Univariate Polynomial Real Root Isolation: Continued Fractions Revisited

Elias P. Tsigaridas and Ioannis Z. Emiris

Department of Informatics and Telecommunications,
National Kapodistrian University of Athens, HELLAS
{et, emiris}@di.uoa.gr

Abstract. We present algorithmic, complexity and implementation results concerning real root isolation of integer univariate polynomials using the continued fraction expansion of real numbers. We improve the previously known bound by a factor of $d\tau$, where d is the polynomial degree and τ bounds the coefficient bitsize, thus matching the current record complexity for real root isolation by exact methods. Namely, the complexity bound is $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ using a standard bound on the expected bitsize of the integers in the continued fraction expansion. We show how to compute the multiplicities within the same complexity and extend the algorithm to non square-free polynomials. Finally, we present an efficient open-source C++ implementation in the algebraic library SYNAPS, and illustrate its efficiency as compared to other available software. We use polynomials with coefficient bitsize up to 8000 and degree up to 1000.

1 Introduction

In this paper we deal with real root isolation of univariate integer polynomials, 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. We use the continued fraction expansion of *real algebraic numbers*. Recall that such a number is a real root of an integer polynomial.

One motivation is to explain the method's good performance in implementations, despite the higher complexity bound which was known until now. Indeed, we show that continued fractions lead to (expected) asymptotic bit complexity bounds that match those recently proven for other exact methods, such as Sturm sequences and Descartes' subdivision.

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 $A = \sum_{i=0}^{d} a_i X^i \in \mathbb{Z}[X]$, $\mathtt{deg}(A)$ 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 $\mathtt{a} \in \mathbb{Q}$, $\mathcal{L}(\mathtt{a}) \geq 1$ 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. $Var(A)$ denotes the sign variations

Y. Azar and T. Erlebach (Eds.): ESA 2006, LNCS 4168, pp. 817–828, 2006.

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.

Previous work and our results: Real root isolation of univariate integer polynomials is a well known problem with a huge bibliography and we only scratch the surface of it. We encourage the reader to refer to the references.

Exact subdivision based algorithms for real root isolation are based either on Descartes' rule of sign 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 root is contained. Quite recently it was proven (cf [12, 13] and references therein) that both approaches (Descartes and Sturm), achieve the same bit complexity bound, namely $\widetilde{\mathcal{O}}_B(d^4\tau^2)$, where d is the polynomial degree and τ bounds the coefficient bitsize, or $\widetilde{\mathcal{O}}_B(N^6)$, where $N = \max\{d, \tau\}$. Moreover using Sturm sequences in a pre-processing and a post-processing step [14, 15] the bound holds for the non square-free case and the multiplicities of the roots can also be computed.

The continued fraction algorithm (from now on called CF) differs from the subdivision algorithms in that instead of bisecting a given initial interval it computes the continued fraction expansions of the real roots of the polynomial. The first formulation of CF is due to Vincent [32], see also [2] for historical references, based on his theorem (Th. 3 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 implies the transformed polynomial has zero (resp. one) real root in $(0, \infty)$. Unfortunately Vincent's algorithm is exponential [9].

Uspensky [29] 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. Using Vincent's theorem, Collins and Akritas [9] derived a polynomial subdivision-based algorithm using Descartes' rule of sign. Akritas [5, 1] dealt with the exponential behavior of CF, by computing the partial quotients as positive lower bounds of the positive real roots, via Cauchy's bound (for details, see Sec. 4), and obtained a complexity of $\widetilde{\mathcal{O}}_B(d^5\tau^3)$ or $\widetilde{\mathcal{O}}_B(N^8)$, without using fast Taylor shifts [33]. However, it is not clear how this approach accounts for the increased coefficient size in the transformed polynomial after applying $X \mapsto b + X$. Another issue is to bound the size of the partial quotients. Refer to Eq. (1) which indicates that the *magnitude* of the partial quotients is unbounded. CF is the standard real root isolation algorithm in MATHEMATICA [3]. For some experiments against subdivision-based algorithms, in MATHEMATICA, the reader may refer to [4].

Another class of univariate solvers are numerical solvers, e.g. [24, 6] that compute an approximation of all the roots of a polynomial up to a desired accuracy. The complexity of these algorithms is $\widetilde{\mathcal{O}}_B(d^3\,\tau)$ or $\widetilde{\mathcal{O}}_B(N^4)$.

The contributions of this paper are the following: First, we improve the bound of the number of steps (transformations) that the algorithm performs. Second, we bound the bitsize of the partial quotients and thus the growth of the transformed polynomials which appear during the algorithm. We revisit the proof of [5, 1] so

as to improve the overall bit complexity bound of the algorithm to $\widetilde{\mathcal{O}}_B(N^6)$, thus matching the current record complexity for real root isolation. The extension to the non square-free case uses the techniques from [14, 15]. Third, we present our efficient open-source C++ implementation in SYNAPS[1] [23], and illustrate it on various data sets, including polynomials of degree up to 1000 and coefficients of 8000 bits. We performed experiments against RS[2], which seems to be one of the fastest available software for exact real root isolation and against ABERTH [6], a numerical solver available through SYNAPS. Our implementation seems to have the best performance in practice. We believe that our software contributes towards reducing the gap between rational and numeric computation, the latter being usually perceived as faster.

The rest of the paper is structured as follows. The next section sketches the theory behind continued fractions. Sec. 3 presents the CF algorithm and Sec. 4 its analysis. 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 [30] which although is far from complete suffices for our purposes. The reader may refer to e.g [5, 34, 7, 30]. 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. By considering the recurrent relations

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

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

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

[1] www-sop.inria.fr/galaad/logiciels/synaps/
[2] fgbrs.lip6.fr/salsa/Software/index.php

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

Let $\gamma = [c_0, c_1, \dots]$ be the continued fraction expansion of a real number. The Gauss-Kuzmin distribution [7, 25] states that for almost all real numbers γ (the set of exceptions has Lebesgue measure zero) the probability for a positive integer δ to appear as an element in the continued fraction expansion of γ is

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

The Gauss-Kuzmin law induces that we can not bound the mean value of the partial quotients, i.e $E[c_i] = \sum_{\delta=1}^{\infty} \delta\, Prob[c_i = \delta] = \infty$, $i > 0$. However the geometric (and the harmonic) mean is not only asymptotically bounded, but is bounded by a constant. For the geometric mean this is the famous Khintchine's constant [17], 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 expected value of the bitsize of the partial quotients is a constant for almost all real numbers, when $n \to \infty$ or n sufficiently big [17, 25]. Following closely [25], we have: $E[\ln c_i] = \frac{1}{n}\sum_{i=1}^{n} \ln c_i = \ln \mathcal{K} = 0.98785...$, 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. "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" [8], and also various experimental results [7, 25, 26] suggest so.

3 The CF Algorithm

Theorem 1 (Budan). *[20, 5] Let $A \in \mathbb{R}[X]$ and $a < b$, where $a, b \in \mathbb{R}$. Let A_a (A_b) be the polynomial produced after we apply the map $X \mapsto X+a$ ($X \mapsto X+b$) to A. Then the followings hold: (i) $Var(A_a) \geq Var(A_b)$, (ii) $\#\{\gamma \in (a,b) | A(\gamma) = 0\} \leq Var(A_a) - Var(A_b)$ and (iii) $\#\{\gamma \in (a,b) | A(\gamma) = 0\} \equiv Var(A_a) - Var(A_b)$* mod 2.

Theorem 2 (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.

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 CF algorithm depends on the following theorem, which dates back to Vincent's theorem in 1836 [32]. It is a very interesting question whether the one and two circle theorems (cf [19] and references therein), employed in the analysis of the subdivision-based real-root isolation algorithm [9], can also be applied and possibly improve the complexity of CF.

Theorem 3. *[5, 29] Let $A \in \mathbb{Z}[X]$, with $\mathtt{deg}(A) = d$ and let Δ be the separation bound. Let n be the smallest index such that $F_{n-1}\Delta > 2$ and $F_{n-1}F_n\Delta > 1 + \frac{1}{\epsilon_d}$, where F_n is the n-th Fibonnaci number and $\epsilon_d = (1 + \frac{1}{d})^{\frac{1}{d-1}} - 1$. 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 $A_n(X)$, which has no more than one sign variation.*

Remark 1. Since $\frac{3}{4d^2} < \epsilon_d < \frac{4}{d^2}$ [10] we conclude that $\frac{1}{\epsilon_d} + 1 < 2d^2$ for $d \geq 2$. Thus, if $d \geq 2$ we can replace the two conditions of Th. 3 by $F_{n-1}\Delta \geq 2d^2$, since $F_n \geq F_{n-1} \geq 1$ and $F_{n-1}F_n\Delta \geq F_{n-1}\Delta \geq 2d^2 > 2$.

Th. 3 can be used 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 CF 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. 2). If $Var(A_1) < Var(A)$ then (possibly) there are real roots of A in $(0, 1)$, due to Budan's theorem (Th. 1). 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. 2). Uspensky's [29] variant of the algorithm (see also [26]) at every step produces both polynomials A_1 and A_2, probably, as Akritas states [2], because he was unaware of Budan's theorem (Th. 1). In both variants, if the transformed polynomial has more than one sign variations, we repeat the process.

We may consider the process of CF as an infinite binary tree in which the root corresponds to the initial polynomial A. The branch from a node to a right (left) child corresponds to the map $X \mapsto X + 1$ ($X \mapsto \frac{1}{1+X}$). Vincent's algorithm (and Uspensky's) results to a sequence of transformations as in Th. 3, and so the leaves of the tree hold (transformed) polynomials that have no more than one sign variations, if Th. 3 holds. Akritas [1, 5] 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, using Cauchy's bound [5, 34]. 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 bitsize increases after a shift operation. Another issue is to bound the size of b.

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. 3, using positive integers $c_0, c_1, \ldots, c_n$. Using the convergents, this transformation becomes

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

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 [5, 34] 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

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

Input: $A \in \mathbb{Z}[X], M(X) = \frac{kX+l}{mX+n}$, $k, l, m, n \in \mathbb{Z}$

```
1  if A(0) = 0 then
2  |  OUTPUT Interval( M(0), M(0)) ;
3  |  A ← A(X)/X;
4  |  CF(A, M);
5  if Var(A) = 0 then RETURN ;
6  if Var(A) = 1 then OUTPUT Interval( M(0), M(∞)), RETURN ;
7  b ← PLB(A) // PLB ≡ PositiveLowerBound ;
8  if b > 1 then A ← A(b + X), M ← M(b + X) ;
9  A1 ← A(1 + X), M1 ← M(1 + X) ;
10 CF(A1, M1) // Looking for real roots in (1, +∞);
11 A2 ← A(1/(1+X)), M2 ← M(1/(1+X)) ;
12 CF(A2, M2) // Looking for real roots in (0, 1) ;
```

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 CF is presented in Alg. 1. The `Interval` function orders the endpoints of the computed isolating interval and $\texttt{PLB}(A)$ computes a lower bound on the positive roots of A. The input of the algorithm is a polynomial $A(X)$ and the trivial transformation $M(X) = X$. Notice that Lines 11 and 12 are 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.

4 The Complexity of the CF Algorithm

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 [34, 20, 21]. We prove the following theorem, which is based on a theorem by Mignotte [20], thus extending [11, 13].

Theorem 4. *Let $A \in \mathbb{Z}[X]$, with $\deg(A) = d$ and $\mathcal{L}(A) = \tau$. Let Ω be any set of k pairs of indices (i, j) such that $1 \le i < j \le d$ and let the non-zero (complex) roots of A be $0 < |\gamma_1| \le |\gamma_2| \le \cdots \le |\gamma_d|$. Then*

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

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

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

and the fact [20, 21] that for any root of A, $\frac{1}{\mathcal{M}(A)} \le |\gamma_i| \le \mathcal{M}(A)$. In order to prove the left inequality

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

Recall [34, 20] that $\mathtt{disc}(A) = \mathtt{lead}(A)^{2d-2} \prod_{i<j} (\gamma_i - \gamma_j)^2$. For the right inequality we consider the absolute value of the discriminant of A:

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

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

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

where we used the inequality $|\gamma_1|^0 |\gamma_2|^1 \cdots |\gamma_d|^{d-1} \le |\mathcal{M}(A)/\,\mathtt{lead}(A)\,|^{d-1}$, and the fact [20] that, since $\forall i$, $|\gamma_i| \ge \mathcal{M}(A)^{-1}$, we have $\prod_{j\in\overline{\Omega}} |\gamma_j| \ge |\gamma_1|^k \ge \mathcal{M}(A)^{-k}$. Thus $\prod_{(i,j)\in\Omega} |\gamma_i - \gamma_j| \ge 2^{k-\frac{d(d-1)}{2}} \mathcal{M}(A)^{1-d-k} \sqrt{|\,\mathtt{disc}(A)|}$. □

A similar theorem but with more strict hypotheses on the roots first appeared in [11] and the conditions were generalized in [13]. Th. 4 has a factor 2^{d^2} instead of d^d in [11, 13], which plays no role when $d = \mathcal{O}(\tau)$ or when notation with N is used. Possibly a more involved proof of Th. 4 may eliminate this factor [22].

Remark 2. There is a simple however crucial observation about Th. 3. When the transformed polynomial has one (zero) sign variation, then the interval with endpoints $\frac{P_{n-1}}{Q_{n-1}} = [c_0, \ldots, c_{n-1}]$ and $\frac{P_n}{Q_n} = [c_0, \ldots, c_n]$ isolates a positive real root (a complex root with positive real part) of A, say γ_i. Then, in order for Th. 3 to hold, it suffices to consider, instead of the separation bound Δ, the quantity $|\gamma_i - \gamma_{c_i}|$, where γ_{c_i} is the (complex) root of A closest to γ_i.

Theorem 5. *The CF algorithm performs at most $\mathcal{O}(d^2 + d\tau)$ steps.*

Proof. Let $0 < |\gamma_1| \le \cdots \le |\gamma_k|$, $k \le d$ 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 consider the binary tree T generated during the execution of CF. The number of steps of CF corresponds to the number of nodes in T, which we denote by $\#(T)$. We use some arguments and the notation from [13] in order to prune T.

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 **type-i** if its interval I_u contains $i \ge 0$ real roots. Since the algorithm terminates the leaves are type-0 or type-1. We will prune certain leaves of T so as to obtain a certain subtree T'. 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 type-1, we arbitrary prune one of them. If one of them is type-1 then we prune the other. If both leaves are 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 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. 2) and the number of nodes from a leaf to the root is n_i, which is such that the condition of Rem. 1 is satisfied. Since n_i is the smallest index such that the condition of Rem. 1 holds, if we reduce n_i by one then the inequality does not hold. Thus

$$F_{n_i-2}|\gamma_i - \gamma_{c_i}| \leq 2d^2 \Rightarrow \phi^{n_i-3}|\gamma_i - \gamma_{c_i}| < 2d^2 \Rightarrow n_i < 4 + 2\lg d - \lg|\gamma_i - \gamma_{c_i}|$$

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

$$\#(T') \leq \sum_{i=1}^{k} n_i \leq 2k(2+\lg d) - \sum_{i=1}^{k} \log|\gamma_i - \gamma_{c_i}| \leq 2k(2+\lg d) - \log\prod_{i=1}^{k}|\gamma_i - \gamma_{c_i}| \quad (6)$$

So as to use Th. 4 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 the worst case is when the product consists only of symmetric products i.e $\prod_{i=1}^{k/2}|(\gamma_j - \gamma_{c_j})(\gamma_{c_j} - \gamma_j)|$. Thus we consider the square of the inequality of Th. 4 taking $\frac{k}{2}$ instead of k and $\mathtt{disc}(A) \geq 1$ (since A is square-free), thus

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

Eq. (6) becomes $\#(T') \leq 2k(2+\lg d) + d^2 - d - k + (2d+k-2)\lg\mathcal{M}(A)$. 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 $\#(T') \leq 2k(2+\lg d) + d^2 - d - k + (2d+k-2)(\tau + \lg d)$. Since $\#(T) < 2\,\#(T')$ and $k \leq d$, we conclude that $\#(T) = \mathcal{O}(d^2 + d\,\tau + d\lg d)$. □

To complete the analysis of CF 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 7) and three transformations, $X \mapsto b + X$, $X \mapsto 1 + X$ and $X \mapsto \frac{1}{1+X}$ (Lines 8, 9 and 11 in Alg. 1). Since inversion can be performed in $\mathcal{O}(d)$, the complexity is dominated by the cost of the shift operation (Line 8 in Alg. 1) if a small number of calls to `PLB` is needed in order to compute a partial quotient. We will justify this in the end of the section. We also will use the following theorem:

Theorem 6 (Fast Taylor shift). *[33] Let $A \in \mathbb{Z}[X]$, with $\deg(A) = d$ and $\mathcal{L}(A) = \tau$ and let $a \in \mathbb{Z}$, such that $\mathcal{L}(a) = \sigma$. 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 $\mathcal{L}(B) = \mathcal{O}(\tau + d\sigma)$.*

Initially A has degree d and bitsize τ. Evidently the degree does not change after a shift operation. Each shift operation by a number of bitsize b_h increases the bit size of the polynomial by an additive factor $d\,b_h$, in the worst case (Th. 6). 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}. Th. 6 states that this can be done in $\mathcal{O}_B\left(\mathsf{M}\left(d^2\lg d + d^2 b_{h+1} + d(\tau + d\sum_{i=1}^{h} b_i)\right)\right)$.

Table 1. Experimental results

		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	*	*	*	*	*
	ABERTH	0.04	0.78	3.24	?	?	?	?	?	?	?
	#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

In order to bound $\sum_{i=1}^{h+1} b_i$ 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. 5), the (expected) worst case cost of step h is $\mathcal{O}_B(\mathsf{M}\left(d^2 \lg d + d\tau + d^2(d^2 + d\tau)\right))$ 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 $d = \mathcal{O}(\tau)$.

Now consider $A_{in} \in \mathbb{Z}[X]$, not necessarily square-free, with $\mathtt{deg}(A_{in}) = d$ and $\mathcal{L}(A_{in}) = \tau$. Following [14, 15] we compute the square-free part A of A_{in} using Sturm-Habicht sequences in $\widetilde{\mathcal{O}}_B(d^2\tau)$ and $\mathcal{L}(A) = \mathcal{O}(d+\tau)$. Using CF we isolate the positive real root 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 $\widetilde{\mathcal{O}}_B(d^3\tau)$, it is possible to find the multiplicities in $\widetilde{\mathcal{O}}_B(d^3\tau)$. The previous discussion leads to the following theorem:

Theorem 7. *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 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 $N = \max\{d, \tau\}$.*

Rational roots and PLB (Positive Lower Bound) realization: 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. Treating them as real algebraic numbers

leads to an overestimation of the number of iterations. There is one exception to this good behavior of rational roots, namely when they are very large, well separated, and we are interested in practical complexity [3], since then PLB must be applied many times. In [25], the authors performed a small number of Newton iterations in order to have a good approximation of a partial quotient. In [3, 4], this problem was solved by performing the transformation $X \mapsto bX$, where b is the computed bound, whenever $b \geq 16$. We follow the latter approach so, after Line 8 in Alg. 1, if $b = \texttt{PLB}(A) \geq 16$, we apply $X \mapsto bX$ to polynomial A.

$\texttt{PLB}(A)$ is computed as the inverse of an upper bound on the roots of $X^d A(\frac{1}{X})$. In general $\texttt{PLB}(A)$ is applied more than once in order to compute some c_i. However this number is very small [5, 1]. Eq. (1) implies that the probability that a partial quotient is ≤ 10 is ~ 0.87, thus in general the partial quotients are of small magnitude. In order to implement PLB we set $\texttt{PLB}(A) = 2 \max_{a_j<0} |\frac{a_j}{a_d}|^{1/j}$, which is nearly optimal [18]. Actually this bound "[...] is to be recommended among all"[31]. In our implementation we compute PLB only as powers of 2 so that we can take advantage of fast operations as in [27]. Notice that PLB is not a general bound on the roots, but a bound on the positive roots only, see [18, 28].

5 Implementation and Experiments

We have implemented CF in SYNAPS [23], which is a C++ library for symbolic-numeric computations. The implementation is based on GMP[3] (v. 4.1.4) and uses only transformations of the form $X \mapsto 2^\beta X$ and $X \mapsto X + 1$. We consider square-free polynomials of degree $\in \{100, 200, \ldots, 1000\}$. Following [27], the first class of experiments concerns well-known ill-conditioned polynomials: 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 products, $\left(X^d - 2(101X - 1)^2\right)\left(X^d - 2((101 + \frac{1}{101})X - 1)^2\right)$, of two such polynomials (M2). 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 that implements a subdivision-based algorithm using Descartes' rule of sign with several optimizations and symbolic-numeric techniques [27]. We used RS through its MAPLE interface and with default options. Timings were reported by its function `rs_time()`. We also test ABERTH [6], 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 (real and complex) roots. Unfortunately, we were not always able to tune its behavior in order to produce the correct number of real roots in all the cases.

So, in Table 1, we report experiments with CF, RS and ABERTH, where the timings are in seconds. The asterisk (*) denotes that the computation did not

[3] www.swox.com/gmp/

finish after 12000s and the question-mark (?) that we were not able to tune ABERTH. The experiments were performed on a 2.6GHz PIII with 1GB RAM, 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 is comparable to 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 roots, thus the semi-numerical techniques of RS isolate all roots using only 63 bits of accuracy. ABERTH is even faster on these experiments. However, even in this case, CF is only a little slower than ABERTH. Finally, we tested a univariate polynomial that appears in the Voronoi diagram of ellipses [16]. The polynomial has degree 184, coefficient bitsize 903, and 8 real roots. CF solves it in 0.12s, RS in 0.3s and ABERTH in 1.7s. We have to mention, as F. Rouillier pointed out to us, that RS can be about 30% faster in (L), (C1) and (C2), if we use it with the (non-default) option `precision=0`.

Acknowledgments. Both authors acknowledge fruitful discussions with A. Akritas and B. Mourrain. The first author is also grateful to M. Mignotte, F. Rouillier and D. Stefanecu for various discussions and suggestions. 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).

References

1. A. Akritas. An implementation of Vincent's theorem. *Numerische Mathematik*, 36:53–62, 1980.
2. A. Akritas. There is no "Uspensky's method". Extended Abstract. In *Proc. Symp. on Symbolic and Algebraic Computation*, pp. 88–90, Waterloo, Canada, 1986.
3. A. Akritas, A. Bocharov, and A. Strzébonski. Implementation of real root isolation algorithms in Mathematica. *Abstracts of Interval'94*, pp. 23–27, Russia, 1994.
4. A. Akritas and A. Strzebonski. A comparative study of two real root isolation methods. *Nonlinear Analysis: Modelling and Control*, 10(4):297–304, 2005.
5. A.G. Akritas. *Elements of Computer Algebra with Applications*. J. Wiley & Sons, New York, 1989.
6. D. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, pp. 127–173, 2000.
7. E. Bombieri and A. van der Poorten. Continued fractions of algebraic numbers. In *Computational algebra and number theory*, pp. 137–152. Kluwer, Dordrecht, 1995.
8. 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*, volume 1122 of *LNCS*, pp. 35–47. Springer, 1996.

9. G. Collins and A. Akritas. Polynomial real root isolation using Descartes' rule of signs. In *SYMSAC '76*, pp. 272–275, New York, USA, 1976. ACM Press.
10. 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*, pp. 83–94. Springer-Verlag, Wien, 2nd edition, 1982.
11. J. H. Davenport. Cylindrical algebraic decomposition. Technical Report 88–10, School of Mathematical Sciences, University of Bath, England, 1988.
12. 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*, pp. 81–93, School of Science, Beihang University, Beijing, China, 2005.
13. A. Eigenwillig, V. Sharma, and C. Yap. Almost tight complexity bounds for the Descartes method. (to appear in ISSAC 2006), 2006.
14. I. Emiris and E. P. Tsigaridas. Computations with one and two algebraic numbers. Technical report, ArXiv, Dec 2005.
15. I. Z. Emiris, B. Mourrain, and E. P. Tsigaridas. Real Algebraic Numbers: Complexity Analysis and Experimentation. RR 5897, INRIA, Apr 2006.
16. I.Z. Emiris, E.P. Tsigaridas, and G.M. Tzoumas. The predicates for the Voronoi diagram of ellipses. In *Proc. 24th Annual ACM SoCG*, pp. 227–236, 2006.
17. A. Khintchine. *Continued Fractions*. University of Chicago Press, Chicago, 1964.
18. J. Kioustelidis. Bounds for the positive roots of polynomials. *Journal of Computational and Applied Mathematics*, 16:241–244, 1986.
19. W. Krandick and K. Mehlhorn. New bounds for the Descartes method. *JSC*, 41(1):49–66, Jan 2006.
20. M. Mignotte. *Mathematics for computer algebra*. Springer-Verlag, New York, 1991.
21. M. Mignotte and D. Stefanescu. *Polynomials*. Springer, 1999.
22. M. Mignotte. On the Distance Between the Roots of a Polynomial. *Appl. Algebra Eng. Commun. Comput.*, 6(6):327–332, 1995.
23. B. Mourrain, J. P. Pavone, P. Trébuchet, and E. Tsigaridas. SYNAPS, a library for symbolic-numeric computation. In *8th MEGA*, Italy, 2005. Software presentation.
24. V. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM Rev.*, 39(2):187–220, 1997.
25. R. Richtmyer, M. Devaney, and N. Metropolis. Continued fraction expansions of algebraic numbers. *Numerische Mathematik*, 4:68–64, 1962.
26. D. Rosen and J. Shallit. A continued fraction algorithm for approximating all real polynomial roots. *Math. Mag*, 51:112–116, 1978.
27. F. Rouillier and Z. Zimmermann. Efficient isolation of polynomial's real roots. *J. of Computational and Applied Mathematics*, 162(1):33–50, 2004.
28. D. Stefanescu. New bounds for the positive roots of polynomials. *Journal of Universal Computer Science*, 11(12):2132–2141, 2005.
29. J. V. Uspensky. *Theory of Equations*. McGraw-Hill, 1948.
30. A. van der Poorten. An introduction to continued fractions. In *Diophantine analysis*, pp. 99–138. Cambridge University Press, 1986.
31. A. van der Sluis. Upper bounds for the roots of polynomials. *Numerische Mathematik*, 15:250–262, 1970.
32. A. J. H. Vincent. Sur la résolution des équations numériques. *J. Math. Pures Appl.*, 1:341–372, 1836.
33. J. von zur Gathen and J. Gerhard. Fast Algorithms for Taylor Shifts and Certain Difference Equations. In *ISSAC*, pp. 40–47, 1997.
34. C.K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.