On the complexity of real root isolation using Continued Fractions

Elias P. Tsigaridas

INRIA-LORIA Lorraine, France

Ioannis Z. Emiris

Department of Informatics and Telecommunications, National Kapodistrian University of Athens, Hellas

Abstract

We present algorithmic, complexity and implementation results concerning real root isolation of integer univariate polynomials using the continued fraction expansion of real algebraic numbers. One motivation is to explain the method's good performance in practice. We derive an expected complexity bound of $\widetilde{\mathcal{O}}_B(d^6 + d^4\tau^2)$, where d is the polynomial degree and τ bounds the coefficient bit size, using a standard bound on the expected bit size of the integers in the continued fraction expansion, thus matching the current worst-case complexity bound for real root isolation by exact methods (Sturm, Descartes and Bernstein subdivision). Moreover, using a homothetic transformation we improve the expected complexity bound to $\widetilde{\mathcal{O}}_B(d^3\tau)$. We compute the multiplicities within the same complexity and extend the algorithm to non square-free polynomials. Finally, we present an open-source `C++` implementation in the algebraic library SYNAPS, and illustrate its completeness and efficiency as compared to some other available software. For this we use polynomials with coefficient bit size up to 8000 bits and degree up to 1000.

Key words: continued fraction, real root isolation, complexity, synaps

Email addresses: `elias.tsigaridas(AT)loria.fr` (Elias P. Tsigaridas), `emiris(AT)di.uoa.gr` (Ioannis Z. Emiris).

URLs: `www.loria.fr/~tsigarie` (Elias P. Tsigaridas), `www.di.uoa.gr/~emiris` (Ioannis Z. Emiris).

Preprint submitted to Elsevier *3 September 2007*

1 Introduction

Real root isolation of univariate integer polynomials is a fundamental problem in computer algebra as well as in many applications ranging from computational geometry to quantifier elimination. The problem consists in computing intervals with rational endpoints which contain exactly one real root of the polynomial and have such an interval for every real root. In this paper we consider an algorithm for real root isolation based on the continued fraction expansion (from now on called CF) of *real algebraic numbers*. Recall that such a number is a real root of an integer polynomial.

A major motivation is to explain the algorithm's good performance in implementations, despite the higher complexity bounds which were known until now. Indeed, we show that continued fractions lead to expected asymptotic bit complexity bounds that match those recently proven (in the worst case) for other exact, subdivision-based algorithms, such as Sturm, Descartes and Bernstein solvers. Using results from the metric theory of continued fractions we prove that the algorithm achieves an expected complexity of $\widetilde{\mathcal{O}}_B(d^6+d^4\tau^2)$, where d is the degree of the polynomial and τ bounds the coefficient bit size. Moreover, we present a variant of the algorithm with expected complexity $\widetilde{\mathcal{O}}_B(d^3\tau)$.

1.1 *Notation*

In what follows $\mathcal{O}$, resp. $\mathcal{O}_B$, means arithmetic, resp. bit, complexity and the $\widetilde{\mathcal{O}}$ and $\widetilde{\mathcal{O}}_B$ notation means that we are ignoring logarithmic factors. For a polynomial $A = \sum_{i=0}^{d} a_i X^i \in \mathbb{Z}[X]$, $d = \mathtt{deg}(A)$ 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 $\mathsf{a} \in \mathbb{Q}$, $\mathcal{L}(\mathsf{a}) > 1$ 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. $Var(A)$ denotes the number of sign variations in the coefficient list of A ignoring zero terms and Δ the separation bound of A, that is the smallest distance between two (complex) roots of A. Lastly, $\mathtt{PLB}(A)$ (Positive Lower Bound) is a function that computes a lower bound on the largest integer, i.e., the floor of the root, possibly complex, of A with the smallest positive real part, and $N = \max\{d, \tau\}$.

Real root isolation of univariate integer polynomials is a well known problem and various algorithms exist for it. Moreover, there is a huge bibliography on the problem so we have to mention that we only scratch the surface of the existing literature and we encourage the reader to refer to the references.

Most exact subdivision-based algorithms for real root isolation are based either on Descartes' rule of sign (Th. 1) or on Sturm sequences. Roughly speaking, the idea behind both approaches is to subdivide a given interval that initially contains all the real roots until it is certified that none or one real root is contained in the tested interval. Descartes' approach [18] achieves this by repeatedly transforming the original polynomial and counting the sign variations in the coefficients' list, while Sturm's approach computes a signed polynomial remainder sequence and evaluates it over the endpoints of the interval of interest. Recently it was proven (refer to [21, 22, 26] and references therein) that both approaches, the one based on Descartes' rule of sign (where the polynomials are represented either in the monomial or in the Bernstein basis) and the one based on Sturm sequences, achieve the same bit complexity bound in the worst case, namely $\widetilde{\mathcal{O}}_B(d^6 + d^4\tau^2)$ or $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$. Moreover, using Sturm(-Habicht) sequences in a pre-processing and a post-processing step [26] the bound holds for the non square-free case and the multiplicities of the roots can also be computed. If the degree of the polynomial is ≤ 4 then real solving can be performed in $\mathcal{O}(1)$ or $\widetilde{\mathcal{O}}_B(\tau)$ [23], instead of $\widetilde{\mathcal{O}}(\tau)$ or $\widetilde{\mathcal{O}}_B(\tau^2)$, which are achieved by the general purpose algorithms.

The continued fraction algorithm (CF) differs from the subdivision based algorithms in that instead of bisecting a given initial interval, it computes the continued fraction expansion for each real root of the polynomial. The first formulation of the algorithm is due to Vincent [57], see also [1, 7] for historical references. It was based on his theorem (Th. 4 without the terminating condition) where it was stated that repeated transformations of the polynomial will eventually yield a polynomial with zero (or one) sign variation, thus Descartes' rule (Th. 1 and Rem. 2) implies that the transformed polynomial has zero (resp. one) real root in $(0, \infty)$. If one sign variation is attained then the inverse transformation can be applied in order to compute an isolating interval for the real root that corresponds to the original polynomial. Moreover, the integers, c_i's, used in the transformations correspond to the partial quotients of the continued fraction expansion of the real root. However, Vincent's algorithm is exponential [18]. He computed the c_i's in the transformation of Th. 4 by repeated shift operations of the form $X \mapsto X + 1$, thus if one of the c_i's (or even the sum of all) is of magnitude, say, 2^τ then an exponential number of steps must be performed.

Uspensky [54] extended Vincent's theorem by computing an upper bound on the number of transformations so as to isolate the real roots, but failed to deal with its exponential behavior. See also [16, 48] where the problem of approximating a real algebraic number is also considered. Using Vincent's theorem, Collins and Akritas [18] derived a polynomial subdivision-based algorithm using Descartes' rule of sign.

Akritas [2, 6] in order to overcome the exponential behavior of the CF algorithm, computed the c_i's in the transformations as positive lower bounds of the positive real roots, via Cauchy's bound (for details, see sec. 3). He achieved a complexity of $\widetilde{\mathcal{O}}_B(d^5\tau^3)$ or $\widetilde{\mathcal{O}}_B(N^8)$, without using fast Taylor shifts [58]. However, it is not clear how this approach accounts for the increased coefficient size in the transformed polynomial after applying a map of the form $X \mapsto c + X$. Another issue is to bound the size of the c_i. Refer to Eq. (1) which indicates that the *magnitude* of the partial quotients is unbounded. CF is the standard real root isolation function in MATHEMATICA [4] and for some experiments against subdivision-based algorithms, also in MATHEMATICA, the reader may refer to [3]. Quite recently, Sharma proved [51] that the worst case complexity of the CF algorithm, using Hong's bound [28] for computing the partial quotients, is $\widetilde{\mathcal{O}}_B(d^7\tau^2)$ or $\widetilde{\mathcal{O}}_B(N^9)$.

Another class of univariate solvers are numerical solvers, e.g. [45, 46, 50], that compute an approximation of all the roots (real and complex) of a polynomial up to a desired accuracy. The complexity of these algorithms is $\widetilde{\mathcal{O}}_B(d^3\,\tau)$.

The contributions of this paper are the following: First, we improve the bound of the number of steps (transformations) that the CF algorithm performs, assuming that a constant number of shift operations is needed in order to compute the partial quotients. The proof of this is achieved through Th. 7. Second, we bound the expected bit size of the partial quotients and thus the growth of the transformed polynomials which appear during the algorithm. For this we use the hypothesis of the continued fraction expansion of real numbers (Hyp. 1) and a standard average case analysis. We revisit the proof of [2, 6] so as to overcome its drawbacks and derive an overall expected bit complexity bound of $\widetilde{\mathcal{O}}_B(N^6)$ for the algorithm, (see Sec. 4.2), thus matching the current record complexity in the worst case for exact real root isolation. From a theoretical perspective, we present a variant of the CF algorithm which has expected complexity $\widetilde{\mathcal{O}}_B(N^4)$, thus matching the complexity of the numerical algorithms. The extension to the non square-free case uses the techniques from [26]. Finally, we present our efficient open-source `C++` implementation of the $\widetilde{\mathcal{O}}_B(N^6)$ algorithm in SYNAPS [1], and illustrate it on various data sets, including polynomials of degree up to 1000 and coefficients of 8000 bits. Our software seems comparable to, and some times faster than the root isolation

[1] `http://www-sop.inria.fr/galaad/logiciels/synaps/`

implementations that we tested, including RS [2], which seems to be one of the fastest implementations for exact real root isolation. We also tested a numeric solver, namely ABERTH [10, 11], which is very efficient in practice but needs special tuning in order to produce the correct number of real roots. When the correct number of real roots is computed, then ABERTH can be up to six times faster than our implementation. We believe that our software contributes towards reducing the gap between rational and numeric computation, the latter being usually perceived as faster.

Part of this work appeared in preliminary form in [53].

The rest of the paper is structured as follows. The next section sketches the theory behind continued fractions and Sec. 3 presents the CF algorithm. In Sec. 4, we propose a way of computing the partial quotients (Sec. 4.1), we present the analysis of the CF algorithm (Sec. 4.2), and we propose a variant of the CF algorithm (Sec. 4.3). We conclude with experiments using our implementation, along with comparisons against other available software for univariate equation solving.

2 Continued fractions

We present a short introduction to continued fractions, following [55] which, although is far from complete, suffices for our purposes. The reader may refer to, e.g., [6, 12, 55, 60]. In general a *simple (regular) continued fraction* is a (possibly infinite) expression of the form

$$c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \dots}} = [c_0, c_1, c_2, \dots],$$

where the numbers c_i are called *partial quotients*, $c_i \in \mathbb{Z}$ and $c_i \geq 1$ for $i > 0$. Notice that c_0 may have any sign, however, in our real root isolation algorithm $c_0 \geq 0$, without loss of generality. By considering the recurrent relations

$$P_{-1} = 1,\ P_0 = c_0,\ P_{n+1} = c_{n+1}\, P_n + P_{n-1},$$
$$Q_{-1} = 0,\ Q_0 = 1,\ Q_{n+1} = c_{n+1}\, Q_n + Q_{n-1},$$

[2] `http://fgbrs.lip6.fr/salsa/Software/`

it can be shown by induction that $R_n = \frac{P_n}{Q_n} = [c_0, c_1, \ldots, c_n]$, for $n = 0, 1, 2, \ldots$ and moreover that

$$\begin{aligned} P_n\, Q_{n+1} - P_{n+1}\, Q_n &= (-1)^{n+1}, \\ P_n\, Q_{n+2} - P_{n+2}\, Q_n &= (-1)^{n+1} c_{n+2}. \end{aligned}$$

If $\gamma = [c_0, c_1, \ldots]$ then $\gamma = c_0 + \frac{1}{Q_0 Q_1} - \frac{1}{Q_1 Q_2} + \cdots = c_0 + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{Q_{n-1} Q_n}$ and since this is a series of decreasing alternating terms it converges to some real number γ. A finite section $R_n = \frac{P_n}{Q_n} = [c_0, c_1, \ldots, c_n]$ is called the $n-$th *convergent* (or *approximant*) of γ and the tails $\gamma_{n+1} = [c_{n+1}, c_{n+2}, \ldots]$ are known as its *complete quotients*. That is $\gamma = [c_0, c_1, \ldots, c_n, \gamma_{n+1}]$ for $n = 0, 1, 2, \ldots$. There is an one to one correspondence between the real numbers and the continued fractions, where evidently the finite continued fractions correspond to rational numbers.

It is known that $Q_n \geq F_{n+1}$ and that $F_{n+1} < \phi^n < F_{n+2}$, where F_n is the $n-$th Fibonacci number and $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. Continued fractions are the best (for a given denominator size), approximation. This is as follows:

$$\frac{1}{Q_n(Q_{n+1} + Q_n)} \leq \left| \gamma - \frac{P_n}{Q_n} \right| \leq \frac{1}{Q_n Q_{n+1}} < \phi^{-2n+1}.$$

Let $\gamma = [c_0, c_1, \ldots]$ be the continued fraction expansion of a real number. The Gauss-Kuzmin distribution [12, 39, 40, 47] states that for almost all real numbers γ (meaning that the set of exceptions has Lebesgue measure zero) the probability for a positive integer δ to appear as an element c_i in the continued fraction expansion of γ is

$$Prob[c_i = \delta] \backsimeq \lg \frac{(\delta+1)^2}{\delta(\delta+2)}, \quad \text{for any fixed } i > 0. \tag{1}$$

The Gauss-Kuzmin law induces that we can not bound the mean value of the partial quotients or in other words that the expected value (arithmetic mean) of the partial quotients is diverging, i.e.

$$E[c_i] = \sum_{\delta=1}^{\infty} \delta\, Prob[c_i = \delta] = \infty, \text{ for } i > 0.$$

Surprisingly enough the geometric (and the harmonic) mean is not only asymptotically bounded, but is bounded by a constant, for almost all $\gamma \in \mathbb{R}$. For the geometric mean this is the famous Khintchine's constant [33], see also [39], i.e.

$$\lim_{n\to\infty} \sqrt[n]{\prod_{i=1}^{n} c_i} = \mathcal{K} = 2.685452001...$$

which is not known if it is an irrational number, let alone transcendental. The reader may refer to [8] for a comprehensive treatment of *Khintchine's means*. The expected value of the bit size of the partial quotients is a constant for almost all real numbers, when $n \to \infty$ or n sufficiently big [33, 47]. Notice that in (1), $i > 0$, thus $\gamma \in \mathbb{R}$ is uniformly distributed in $(0, 1)$. Following closely [47], we have:

$$E[\ln c_i] = \frac{1}{n}\sum_{i=1}^{n} \ln c_i = \ln \mathcal{K} = 0.98785..., \quad \text{as } n \to \infty, \, \forall\, i > 0.$$

Let $\mathcal{L}(c_i) \triangleq b_i$, then

$$E[b_i] = \mathcal{O}(1). \tag{2}$$

A real number has an (eventually) periodic continued fraction expansion if and only if it is a root of an irreducible quadratic polynomial. The set of real algebraic numbers is countable and has Lebesgue measure zero, thus there is chance that Gauss-Kuzmin distribution and Khintchine's law do not hold for this set. However, "There is no reason to believe that the continued fraction expansions of non-quadratic algebraic irrationals generally do anything other than faithfully follow Khintchine's law"[13]. For our analysis we rely on the *conjecture* that Gauss-Kuzmin's distribution and Khintchine's law hold for the set of real algebraic numbers; various experimental results [12, 47, 48] support the conjecture. It is a major open problem to find an irreducible integer polynomial such that the continued fraction expansions of its real roots do not follow the conjecture or to prove the conjecture.

This is not the first time that the continuous distribution of the continued fraction expansion is used for an analysis of an algorithm. Brent [14, 15], see also [35], used the Gauss-Kuzmin distribution to model the expected bit size of the partial quotients of a rational number, in order to study the average complexity of the binary gcd algorithm. For the largest digit that can appear in the partial quotients of a rational number the reader may refer to [27].

Lévy loosened the assumptions of Khintchine and proved [39] that the distribution also holds for $\gamma \in \mathbb{R}$ with any density function in the set of Lebesgue measurable functions.

Hypothesis 1 *The set of real algebraic numbers of degree greater or equal to three follows Gauss-Kuzmin's distribution and Khintchine's law, i.e., the* *expected bit size of the partial quotients corresponding to the continued fraction expansion of these real algebraic numbers is a constant.*

3 The CF algorithm

Theorem 1 (Descartes' rule of sign) *The number R of real roots of $A(X)$ in $(0, \infty)$ is bounded by $Var(A)$ and we have $R \equiv Var(A) \mod 2$.*

Remark 2 *In general Descartes' rule of sign obtains an overestimation of the number of the positive real roots. However, if we know that A is* hyperbolic*, i.e. has only real roots, or when the number of sign variations is 0 or 1 then it counts exactly.*

The proof of Th. 1 follows from the following theorem which is due to Budan:

Theorem 3 (Budan) [6, 41] *Let a polynomial A, such that $\deg(A) = d$ and let $a < b$, where $a, b \in \mathbb{R}$. Let A_a, resp. A_b, be the polynomial produced after we apply the map $X \mapsto X + a$, resp. $X \mapsto X + b$, to A. Then the following hold:*

(1) $Var(A_a) \geq Var(A_b)$,
(2) $\#\{\gamma \in (a, b) \,|\, A(\gamma) = 0\} \leq Var(A_a) - Var(A_b)$, *and*
(3) $\#\{\gamma \in (a, b) \,|\, A(\gamma) = 0\} \equiv Var(A_a) - Var(A_b) \mod 2$.

The CF algorithm depends on the following theorem, which dates back to Vincent's theorem in 1836 [57]. The inverse of Th. 4 can be found in [6, 19, 41]. The version of the theorem that we present is due to Alesina and Galuzzi [7]; it improves the conditions of all the previous versions [1, 2, 6, 54] and involves in its proof the one and two circle theorems (refer to [7, 37] and references therein), employed in the analysis of the Descartes/Bernstein algorithm [18].

Theorem 4 [7] *Let $A \in \mathbb{Z}[X]$ be square-free and let $\Delta > 0$ be the separation bound, i.e. the smallest distance between two (complex) roots of A. Let n be the smallest index such that*

$$F_{n-1} \, F_n \, \Delta > \frac{2}{\sqrt{3}},$$

where F_n is the n-th Fibonnaci number. Then the map $X \mapsto [c_0, c_1, \ldots, c_n, X]$, where $c_0, c_1, \ldots, c_n$ is an arbitrary sequence of positive integers, transforms $A(X)$ to $A_n(X)$, whose list of coefficients has no more than one sign variation.

A similar theorem holds for non square-free polynomials but we will not use it for the analysis of the CF algorithm. The extension of Vincent's theorem to the non square-free case is due to Wang [59], see also [17] and for an improved version and historical references see [7].

Theorem 5 [7, 59] *Let $A \in \mathbb{Z}[X]$, not necessarily square-free, with* $\mathtt{deg}(A) = d$ *and let $\Delta > 0$ be the separation bound. Let k be the smallest index such that $F_{k-1}^2 \Delta > 1$, m be the smallest integer such that $m > \frac{1}{2}\log_\phi d$ and $n = k+m$. Then the map $X \mapsto [c_0, c_1, \ldots, c_n, X]$, where $c_0, c_1, \ldots, c_n$ is an arbitrary sequence of positive integers, transforms $A(X)$ to $A_n(X)$. If $Var(A_n) > 0$ then A_n has a unique positive real root of multiplicity $Var(A_n)$.*

The previous extension of Vincent's theorem implies, as already mentioned by Alesina and Galuzzi [7, Rem. 9], that Descartes' rule of sign can be used to isolate the real roots of non square-free polynomials and to compute their multiplicities, contrary to what it is believed up to now.

In our analysis we will assume that the input polynomial is square-free, except if explicitly stated otherwise, since we compute the multiplicities of the real roots differently. Thus we will rely on Th. 4 to isolate the positive real roots of a square-free polynomial A. In order to isolate the negative roots we perform the transformation $X \mapsto -X$, so in what follows we will consider only the positive real roots of A.

Vincent's variant of the CF algorithm goes as follows: A polynomial A is transformed to A_1 by the transformation $X \mapsto 1 + X$ and if $Var(A_1) = 0$ or $Var(A_1) = 1$ then A has 0, resp. 1, real root greater than 1 (Th. 1). If $Var(A_1) < Var(A)$ then (possibly) there are real roots of A in $(0, 1)$, due to Budan's theorem (Th. 3). A_2 is produced by applying the transformation $X \mapsto 1/(1 + X)$ to A. If $Var(A_2) = 0$ or $Var(A_2) = 1$ then A has 0, resp. 1, real root less than 1 (Th. 1). Uspensky's [54] variant of the algorithm (see also [48]) at every step produces both polynomials A_1 and A_2 probably, as Akritas states [1], because he was unaware of Budan's theorem. In both variants, if the transformed polynomial has more than one sign variations, we repeat the process.

We may consider the process of the algorithm as an infinite binary tree in which the root corresponds to the original polynomial A. The branch from a node to a right child corresponds to the map $X \mapsto X + 1$, while to the left child to the map $X \mapsto \frac{1}{1+X}$. Notice that a sequence of c transformations $X \mapsto 1 + X$ followed by one of the type $X \mapsto 1/(1 + X)$ is equivalent to two transformations, one of the type $X \mapsto c + 1/X$ followed by $X \mapsto 1 + X$. Thus Vincent's algorithm (and Uspensky's) results to a sequence of transformations like the one described in Th. 4, and so the leaves of the binary tree that we considered hold (transformed) polynomials that have no more than one sign variations, if Th. 4 holds. Akritas [2, 6] replaced a series of $X \mapsto X + 1$ transformations by $X \mapsto X + b$, where b is the positive lower bound (`PLB`) on the positive roots of the tested polynomial. This was computed by Cauchy's bound [6, 41, 60]. This way, the number of steps is polynomial and the complexity is in $\widetilde{\mathcal{O}}_B(d^5\tau^3)$. However, it is not clear whether or how the analysis takes into account that the coefficient bit size increases after a shift. Another issue is to bound the size of the b's.

For these polynomials that have one sign variation we still have to find the interval where the real root of the initial polynomial A lies. Consider a polynomial A_n that corresponds to a leaf of the binary tree that has one sign variation. Notice that A_n is produced after a transformation as in Th. 4, using positive integers $c_0, c_1, \ldots, c_n$. This transformation can be written in a more compact form using the convergents

$$M : X \mapsto \frac{P_n X + P_{n-1}}{Q_n X + Q_{n-1}}, \tag{3}$$

where $\frac{P_{n-1}}{Q_{n-1}}$ and $\frac{P_n}{Q_n}$ are consecutive convergents of the continued fraction $[c_0, c_1, \ldots, c_n]$. Notice that (3) is a Möbius transformation, see [6, 60] for more details. Since A_n has one sign variation it has one and only one real root in $(0, \infty)$, so in order to obtain the isolating interval for the corresponding real root of A we evaluate the right part of Eq. (3) once over 0 and once over ∞. The (unordered) endpoints of the isolating interval are $\frac{P_{n-1}}{Q_{n-1}}$ and $\frac{P_n}{Q_n}$.

The pseudo-code of the CF algorithm is presented in Alg. 1. Notice that the `Interval` function orders the endpoints of the computed isolating interval and that $\texttt{PLB}(A)$ computes a lower bound on the positive roots of A. The initial input of the algorithm is a polynomial $A(X)$ and the trivial transformation $M(X) = X$. We need the functional M in order to keep track of the transformations that we perform so that to derive the isolating intervals. Notice that Line 15 is to be executed only when $Var(A_1) < Var(A_2)$, but in order to simplify the analysis we omit this, since it only doubles the complexity.

Remark 6 *The CF algorithm takes into account* all *the complex roots of the polynomial; in other words depends on all the roots with positive real part.*

4 The complexity of the CF algorithm

The complexity of the CF algorithm depends on the number of transformations and the cost of each. However, special care should be taken since after each transformation the bit size of the coefficients of the polynomial increases.

Let $\texttt{disc}(A)$ be the discriminant and $\texttt{lead}(A)$ the leading coefficient of A. Mahler's measure of a polynomial A is $\mathcal{M}(A) = |\,\texttt{lead}(A)\,| \prod_{i=1}^{d} \max\{1, |\gamma_i|\}$, where γ_i are all the (complex) roots of A [9, 41, 43, 60]. Moreover $\mathcal{M}(A) \leq$

Algorithm 1: $\texttt{CF}(A, M)$

Input: $A \in \mathbb{Z}[X], M(X) = \frac{kX+l}{mX+n}$, $k, l, m, n \in \mathbb{Z}$
Output: A list of isolating intervals
Data: Initially $M(X) = X$, i.e. $k = n = 1$ and $l = m = 0$

if $A(0) = 0$ **then**
  $\texttt{OUTPUT Interval(}\ M(0), M(0))$;
  $A \leftarrow A(X)/X$;
  $\texttt{CF}(A, M)$;
$V \leftarrow \texttt{Var}(A)$;
if $V = 0$ **then** RETURN ;
if $V = 1$ **then**
  $\texttt{OUTPUT Interval(}\ M(0), M(\infty))$;
  RETURN ;
$b \leftarrow \texttt{PLB}(A)$ `// PLB` $\equiv$ `PositiveLowerBound` ;
if $b \geq 1$ **then** $A \leftarrow A(b + X), M \leftarrow M(b + X)$;
$A_1 \leftarrow A(1 + X)$, $M_1 \leftarrow M(1 + X)$;
$\texttt{CF}(A_1, M_1)$ `// Looking for real roots in` $(1, +\infty)$;
$A_2 \leftarrow A(\frac{1}{1+X})$, $M_2 \leftarrow M(\frac{1}{1+X})$;
$\texttt{CF}(A_2, M_2)$ `// Looking for real roots in` $(0, 1)$;
RETURN ;

$2^{\tau}\sqrt{d+1}$. We prove the following theorem, which is based on a theorem by Mignotte [41], thus extending [20, 22].

Theorem 7 *Let $A \in \mathbb{Z}[X]$, with $\deg(A) = d$ and $\mathcal{L}(A) = \tau$. 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*

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

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

$$\forall\, a, b \in \mathbb{C} \quad |a - b| \leq 2 \max\{|a|, |b|\}, \tag{4}$$

and the fact [41, 43] 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 [41, 60] that $\texttt{disc}(A) = \texttt{lead}\,(A)^{2d-2} \prod_{i<j} (\gamma_i - \gamma_j)^2$. For the right

inequality we consider the absolute value of the discriminant of A, i.e

$$\begin{aligned} |\,\mathtt{disc}(A)| &= |\,\mathtt{lead}\,(A)\,|^{2d-2} \prod_{i<j} |\gamma_i - \gamma_j|^2 \\ &= |\,\mathtt{lead}\,(A)\,|^{2d-2} \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j|^2 \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j|^2 \Leftrightarrow \\ \sqrt{|\,\mathtt{disc}(A)|} &= |\,\mathtt{lead}\,(A)\,|^{d-1} \prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j|. \end{aligned}$$

We consider the product $\prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j|$ and we apply $\frac{d(d-1)}{2} - k$ times inequality (4), thus

$$\begin{aligned} \prod_{(i,j)\notin\Omega} |\gamma_i - \gamma_j| &\leq 2^{\frac{d(d-1)}{2}-k} |\gamma_1|^0 |\gamma_2|^1 \cdots |\gamma_d|^{d-1} \left(\prod_{j\in\overline{\Omega}} |\gamma_j|\right)^{-1} \\ &\leq 2^{\frac{d(d-1)}{2}-k} \mathcal{M}(A)^{d-1} |\,\mathtt{lead}\,(A)\,|^{1-d} \mathcal{M}(A)^k. \end{aligned} \tag{5}$$

where we used the inequality $|\gamma_1|^0|\gamma_2|^1 \cdots |\gamma_d|^{d-1} \leq |\mathcal{M}(A)/\,\mathtt{lead}\,(A)\,|^{d-1}$, and the fact [41] that, since $\forall i$, $|\gamma_i| \geq \mathcal{M}(A)^{-1}$, we have $\prod_{j\in\overline{\Omega}} |\gamma_j| \geq |\gamma_1|^k \geq \mathcal{M}(A)^{-k}$. Thus we conclude that

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

□

A similar theorem but with more strict hypotheses on the roots first appeared in [20], see also [32], and the conditions were generalized in [22]; namely in order for the bound [20, 22] to hold the sets of indices i and j should be rearranged such that they form an acyclic graph where each node has out-degree at most one. The bound of Th. 7 has a factor 2^{d^2} instead of d^d in [20, 22, 32], which plays no role when the polynomial is not square-free or when $d = \mathcal{O}(\tau)$ or when the notation with N is used. Moreover, we loosen the hypotheses of the theorem and thus all the proofs concerning the number of steps of the subdivision-based solvers [22, 26] are simplified, since there is no need to rearrange the roots and apply the one and two circle theorems. Possibly a more involved proof of Th. 7 may eliminate this factor using [42].

Remark 8 *There are two crucial observations*[3] *about Th. 4. When the transformed polynomial has one sign variation, then the interval with endpoints* $\frac{P_{n-1}}{Q_{n-1}} = [c_0, c_1, \ldots, c_{n-1}]$ *and* $\frac{P_n}{Q_n} = [c_0, c_1, \ldots, c_n]$ *(possibly unordered) isolates a positive real root of A, say γ_i. Then, in order for Th. 4 to hold, it suffices to consider, instead of Δ, the quantity $|\gamma_i - \gamma_{c_i}|$, where γ_{c_i} is a (complex) root of A closest to γ_i. When the transformed polynomial has no sign variation and $[c_0, c_1, \ldots, c_n]$ is the continued fraction expansion of the (positive) real part of a complex root of A, say γ_i, then again it suffices to replace Δ by $|\gamma_i - \gamma_{c_i}|$.*

For the following theorem we assume that a small number of calls to `PLB` is

[3] For a proof of these observations, the reader may also refer to [51].

needed in order to compute the floor of the root with smallest positive real part. We will justify this in the next section for the expected case.

Theorem 9 *The CF algorithm performs at most $\mathcal{O}(d^2 + d\tau)$ transformation steps, assuming that at each step a constant number of calls to* `PLB` *is needed in order to compute the floor of the root with the smallest positive real part.*

Proof. Let $0 < |\gamma_1| \leq |\gamma_2| \leq \cdots \leq |\gamma_k|$, $k \leq d$ be the (complex) roots of A with positive real part and let γ_{c_i} denote the root of A that is closest to γ_i. We assume that at each step a constant number of calls to `PLB` is needed in order to compute the partial quotient of the root that we are trying to isolate. Or in other words that Lines 10 and 11 of Alg. 1 are executed a constant number of times, at each step, in order to compute the partial quotient.

We consider the binary tree T generated during the execution of the CF algorithm. The number of steps of the CF algorithm corresponds to the number of nodes in T, which we denote by $\#(T)$. We use some arguments and the notation from [22] in order to prune the tree.

With each node v of T we associate a Möbius transformation $M_v : X \mapsto \frac{kX+l}{mX+n}$, a polynomial A_v and implicitly an interval I_v whose unordered endpoints can be found if we evaluate M_v on 0 and on ∞. Recall that A_v is produced after M_v is applied to A. The root of T is associated with A, $M(X) = X$ (i.e. $k = n = 1, l = m = 0$) and implicitly with the interval $(0, \infty)$.

Let a leaf u of T be of **type-i** if its interval I_u contains $i \geq 0$ real roots. Since the algorithm terminates the leaves are of type-0 or type-1. We will prune certain leaves of T so as to obtain a certain sub-tree T' where it is easy to count the number of nodes. We remove every leaf that has a sibling that is not a leaf. Now we consider the leaves that have a sibling that is also a leaf. If both leaves are of type-1, we arbitrary prune one of them. If one of them is of type-1 then we prune the other. If both leaves are of type-0, this means that the polynomial on the parent node has at least two sign variations and thus that we are trying to isolate the (positive) real part of some complex root. We keep the leaf that contains the (positive) real part of this root. And so $\#(T) < 2\,\#(T')$.

Now we consider the leaves of T'. All are of type-0 or type-1. In both cases they hold the positive real part of a root of A, the associated interval is $|I_v| \geq |\gamma_i - \gamma_{c_i}|$ (Rem. 8) and the number of nodes from a leaf to the root is n_i, which is such that the hypothesis of Th. 4 is satisfied. Since n_i is the smallest index such that the hypothesis of Th. 4 holds, if we reduce n_i by one then the inequality does not hold. Thus

$$F_{n_i-2}\, F_{n_i-1}\, |\gamma_i - \gamma_{c_i}| \leq \frac{2}{\sqrt{3}} \Rightarrow \phi^{2n_i-5}\, |\gamma_i - \gamma_{c_i}| < \frac{2}{\sqrt{3}} \Rightarrow n_i < 2 - \frac{1}{2} \lg |\gamma_i - \gamma_{c_i}|.$$

We sum over all n_i to bound the nodes of T', thus

$$\#(T') \leq \sum_{i=1}^{k} n_i \leq 2k - \frac{1}{2}\sum_{i=1}^{k} \lg|\gamma_i - \gamma_{c_i}| \leq 2k - \frac{1}{2}\lg\prod_{i=1}^{k}|\gamma_i - \gamma_{c_i}|. \qquad (6)$$

In order to apply Th. 7 we should rearrange $\prod_{i=1}^{k}|\gamma_i - \gamma_{c_i}|$ so that the requirements on the indices of roots are fulfilled. This can not be achieved when symmetric products occur and thus the worst case is when the product consists only of symmetric products i.e. $\prod_{i=1}^{k/2}|(\gamma_j - \gamma_{c_j})(\gamma_{c_j} - \gamma_j)|$. We consider the square of the inequality of Th. 7 taking $\frac{k}{2}$ instead of k and $\texttt{disc}(A) \geq 1$ (since A is square-free), thus

$$\begin{aligned} \textstyle\prod_{i=1}^{k}|\gamma_i - \gamma_{c_i}| &\geq \left(2^{\frac{k}{2} - \frac{d(d-1)}{2}}\,\mathcal{M}(A)^{1-d-\frac{k}{2}}\right)^2 \\ -\lg\textstyle\prod_{i=1}^{k}|\gamma_i - \gamma_{c_i}| &\leq d^2 - d - k + (2d+k-2)\lg\mathcal{M}(A). \end{aligned} \qquad (7)$$

Eq. (6) becomes $\#(T') < 2k + d^2 - d - k + (2d+k-2)\lg\mathcal{M}(A)$. However, for Mahler's measure it is known that $\mathcal{M}(A) \leq 2^{\tau}\sqrt{d+1} \Rightarrow \lg\mathcal{M}(A) \leq \tau + \lg d$, for $d \geq 2$, thus $\#(T') \leq 2k + d^2 - d - k + (2d+k-2)(\tau + \lg d)$. Since $\#(T) < 2\,\#(T')$ and $k \leq d$, we conclude that $\#(T) = \mathcal{O}(d^2 + d\,\tau + d\lg d)$. $\square$

4.1 Rational roots and `PLB` (Positive Lower Bound) realization

This section studies a way to compute a lower bound on the positive roots, and presents its efficiency and accuracy. This also supports the assumption that the expected number of calls to `PLB` is $\mathcal{O}(1)$. Notice that different variants of the CF algorithm can be introduced depending on how the positive lower bound is computed. The accuracy and the efficiency of the bounds in the context of the CF algorithm are seldom, if at all, discussed. For a recent result concerning the worst case complexity, the reader may refer to [51].

There are two issues that we have to discuss further.

The first one concerns the rational numbers. If the polynomial A has (only) rational real roots then their continued fraction expansion neither follows the Gauss-Kuzmin distribution nor Khintchine's law. However, recall that if $\frac{p}{q}$ is a root of A then p divides a_0 and q divides a_d, thus in the worst case $\mathcal{L}(p/q) = \mathcal{O}(\tau)$ and so the rational roots are isolated fast among themselves. Treating them as real algebraic numbers leads to an overestimation of the number of iterations.

The second issue concerns the number of calls of function PLB in order to compute a partial quotient. We made the assumption that this number of calls is small. In practice this is always the case. The assumption that the number of calls to PLB is small enough, is strengthened by (1), since it implies that the probability that a partial quotient is of magnitude ≤ 10 is ~ 0.87. This is why in practice the partial quotients are of very small magnitude, except when the polynomial has only rational real roots, of great magnitude, well separated and we are interested in the practical complexity. In this case PLB must be applied many times in order to compute a partial quotient. Richtmyer et al. [47] in order to overcome this situation perform a small number of Newton-like iterations in order to get a good approximation of the partial quotient. In [4], see also [2, 3], the problem was solved partially by applying the map $X \mapsto bX$, where b is the computed positive root bound, when $b \geq 16$. This is what we do in our implementation.

Moreover, the relation

$$\left|\gamma - \frac{P_n}{Q_n}\right| < \frac{1}{c_{n+1}\, Q_n^2},$$

implies that the appearance of a partial quotient of an extra-ordinary big magnitude means that the *previous* approximation of the algebraic number was extremely good. However, the previous discussion does not provide a theoretical explanation. We will present a way of computing the positive lower bound that supports our assumption.

Recall that a lower bound on the positive roots of a polynomial is computed as the inverse of the upper bound on the positive roots of the reciprocal polynomial. Thus in what follows we will consider only upper bounds for the positive roots. The bound that we will consider, and that we also use in our implementation of PLB, is

$$B_2(A) := 2 \max_{a_k < 0} \left\{ \left| \frac{a_k}{a_d} \right|^{\frac{1}{d-k}} \right\}, \tag{8}$$

where $0 \leq k < d$, which is due to [34], see also [32, 35]. Notice that B_2 is a bound for the positive roots only and not a bound for the absolute value of all the (complex) roots of the polynomial. For such bounds, the reader may refer to e.g. [41, 43, 56]. For other bounds on the positive roots the reader may refer to [5, 28, 34, 52]. Last, but not least, we have to mention that the implementation of B_2 requires $\widetilde{\mathcal{O}}(d)$ arithmetic operations [6, 36, 56] and as van der Sluis [56] says, this bound "is to be recommended among all" because of its simplicity and the good quality of its results.

Under Hyp. 1, the expected bit size of the partial quotients is a constant. We compute them, using a combination of binary and exponential search [4] [38].

[4] This approach was also proposed to the first author by K. Mehlhorn, A. Eigenwillig and M. Sagraloff during his visit to MPI-Saarbrücken.

First, we compute $B_2(A)$ and perform the transformation $X \mapsto X + B_2(A)$. Next, we perform the transformation $X \mapsto X + 1$. If the number of sign variations decreases then $B_2(A)$ is the partial quotient. If not, then we perform $X \mapsto X+2$. If the number of sign variations does not decrease, then we perform $X \mapsto X+2^2$. Again if the number of sign variations does not decrease, then we perform $X \mapsto X + 2^3$ and so on. Eventually, for some positive integer k, there would be a loss in the sign variations between transformations $X \mapsto X+2^k$ and $X \mapsto X + 2^{k+1}$, i.e. the partial quotient c_i that we want to compute satisfies $B_2(A) + 2^k < c_i < B_2(A) + 2^{k+1} < 2\,c_i$. We compute the c_i by performing binary search in the interval $[B_2(A) + 2^k, B_2(A) + 2^{k+1}]$. Thus, the number of transformations that we need to perform is $\mathcal{O}(\lg c_i) = \mathcal{O}(b_i)$, which is $\mathcal{O}(1)$, by (2). We do not consider the cases $c_i = 2^k$ or $c_i = 2^{k+1}$, since then we have computed a rational root.

The previous discussion implies that at every step of the algorithm (in the expected case) we must perform one call to `PLB` and a small number of shift operations, in order to compute the floor of a root with the smallest positive real part. Notice that this root may not be unique.

In practice the tightness of the positive root bounds is usually very good, thus we do not use this exponential search in order to compute the partial quotients.

We believe that this technique of computing the positive lower bound might be used in order to improve the worst case complexity bound of CF [51].

4.2 Real root isolation

To complete the analysis of the CF algorithm we have to compute the cost of every step that the algorithm performs. In the worst case every step consists of a computation of a positive lower bound b (Line 10) and three transformations, $X \mapsto b+X$, $X \mapsto 1+X$ and $X \mapsto \frac{1}{1+X}$ (Lines 11, 12 and 14 in Alg. 1). Recall, that inversion can be performed in $\mathcal{O}(d)$. Thus, the complexity is dominated by the shift operation (Line 11 in Alg. 1) since in the expected case a constant number of calls to `PLB` is needed, as justified in Sec. 4.1. In order to compute this cost, a bound on $\mathcal{L}(c_k) \triangleq b_k, 0 \leq k \leq m_i$ is needed, see Eq. (2).

For the analysis of the CF algorithm we will need the following:

Proposition 10 (Fast Taylor shift) [58] *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 $\mathcal{O}_B(\mathsf{M}(d^2 \lg d + d^2\sigma + d\tau))$. Moreover $\mathcal{L}(B) = \mathcal{O}(\tau + d\sigma)$.*

Initially A has degree d and bit size τ. Evidently the degree does not change after a shift operation. Each shift operation by a number of bit size b_h increases the bit size of the polynomial by an additive factor $d\, b_h$, in the worst case (Prop. 10). At the $h-$th step of the algorithm the polynomial has bit size $\mathcal{O}(\tau + d\sum_{i=1}^{h} b_i)$ and we perform a shift operation by a number of bit size b_{h+1}. Prop. 10 states that this can be done in $\mathcal{O}_B\left(\mathsf{M}\left(d^2 \lg d + d^2 b_{h+1} + d(\tau + d\sum_{i=1}^{h} b_i)\right)\right)$ or $\mathcal{O}_B\left(\mathsf{M}\left(d^2 \lg d + d\tau + d^2 \sum_{i=1}^{h+1} b_i\right)\right)$.

Now we have to bound $\sum_{i=1}^{h+1} b_i$. For this we use Hyp. 1 and we derive a bound on the expected complexity. Below we elaborate on this. We use Eq. (2) which bounds $E[b_i]$. By linearity of expectation it follows that $E[\sum_{i=1}^{h+1} b_i] = \mathcal{O}(h)$. Since $h \leq \#(T) = \mathcal{O}(d^2 + d\tau)$ (Th. 9 and Sec. 4.1), the (expected) worst case cost of step h is $\mathcal{O}_B(\mathsf{M}\,(d^2 \lg d + d\tau + d^2(d^2 + d\tau)))$ or $\widetilde{\mathcal{O}}_B(d^2(d^2 + d\tau))$. Finally, multiplying by the number of steps, $\#(T)$, we conclude that the overall complexity is $\widetilde{\mathcal{O}}_B(d^6 + d^5\tau + d^4\tau^2)$, or $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ if $d = \mathcal{O}(\tau)$, or $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$.

Now let us isolate, and compute the multiplicities, of the real roots of $A_{in} \in \mathbb{Z}[X]$, which is not necessarily square-free, with $\mathtt{deg}(A_{in}) = d$ and $\mathcal{L}\,(A_{in}) = \tau$. We use the technique from [26] and compute the square-free part A of A_{in} using Sturm-Habicht sequences in $\widetilde{\mathcal{O}}_B(d^2\tau)$. The bit size of A is $\mathcal{L}\,(A) = \mathcal{O}(d + \tau)$. Using the CF algorithm we isolate the positive real roots of A and then, by applying the map $X \mapsto -X$, we isolate the negative real roots. Finally, using the square-free factorization of A_{in}, which can be computed in $\widetilde{\mathcal{O}}_B(d^2\tau)$, it is possible to find the multiplicities in $\widetilde{\mathcal{O}}_B(d^3\tau)$.

The previous discussion leads to the following theorem.

Theorem 11 *Let $A \in \mathbb{Z}[X]$ (not necessarily square-free) such that* $\mathtt{deg}(A) =$ $d > 2$ *and* $\mathcal{L}\,(A) = \tau$. *We can isolate the real roots of A, using CF, and compute their multiplicities in expected time $\widetilde{\mathcal{O}}_B(d^6 + d^4\tau^2)$, or $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$, if Hypothesis 1 holds.*

In order to work in the expected case, we assume that for random polynomials, by considering a distribution on their coefficients, the real algebraic numbers which are roots of these polynomials follow Hyp. 1. One way to formalize this is as follows: The (complex) roots of the polynomials cluster uniformly around (and very close to) the unit circle [29, 30, 31]. The density of the real parts of the roots is a Lebesgue measurable function, if we consider the set of the real parts as a set of real numbers, and thus Hyp. 1 holds [39].

For non square-free polynomials, instead of using Sturm-Habicht sequences in order to compute to the square-free part of the polynomial and compute the multiplicities, we may rely on Th. 5 and force the algorithm to compute isolating intervals as small as the (theoretical) separation bound.

A closer look to the proof of Th. 9 reveals that in order to derive the number of steps of the CF algorithm we do not depend on an interval that initially contains all the real roots. Notice that this dependence is intrinsic for the subdivision algorithms [22, 26]. This will allow us to improve the complexity of the CF algorithm by spreading away the roots.

We consider the square-free polynomial A and we apply the homothetic transformation $X \mapsto X/2^{\ell(d+\tau)}$, where ℓ is a constant specified in the sequel. Notice that in any case ℓ should be chosen such that to be an integer value. Besides that, the algorithm is exactly the same as in Alg. 1. The transformed polynomial, say C, has bit size $\mathcal{O}(\tau + \ell d^2 + \ell d\tau)$ and its roots β_j are the roots of A multiplied by $2^{\ell(d+\tau)}$, i.e.

$$\beta_i = 2^{\ell(d+\tau)}\,\gamma_i, \tag{9}$$

where γ_i are the roots of A and $1 \leq i \leq d$. Evidently it suffices to isolate the real roots of C.

We will use the notation of the proof of Th. 9. Let k_1 be the number roots of C with positive real part and k_2 those with negative real part. Notice that $k_1 + k_2 = d$. Following the proof of Th. 9, see Eq. (6), the number of steps that the CF algorithm must perform, besides the small number of shift operations needed to compute partial quotients (see Sec. 4.1), in order to isolate the real roots of C is

$$\begin{aligned}
\#(T') &\leq \sum_{i=1}^{k_1} n_i + \sum_{j=1}^{k_2} n_j \\
&\leq 2k_1 - \frac{1}{2}\sum_{i=1}^{k_1} \lg|\beta_i - \beta_{c_i}| + 2k_2 - \frac{1}{2}\sum_{j=1}^{k_2} \lg|\beta_j - \beta_{c_j}| \\
&\leq 2d - \frac{1}{2}\lg\prod_{i=1}^{d}|\beta_i - \beta_{c_i}|.
\end{aligned} \tag{10}$$

If we consider the product term of the previous equation and (9) then

$$\prod_{i=1}^{d}|\beta_i - \beta_{c_i}| = 2^{\ell d^2 + \ell d\tau}\prod_{i=1}^{d}|\gamma_i - \gamma_{c_i}|.$$

Combining the previous equation with (7) we have

$$
\begin{aligned}
-\lg \prod_{i=1}^{d} |\beta_i - \beta_{c_i}| &\leq d^2 + (3d-2)\tau - 2d - 2\lg d + 3d\lg d - \ell\,(d^2 + d\tau) \\
&\leq 4d^2 + 4d\tau + 3d\lg d - \ell\,(d^2 + d\tau)\,.
\end{aligned}
\tag{11}
$$

We want to specify the value of ℓ in such way so as to eliminate the quantities of the form d^2 and $d\tau$ from Eq. (11). By elementary calculus we see that $\ell = 4$.

Using this result and combining Eq. (10) and (11), we conclude that $\#(T) = \mathcal{O}(d\,\lg d) = \widetilde{\mathcal{O}}(d)$. If we substitute this value of $\#(T)$ in the proof of Th. 11, presented in Sec. 4.2, recalling that a small number of shift operations is needed on order to compute the partial quotients (Sec. 4.1) and taking into account that the bit size of C is $\mathcal{O}(\tau + \ell d^2 + \ell d\tau)$, then we conclude the following.

Corollary 12 *The expected complexity of this variant of the CF algorithm is* $\widetilde{\mathcal{O}}_B(d^3\tau)$*, if Hypothesis 1 holds.*

This variant of the CF algorithm is of small practical interest, as our preliminary experiments also indicate, since the transformation $X \mapsto X/2^{\ell(d+\tau)}$ increases the bit size a lot. However, to the best of our knowledge this is the first complexity bound, even using average case analysis, that matches the complexity bounds of the numerical algorithms [45, 46, 50].

5 Implementation and experiments

We have implemented the CF algorithm in SYNAPS [44], which is a `C++` library for symbolic-numeric computations that provides data-structures, classes and operations for univariate and multivariate polynomials, vector and matrices. Our code is already included [5] in the current version of SYNAPS. Our implementation uses (8) for computing the positive lower bound. There are also implementations by B. Mourrain and V. Sharma that use other positive lower bounds, e.g. [5, 28]. The implementation is based on the integer arithmetic of GMP [6] (v. 4.1.4) and uses only transformations of the form $X \mapsto 2^{\beta}X$ and $X \mapsto X+1$ to benefit from the fast implementations that are available in GMP. However, our implementation follows the generic programming paradigm, thus any library that provides arbitrary precision integer arithmetic can be used instead of GMP.

We restrict ourselves to square-free polynomials of degree $\in \{100, 200, \ldots, 1000\}$. Following [49], the first class of experiments concerns well-known ill-conditioned

[5] The reader may refer to the file `synaps/usolve/bin/solve_cf.cc`

[6] `www.swox.com/gmp/`

polynomials namely: Laguerre (L), first (C1) and second (C2) kind Chebyshev, and Wilkinson (W) polynomials. We also consider Mignotte (M1) polynomials $X^d - 2(101X - 1)^2$, that have 4 real roots but two of them very close together, and a product $\left(X^d - 2(101X-1)^2\right)\left(X^d - 2((101+\frac{1}{101})X-1)^2\right)$ of two such polynomials (M2) that has 8 real roots. Finally, we consider polynomials with random coefficients (R1), and monic polynomials with random coefficients (R2) in the range [-1000, 1000], produced by MAPLE, using 101 as a seed for the pseudo-random number generator.

We performed experiments against RS which seems to be one of the fastest available software for exact real root isolation. It implements a subdivision-based algorithm using Descartes' rule of sign with several optimizations and symbolic-numeric techniques [49]. Note that we had to use RS through its MAPLE interface. Timings were reported by its internal function `rs_time()`.

We also test ABERTH [10, 11], which a numerical solver with unknown (bit) complexity but very efficient in practice, available through SYNAPS. In particular, it uses multi-precision floats and provides a floating-point approximation of all the complex roots. Since ABERTH is a numerical solver it approximates the roots up to a desired accuracy. Even though we tuned ABERTH to search for roots on the real axis only, unfortunately, we were not always able to tune its behavior in order to produce the correct number of real roots in all the cases, i.e. to specify the output precision.

In SYNAPS, there are several univariate solvers, based on Sturm sequences, Descartes' rule of sign, Bernstein basis, etc (see [26] for details and experimental results). CF is clearly faster than all these solvers, therefore we do not report on these experiments. In particular, the large inputs used here are not tractable by the Sturm-sequence solver in SYNAPS, and this is also the case for another implementation of the Sturm-sequence solver in CORE[7].

So, in Table 1, we report experiments with CF, RS, ABERTH, where the timings are in seconds. The asterisk (*) denotes that the computation did not finish after 12000s. The experiments were performed on a 2.6 GHz Pentium with 1 GB RAM, and our code was compiled using g++ 3.3 with option -O3.

For (M1) and (M2), there are rational numbers with a very simple continued fraction expansion that isolate the real roots which are close. These experiments are extremely hard for RS. On (M1), ABERTH is the fastest and correctly computes all real roots, but on (M2), which has 4 real roots close together, it is slower than CF. CF is advantageous on (W) since, as soon as a real root is found, transformations of the form $X \mapsto X + 1$ rapidly produce the other real roots. We were not able to tune ABERTH on (W). For (L), (C1) and (C2), CF

[7] `cs.nyu.edu/exact/core_pages/`

		100	200	300	400	500	600	700	800	900	1000
L	CF	0.27	2.24	9.14	25.27	55.86	110.13	214.99	407.09	774.22	1376.34
	RS	0.65	3.65	13.06	35.23	77.21	151.17	283.43	527.42	885.86	1387.45
	#roots	100	200	300	400	500	600	700	800	900	1000
C1	CF	0.11	0.85	3.16	8.61	19.67	38.23	77.75	139.18	247.11	414.51
	RS	0.21	1.36	3.80	10.02	23.15	46.02	82.01	150.01	269.35	458.67
	#roots	100	200	300	400	500	600	700	800	900	1000
C2	CF	0.11	0.77	3.14	8.20	19.28	38.58	73.59	133.52	233.48	386.61
	RS	0.23	1.48	3.80	9.84	23.28	46.34	83.58	146.04	273.00	452.77
	#roots	100	200	300	400	500	600	700	800	900	1000
W	CF	0.11	0.76	2.54	6.09	12.07	21.43	34.52	53.35	81.88	120.21
	RS	0.09	0.59	2.25	6.34	14.62	29.82	55.47	104.56	179.23	298.45
	#roots	100	200	300	400	500	600	700	800	900	1000
M1	CF	0.02	0.08	0.21	0.42	0.73	1.19	1.84	2.75	4.16	6.22
	RS	7.83	287.27	1936.48	7328.86	*	*	*	*	*	*
	ABERTH	0.01	0.04	0.07	0.11	0.12	0.26	0.43	0.37	0.47	0.90
	#roots	4	4	4	4	4	4	4	4	4	4
M2	CF	0.08	0.43	1.10	2.78	4.71	8.67	18.26	25.28	40.15	60.10
	RS	1.24	144.64	1036.785	4278.275	12743.79	*	*	*	*	*
	#roots	8	8	8	8	8	8	8	8	8	8
R1	CF	0.001	0.04	0.07	0.33	0.06	0.37	0.66	0.76	1.03	1.77
	RS	0.026	0.09	0.11	0.68	0.22	0.89	0.95	0.69	1.55	2.09
	ABERTH	0.02	0.03	0.07	0.14	0.21	0.31	0.44	0.51	0.64	0.80
	#roots	4	4	2	6	2	4	4	2	4	4
R2	CF	0.01	0.04	0.08	0.36	0.14	0.38	0.74	0.77	1.24	1.42
	RS	0.05	0.23	0.47	1.18	0.81	1.64	2.68	3.02	4.02	4.88
	ABERTH	0.01	0.05	0.08	0.14	0.23	0.33	0.44	0.55	0.67	0.83
	#roots	4	4	4	6	4	4	6	4	6	4

Table 1
Experimental results

is clearly faster than RS, while we were not able to appropriately tune ABERTH to produce the correct number of real roots. The polynomials in (R1) and (R2) have few and well separated real roots, thus the semi-numerical techniques in RS are very effective. To be more specific, RS isolates all roots using only 63 bits of accuracy (this information was extracted using the function `rs_verbose(1)`). However, even in this case, CF is comparable to RS. ABERTH is even faster on these experiments (see Table 1). We have to mention that, as F. Rouillier pointed out to us, RS can about 30% faster in (L), (C1) and (C2) if we use it with the (non-default) option `precision=0`.

We also tested a univariate polynomial that appears in the Voronoi diagram of ellipses [25]. The polynomial has degree 184, coefficient bit size 903, and 8 real roots. CF solves it in 0.12s, RS in 0.3s and ABERTH in 1.7s. Finally, for polynomials of the form $(X^d - 2(a\,X-1)^2)(X^d - (a\,X-1)^2)$ [22], which have three real roots very close to $\frac{1}{a}$, the behavior of the CF solver is similar to that of the subdivision-based algorithms.

In short, CF is complete, simple to use and is at least as efficient as the state of the art.

6 Future work

The first thing that comes in mind is to drop the dependence of the analysis on Hyp. 1. This conjecture is one the most important open problems in the theory of continued fractions.

Th. 5 implies that Descartes' rule of sign can be used for non square-free polynomials. Of course the obstacle is that we have to perform iterations up to the theoretical separation bound, which is a very bad overestimation of the actual one.

As for the implementation of the CF algorithm, there are several ways that should improve our solver. First, instead of exact integer arithmetic we may use semi-numerical techniques like those in RS [49]. These techniques may be based on interval arithmetic.

Last, but not least, the expected complexity bounds for the CF algorithm that we present in this paper motivate questions about similar bounds for the subdivision-based algorithms. We conjecture that the expected complexity of the subdivision-based solvers, i.e. Descartes/Bernstein and Sturm is $\widetilde{\mathcal{O}}_B(N^5)$ if not $\widetilde{\mathcal{O}}_B(N^4)$. For some preliminaries results the reader may refer to [24].

Acknowledgements Both authors thank the anonymous referees for their detailed comments that improved the quality of the paper and acknowledge fruitful discussions with Alkiviadis Akritas and Bernard Mourrain. The first author is grateful to Maurice Mignotte for discussions about the separation bound, to Doru Ştefănescu for various discussions and suggestions about the bounds of the positive roots of polynomials, and to Fabrice Rouillier for various discussions about RS and the experiments. Both 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) and through PENED 2003 program, contract nr. 70/03/8473. The program is co-funded by the EU – European Social Fund (75% of public funding), national resources – General Secretariat of Research and Technology of Greece (25% of public funding) as well as the private sector, in the framework of Measure 8.3 of the Community Support Framework.

References

[1] A. Akritas. There is no "Uspensky's method". Extended Abstract. In *Proc. Symposium on Symbolic and Algebraic Computation*, pages 88–90, Waterloo, Ontario, Canada, 1986.

[2] A. Akritas. An implementation of Vincent's theorem. *Numerische Mathematik*, 36:53–62, 1980.

[3] A. Akritas and A. Strzeboński. A comparative study of two real root isolation methods. *Nonlinear Analysis: Modelling and Control*, 10(4): 297–304, 2005.

[4] A. Akritas, A. Bocharov, and A. Strzeboński. Implementation of real root isolation algorithms in Mathematica. In *Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (Interval '94)*, pages 23–27, St. Petersburg, Russia, March 1994.

[5] A. Akritas, W. Strzeboński, and P. S. Vigklas. Implementations of a New Theorem for Computing Bounds for Positive Roots of Polynomials. *Computing*, 78:355–367, 2006.

[6] A. Akritas. *Elements of Computer Algebra with Applications.* J. Wiley & Sons, New York, 1989.

[7] A. Alesina and M. Galuzzi. A new proof of Vincent's theorem. *L'Enseignement Mathématique*, 44:219–256, 1998.

[8] D. Bailey, J. Borwein, and R. Crandall. On the Khintchine Constant. *Mathematics of Computation*, 66:417–431, 1997.

[9] S. Basu, R. Pollack, and M-F.Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics.* Springer-Verlag, 2003. ISBN 3-540-00973-6.

[10] D. Bini. Numerical computation of polynomial zeros by means of Aberth's method. *Numerical Algorithms*, 13(3–4):179–200, 1996.

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

[12] E. Bombieri and A. van der Poorten. Continued fractions of algebraic numbers. In *Computational algebra and number theory (Sydney, 1992)*, pages 137–152. Kluwer Acad. Publ., Dordrecht, 1995.

[13] R. Brent, A. van der Poorten, and H. Riele. A Comparative Study of Algorithms for Computing Continued Fractions of Algebraic Numbers. In Henri Cohen, editor, *ANTS*, LNCS, pages 35–47. Springer, 1996.

[14] R. P. Brent. Analysis of the binary Euclidean algorithmd. In J. F. Traub, editor, *New Directions and Recent Results in Algorithms and Complexity*, pages 321–355. Academic Press, New York, 1976.

[15] R. P. Brent. Twenty years' analysis of the binary Euclidean algorithm. In J. Davies, A. W. Roscoe, and J. Woodcock, editors, *Millenial Perspectives in Computer Science: Proc. of the 1999 Oxford-Microsoft Symposium in honour of Professor Sir Antony Hoare*, pages 41–53. Palgrave, New York,

2000.

[16] D. Cantor, P. Galyean, and H. Zimmer. A continued fraction algorithm for real algebraic numbers. *Mathematics of Computation*, 26(119):785–791, July 1972. ISSN 0025-5718.

[17] J. Chen. A new algorithm for the isolation of real roots of polynomial equations. In *Proc. 2nd International Conference on Computers and Applications*, pages 714–719. IEEE Computer Soc. press, 1987.

[18] 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.

[19] G. 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.

[20] 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.

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

[22] 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. ISBN 1-59593-276-3.

[23] I. Z. Emiris and E. P. Tsigaridas. Computing with real algebraic numbers of small degree. In S. Albers and T. Radzik, editors, *Proc. 12th European Symposium of Algorithms (ESA)*, volume 3221 of *LNCS*, pages 652–663, Bergen, Norway, Sep 14–17 2004. Springer Verlag.

[24] I. Z. Emiris and E. P. Tsigaridas. A note on the complexity of univariate root isolation. Research Report RR-6043, INRIA, Oct 2006. URL `http://hal.inria.fr/inria-00116985/en/`.

[25] I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. The predicates for the Voronoi diagram of ellipses. In *Proc. 22th Annual ACM Symp. on Computational Geometry*, pages 227–236, Sedona, USA, 2006.

[26] 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, 2007. also available in www.inria.fr/rrrt/rr-5897.html.

[27] D. Hensley. The largest digit in the continued fraction expansion of a rational number. *Pacific Journal of Mathematics*, 151(2):237–255, 1991.

[28] H. Hong. Bounds for absolute positiveness of multivariate polynomials. *Journal of Symbolic Computation*, 25(5):571–585, May 1998.

[29] C. P. Hughes and A. Nikeghbali. The zeros of random polynomials cluster uniformly near the unit circle, March 15 2004. URL

http://arxiv.org/abs/math/0406376v3.

[30] I. Ibragimov and D. Zaporozhets. On the distribution of roots of random polynomials. Technical Report CRC 06-060, Collaborative Research Centre 701, Universität Bielefeld, 2006.

[31] I. Ibragimov and O. Zeitouni. On the roots of random polynomials. *Trans. Amer. Math. Soc.*, 349:2427–2441, 1997.

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

[33] A. Khintchine. *Continued Fractions.* University of Chicago Press, Chicago, 1964.

[34] J. Kioustelidis. Bounds for the positive roots of polynomials. *Journal of Computational and Applied Mathematics*, 16:241–244, 1986.

[35] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison-Wesley, Reading, MA, 3rd edition, 1998.

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

[37] W. Krandick and K. Mehlhorn. New bounds for the Descartes method. *JSC*, 41(1):49–66, Jan 2006.

[38] S. Kwek and K. Mehlhorn. Optimal search for rationals. *Information Processing Letters*, 86(1):23–26, 2003.

[39] P. Lévy. Sur les lois de probabilitié dont dependent les quotients complets et incomplets d' une fraction continue. *Bull. Soc. Math.*, 57:178–194, 1929.

[40] M. Michelini. *Kuzmin's Extraordinary Zero Measure Set.* Senior Thesis, Mathematics Department, Princeton University, 2004.

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

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

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

[44] B. Mourrain, J-P. Pavone, P. Trébuchet, and E. P. Tsigaridas. SYNAPS: a library for symbolic-numeric computing. In *Proc. 8th Int. Symp. on Effective Methods in Algebraic Geometry (MEGA)*, Italy, May 2005. (software presentation).

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

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

[47] R. Richtmyer, M. Devaney, and N. Metropolis. Continued fraction expansions of algebraic numbers. *Numerische Mathematik*, 4:68–64, 1962.

[48] D. Rosen and J. Shallit. A continued fraction algorithm for approximating all real polynomial roots. *Math. Mag*, 51:112–116, 1978.

[49] F. Rouillier and Z. Zimmermann. Efficient isolation of polynomial's real

roots. *J. of Computational and Applied Mathematics*, 162(1):33–50, 2004.
[50] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Manuscript. Univ. of Tübingen, Germany, 1982.
[51] V. Sharma. Complexity of real root isolation using Continued Fractions. In C. W. Brown, editor, *Proc. Annual ACM ISSAC*, pages 339–346, Waterloo, Canada, 2007.
[52] D. Ştefănescu. New bounds for the positive roots of polynomials. *Journal of Universal Computer Science*, 11(12):2132–2141, 2005.
[53] E. P. Tsigaridas and I. Z. Emiris. Univariate polynomial real root isolation: Continued fractions revisited. In Y. Azar and T. Erlebach, editors, *Proc. 14th European Symposium of Algorithms (ESA)*, volume 4168 of *LNCS*, pages 817–828, Zurich, Switzerland, 2006. Springer Verlag.
[54] J. V. Uspensky. *Theory of Equations*. McGraw-Hill, 1948.
[55] A. van der Poorten. An introduction to continued fractions. In *Diophantine analysis*, pages 99–138. Cambridge University Press, 1986.
[56] A. van der Sluis. Upper bounds for the roots of polynomials. *Numerische Mathematik*, 15:250–262, 1970.
[57] A. J. H. Vincent. Sur la résolution des équations numériques. *J. Math. Pures Appl.*, 1:341–372, 1836.
[58] J. von zur Gathen and J. Gerhard. Fast Algorithms for Taylor Shifts and Certain Difference Equations. In *ISSAC*, pages 40–47, 1997.
[59] X. Wang. A method for isolating roots of algebraic equations. Number N. 1. 1960. Univ. Academic Press.
[60] C. K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.