method to count the correct number of distince complex roots in a square in the complex plane. If the number of complex roots is > 1 then the square is subdivided to four (equal) squares using the center point of the original square and the algorithm continues on each box.

Suppose that initialy all the complex roots are in a square of side B. This is a bound for all the complex roots, it can easily be computed [19, 18, 33] and typically is $B = 2^{\tau+1}$. At the h step of the algorithm we have to check for complex roots in squares that have sides equal to $B/2^h$.

We can consider the process of the algorithm as a ternary tree, where each node holds a square and the root of the tree holds the intial one. Each leaf of the tree contains contains a square that isolates a complex root of the polynomial and since there are at most d complex roots, this is also the number of the leaves of the tree. The squares that correspond to the leaves of the tree have sides of length equal to Δ_j and the number of nodes from a leaf to the root of the tree is

$$\left\lceil \log \frac{R}{\Delta_j} \right\rceil .$$

The number of subdivisions, $\#(T)$, that the algorithm perfroms equals the number of nodes of the subdivision tree, which is

$$\#(T) = \sum_{j=1}^{d} \left\lceil \lg \frac{B}{\Delta_j} \right\rceil = 2d + d\tau - \sum_{j=1}^{d} \lg \Delta_j = 2d + d\tau - \lg \prod_{j=1}^{d} \Delta_j. \tag{5}$$

It remains to bound the quantity $\prod_{j=1}^{d} \Delta_j$. For this we use Th. 1. Recall that the hypotheses of the theorem are not fullfilled when symmetric products occur, that we factorize

quantity as $\prod_{i=1}^{d} \Delta_i = \prod_{i=1}^{k_1} \Delta_i \prod_{i=1}^{k_2} \Delta_i$, where $k_1 + k_2 = d$ and the factors are such that no symmetric products occur.

By applying to each factor Th. 1 and by taking into account that $\mathsf{disc}(A) \geq 1$ we have that

$$\prod_{i=1}^{d} \Delta_i \geq 2^{d^2} \, \mathcal{M}\,(A)^{2-3d} \geq 2^{d^2} \, (2^{\tau}\sqrt{d+1})^{2-3d}$$

since $\mathcal{M}\,(A) \leq 2^{\tau}\sqrt{d+1}$. If we combine the last equation with (5) we conclude that the total number of subdivision is $\mathcal{O}(d^2 + d\tau)$. Thus we can state the following lemma:

Lemma 10 *The number of subdivisions for complex root isolation is* $\mathcal{O}(d^2 + d\tau)$.

The proof of the previous theorem simplifies significantly the proof appeared in [6] where an amortized-like argument is used. Moreover, our bound on the number of subdivisions has a factor d^2 instead of a factor $d \lg d$, which as in the real root case plays no role when the initial polynomial is not square-free or when the notation with N is used.

The only thing that remains is in order to complete the algorithm is to describe how we count the number or complex roots inside a box in the complex plane. For this we will follow Wilf [32].

Let $\mathbf{SR}(f, g)$ denote a signed polynomial remainder sequence of the polynomials f and g and $\mathtt{VAR}(\mathbf{SR}(f, g)\,;\,\mathsf{a}))$ the number of sign variation occur when we evaluate the sequence over a rational number a. The Cauchy index of the real function g/f in an interval $[\mathsf{a}, \mathsf{b}]$ is

$$I_{\mathsf{a}}^{\mathsf{b}}(g/f) = \mathtt{VAR}(\mathbf{SR}(f, g)\,;\,\mathsf{a})) - \mathtt{VAR}(\mathbf{SR}(f, g)\,;\,\mathsf{b})).$$

Consider a box in the complex plane with vertices, q_1, q_2, q_3, q_4 ($q_5 = q_1$) in counter-clockwise order, centered around a point p. We expand the polynomial A about the point Q_k, i.e we apply the transformation $X \mapsto a_k + \mathbf{i}^{k-1} X$, where $\mathbf{i} = \sqrt{-1}$. Let the resulting polynomial be

$$A_k(X) = \sum_{j=o}^{d} (\mathsf{a}_j - \mathbf{i}\,\mathsf{b}_j) X^j = \sum_{j=0}^{d} \mathsf{a}_i \, X^j + \mathbf{i} \sum_{j=0}^{d} \mathsf{b}_i \, X^j.$$

Moreover, let

$$A_{k1}(X) = \sum_{j=0}^{d} \mathsf{a}_i \, X^j, \quad A_{k2}(X) = \sum_{j=0}^{d} \mathsf{b}_i \, X^j.$$

Then the number of complex roots of A inside the box is

$$\frac{1}{2} \sum_{i=1}^{d} (\mathtt{VAR}(\mathbf{SR}(A_{k1}, A_{k2})\,;\,|q_{k+1} - q_k|) - \mathtt{VAR}(\mathbf{SR}(A_{k1}, A_{k2})\,;\,0)).$$

In order to compute the overall complexity of the algorithm, we have to take into account that, at each step the algorithm evaluates a signed polynomial remainder sequence of

polynomials of degree d and coefficient bit size $\mathcal{O}(d\tau)$. This is so because at each step we have to translate the polynomial. In the worst case we have to translate it by a number of bit size equal to the bit size of the separation bound, i.e $\mathcal{O}(d\tau)$, thus by Th. 8 its bit size will be $\mathcal{O}(d^2\tau)$. The computation of the Cauchy index corresponds to the evaluation of a signed polynomial remainder sequence over a rational number, which has bit size at most $\mathcal{O}(d\tau)$. Using Th. 4 this evaluation costs $\widetilde{\mathcal{O}}_B(d^4\tau)$. If we multiply the previous cost by the number of subdivisions we conclude that the overall cost is $\widetilde{\mathcal{O}}_B(d^6\tau + d^5\tau^2)$, or $\widetilde{\mathcal{O}}_B(N^7)$.

4.1 On the average complexity of complex root isolation

In order to average complexity of complex root isolation we will use the same techniques that we used in real root isolation. Recall that the average bit size of a separation point is $\mathcal{O}(d + \tau)$ (Lem. 2).

As we did in the proof of Lem. 3 we assume that the number of complex roots is c and by direct computation we can easily prove that the number of subdivisions is $\mathcal{O}(c(d + \tau))$. Using again Lem. 3 the cost of each step is $\widetilde{\mathcal{O}}_B(d^4 + d\tau)$ and thus the overall cost in the average case is $\widetilde{\mathcal{O}}_B(c(d^5 + d^4\tau + d^3\tau^2))$, or $\widetilde{\mathcal{O}}_B(cN^5)$.

5 Kronecker's algorithm

Kronecker [15], see also [4], presented an algorithm, hereafter called KRONECKER, for real root isolation of square-free polynomials. The algorithm depends on Rolle's theorem: if a square-free polynomial has a unique real root in an interval, then it changes sign at the endpoints of the interval. The algorithm is exponential but we discuss it for two reasons: First, in order to answer an open question posed by Collins and Loos [4] about the expected number of intervals that the algorithm tests. Second, for completeness.

The input of the algorithm is a square-free polynomial A and its output is a list of the isolating intervals of the real roots. First the algorithm computes an absolute bound on the real roots. All bounds are the same asymptotically, so $B = 2^\tau$, without loss of generality. Next the algorithm computes the theoretical separation bound $\Delta = 2^{-\mathcal{O}(d\tau)}$. The main loop of the algorithm consists in splitting the initial interval $[-B, B]$ to sub-intervals of length Δ and to check, using Rolles' theorem, which of them contain a real root of A.

The complexity of the algorithm depends on the number of times its main loop is executed. Or in other words on the number of intervals that the algorithm tests. Since the initial interval is $[-2^\tau, 2^\tau]$ its length is $2^{\tau+1}$. Thus the algorithm tests $2^{\mathcal{O}(d\tau)}$ intervals. At every step KRONECKER evaluates A over a rational number of bit size at most $\mathcal{O}(d\tau)$. The complexity of the evaluation is $\widetilde{\mathcal{O}}_B(d^3\tau)$ using Horner's scheme and fast multiplication algorithms [31, 3]. This leads to the following known theorem:

Theorem 11 (Kronecker) [4] *Let a square-free polynomial $A_{red} \in \mathbb{Z}[X]$ such that $\mathsf{deg}(A_{red}) = d$ and $\mathcal{L}(A_{red}) = \tau$. Algorithm* KRONECKER *isolates all real roots of A_{red} in $\widetilde{\mathcal{O}}_B(2^{d\tau}d^3\tau)$.*

There is an obvious improvement to KRONECKER algorithm. At each step, instead of computing the evaluation of A over the two endpoints of one interval, one can compute simultaneously the evaluation of A over the endpoints of $\frac{d}{2}$ intervals, with a logarithmic overhead on the asymptotic complexity [33, 31]. Unfortunately, this improvement does not tackle the exponential behavior of the algorithm, since we still have to test $2^{\mathcal{O}(d\tau)}/d = 2^{\mathcal{O}(d\tau)}$ intervals.

Collins and Loos [4] posed the question about the expected number of intervals that KRONECKER tests. We will answer this question under the assumption that *each interval may contain a real root of A with equal probability.*

If we assume that the number of real roots, say r, it is not known and that $r < d$, then the algorithm, always, must test all the intervals since it is not possible to know when to stop.

In what follows we assume that we know in advance the exact number of real roots, $r \leq d$. Let K be the number of the intervals that we have to test. All the possible positions of the real roots are $\binom{K}{r}$. The number of intervals that we have to test depends on the position of the real root with the biggest magnitude. The dominant real root can be anywhere from position r to position K. If it is in position ℓ, where $r \leq \ell \leq K$, then the algorithm tests ℓ intervals. There are $\binom{\ell-1}{r}$ such cases. Thus the expected value of the tested intervals is

$$\frac{\sum_{\ell=r}^{K} \binom{\ell-1}{r-1} \ell}{\binom{K}{r}} = \frac{r\,(K+1)}{r+1}.$$

If $r = 1$, then obviously there is no need to run KRONECKER algorithm since we already have an isolating interval for the real root, i.e $[-2^\tau, 2^\tau]$. However, if we run this naive algorithm anyway, for example because to we want to find a $2^{-\mathcal{O}(d\tau)}$ approximation of the real root, then the expected number of intervals that we have to test is $\frac{K+1}{2}$. This number coincides with the expected number of comparisons needed in order to locate an element in an array using serial search [10].

6 Open questions

The number of steps of the STURM solver depends on the number of real roots. If there are no real roots, then STURM identifies this without performing any subdivision. This is not the case for DESCARTES: the number of steps that it performs depends on all complex roots of the polynomial. A relevant open question is whether there exists a class of polynomials with no real roots, such that the number of steps of DESCARTES is $\mathcal{O}(d\tau)$ or even $\mathcal{O}(\tau)$?

Is there a exact algorithm for real root isolation with complexity $\widetilde{\mathcal{O}}_B(N^4)$ or even $\widetilde{\mathcal{O}}_B(N^5)$?

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

[4] G.E. Collins and R. Loos. Real zeros of polynomials. In B. Buchberger, G.E. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 83–94. Springer-Verlag, Wien, 2nd edition, 1982.

[5] 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.

[6] 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.

[7] A. Edelman and E. Kostlan. How may zeros of a random polynomial are real? *Bulletin of American Mathematical Society*, 32(1):1–37, Jan 1995.

[8] 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.

[9] 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*, LNCS (to appear). Springer Verlag, 2006. also available in www.inria.fr/rrrt/rr-5897.html.

[10] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, Mass., 1989.

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

[12] J. R. Johnson, W. Krandick, K. Lynch, D. Richardson, and A. Ruslanov. High-performance implementations of the Descartes method. In *ISSAC '06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 154–161, New York, NY, USA, 2006. ACM Press.

[13] M. Kac. On the average number of real roots of a random algebraic equation. *Bulletin of American Mathematical Society*, 49:314–320 and 938, 1943.

[14] W. Krandick. Isolierung reeller nullstellen von polynomen. In J. Herzberger, editor, *Wissenschaftliches Rechnen*, pages 105–154. Akademie-Verlag, Berlin, 1995.

[15] L. Kronecker. Über den zahlbergriff. *Crelle J. reine und angew. Mathematik*, 101:337–395, 1887.

[16] J. M. Lane and R. F. Riesenfeld. Bounds on a polynomial. *BIT*, 21:112–117, 1981.

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

[18] M. Mignotte. *Mathematics for computer algebra*. Springer-Verlag, New York, 1991.

[19] M. Mignotte and D. Stefanescu. *Polynomials: An algorithmic approach*. Springer, 1999.

[20] Maurice Mignotte. On the Distance Between the Roots of a Polynomial. *Appl. Algebra Eng. Commun. Comput.*, 6(6):327–332, 1995.

[21] 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.

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

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

[24] James R. Pinkert. An exact method for finding the roots of a complex polynomial. *ACM Trans. Math. Softw.*, 2(4):351–363, 1976.

[25] D. Reischert. Asymptotically fast computation of subresultants. In *ISSAC*, pages 233–240, 1997.

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

[27] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany, 1982.

[28] E. P. Tsigaridas and I. Z. Emiris. On the complexity of real root isolation using Continued Fractions. (submitted for journal publication), Sep 2006.

[29] 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.

[30] J. von zur Gathen and J. Gerhard. Fast Algorithms for Taylor Shifts and Certain Difference Equations. In *ISSAC*, pages 40–47, 1997.

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

[32] Herbert S. Wilf. A global bisection algorithm for computing the zeros of polynomials in the complex plane. *J. ACM*, 25(3):415–420, 1978.

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

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399