Nonlinear Analysis: Modelling and Control, 2008, Vol. 13, No. 3, 265–279

Improving the Performance of the Continued Fractions Method Using New Bounds of Positive Roots

A. G. Akritas[1], **A. W. Strzeboński**[2], **P. S. Vigklas**[1]

[1]University of Thessaly, Department of Computer and Communication Engineering
GR-38221 Volos, Greece
{akritas, pviglas}@uth.gr

[2]Wolfram Research, Inc., 100 Trade Center Drive, Champaign, IL 61820, USA
adams@wolfram.com

Received: 13.02.2008 **Revised:** 09.06.2008 **Published online:** 28.08.2008

Abstract. In this paper we compare four implementations of the Vincent-Akritas-Strzeboński Continued Fractions (*VAS-CF*) real root isolation method using four different (two linear and two quadratic complexity) bounds on the values of the positive roots of polynomials. The quadratic complexity bounds were included to see if the quality of their estimates compensates for their quadratic complexity. Indeed, experimentation on various classes of special and random polynomials revealed that the *VAS-CF* implementation using *LMQ*, the *Q*uadratic complexity variant of our *L*ocal *M*ax bound, achieved an overall average speed-up of $40\,\%$ over the original implementation using Cauchy's linear bound.

Keywords: Vincent's theorem, polynomial real root isolation, continued fractions method, upper bounds on positive roots, linear and quadratic complexity bounds.

1 Introduction

We begin by first reviewing some basic facts about the *VAS-CF* method for isolating the positive roots of polynomials. This method is based on Vincent's theorem of 1836, [1], which states:

Theorem 1. *If in a polynomial, $p(x)$, of degree n, with rational coefficients and without multiple roots we perform sequentially replacements of the form*

$$x \leftarrow \alpha_1 + \frac{1}{x}, \;\; x \leftarrow \alpha_2 + \frac{1}{x}, \;\; x \leftarrow \alpha_3 + \frac{1}{x}, \ldots,$$

where $\alpha_1 \geq 0$ is an arbitrary non negative integer and $\alpha_2, \alpha_3, \ldots$ are arbitrary positive integers, $\alpha_i > 0$, $i > 1$, then the resulting polynomial either has no sign variations or it has one sign variation. In the last case the equation has exactly one positive root, which

is represented by the continued fraction

$$\alpha_1 + \cfrac{1}{\alpha_2 + \cfrac{1}{\alpha_3 + \cfrac{1}{\ddots}}},$$

whereas in the first case there are no positive roots.

See the papers by Alesina & Galuzzi, [2] and Chapter 7 in [3] for a complete historical survey of the subject and implementation details respectively. The thing to note is that the quantities α_i (the partial quotients of the continued fraction) are computed by repeated application of a method for estimating *lower* bounds[1] on the values of the positive roots of a polynomial.

Cauchy's (linear complexity) bound on the values of the positive roots of a polynomial, was used until recently in the *VAS-CF* real root isolation method, [4]. In the SYNAPS implementation of the *VAS-CF* method, [5], Emiris and Tsigaridas used Kioustelidis' (linear complexity) bound, [6] and independently verified the results obtained by Akritas and Strzeboński, [4][2].

In this paper we implemented the *VAS-CF* real root isolation process using both linear *and* quadratic complexity bounds on the values of the positive roots of a polynomial — all of which are based on Theorem 2 below, [9], [10], [11] — and discovered that the latter substantially improve its performance!

The rest of the paper is structured as follows:

In Section 2 we present the *VAS-CF* algorithm, as found in [4]; this is done both for completeness of the present paper *and* to correct a misprint that appeared in Step 5 in [4]. We also present the theoretical background of our new quadratic complexity bounds.

In Section 3, we compare four different implementations of the *VAS-CF* algorithm using two linear and two quadratic complexity bounds — all suitably adjusted for computing lower bounds on the values of the positive roots.

Finally, in Section 4 we present our conclusions; namely, when *LMQ*, the *Q*uadratic complexity variant of our *L*ocal *M*ax bound, is implemented in *VAS-CF*, an overall average speed-up of $40\,\%$ is achieved over the original version of *VAS-CF*, where Cauchy's linear bound is implemented.

2 Algorithmic and theoretical background

In Subsection 2.1 we present the *VAS-CF* algorithm — as found in [4] — and correct a misprint in Step 5 that had appeared in that presentation; moreover, we explain where the new bound on the positive roots is used. We also present our Theorem 2 from which *all* the bounds on the values of the positive roots are derived. Linear complexity bounds

[1] A *lower* bound, ℓb, on the values of the positive roots of a polynomial $f(x)$, of degree *n*, is found by first computing an *upper* bound, ub, on the values of the positive roots of $x^n f(\frac{1}{x})$ and then setting $\ell b = \frac{1}{ub}$.

[2] See also Sharma's work, [7] and [8], where he used the worst possible positive lower bound to prove that the *VAS-CF* method is still polynomial in time!

are presented in Subsection 2.2, whereas the quadratic complexity ones are presented in Subsection 2.3.

2.1 Description of the Continued Fractions Algorithm VAS-CF

Using the notation of paper [4], let $f \in Z[x] \setminus \{0\}$. By $sgc(f)$ we denote the number of sign changes in the sequence of nonzero coefficients of f. For nonnegative integers a, b, c, and d, such that $ad - bc \neq 0$, we put

$$intrv(a, b, c, d) := \Phi_{a,b,c,d}\big((0, \infty)\big),$$

where

$$\Phi_{a,b,c,d}\colon (0, \infty) \ni x \longrightarrow \frac{ax + b}{cx + d} \in \left(min\Big(\frac{a}{c}, \frac{b}{d}\Big), max\Big(\frac{a}{c}, \frac{b}{d}\Big)\right),$$

and by *interval data* we denote a list

$$\{a, b, c, d, p, s\},$$

where p is a polynomial such that the roots of f in $intrv(a, b, c, d)$ are images of positive roots of p through $\Phi_{a,b,c,d}$, and $s = sgc(p)$.

The value of parameter α_0 used in Step 4 below needs to be chosen empirically. In our implementation $\alpha_0 = 16$.

Algorithm Continued Fractions (VAS-CF)

Input: A squarefree polynomial $f \in Z[x] \setminus \{0\}$
Output: The list $rootlist$ of the isolation intervals of the positive roots of f

1. Set $rootlist$ to an empty list. Compute $s \leftarrow sgc(f)$. If $s = 0$ return an empty list. If $s = 1$ return $\{(0, \infty)\}$. Put interval data $\{1, 0, 0, 1, f, s\}$ on $intervalstack$.

2. If $intervalstack$ is empty, return $rootlist$, else take interval data $\{a, b, c, d, p, s\}$ off $intervalstack$.

3. Compute a lower bound $\alpha \in \mathbb{Z}$ on the positive roots of p.

4. If $\alpha > \alpha_0$ set $p(x) \leftarrow p(\alpha x)$, $a \leftarrow \alpha a$, $c \leftarrow \alpha c$, and $\alpha \leftarrow 1$.

5. If $\alpha \geq 1$, set $p(x) \leftarrow p(x + \alpha)$, $b \leftarrow \alpha a + b$, and $d \leftarrow \alpha c + d$. If $p(0) = 0$, add $[b/d, b/d]$ to $rootlist$, and set $p(x) \leftarrow p(x)/x$. Compute $s \leftarrow sgc(p)$. If $s = 0$ go to Step 2. If $s = 1$ add $intrv(a, b, c, d)$ to $rootlist$ and go to Step 2.

6. Compute $p_1(x) \leftarrow p(x + 1)$, and set $a_1 \leftarrow a$, $b_1 \leftarrow a + b$, $c_1 \leftarrow c$, $d_1 \leftarrow c + d$, and $r \leftarrow 0$. If $p_1(0) = 0$, add $[b_1/d_1, b_1/d_1]$ to $rootlist$, and set $p_1(x) \leftarrow p_1(x)/x$, and $r \leftarrow 1$. Compute $s_1 \leftarrow sgc(p_1)$, and set $s_2 \leftarrow s - s_1 - r$, $a_2 \leftarrow b$, $b_2 \leftarrow a + b$, $c_2 \leftarrow d$, and $d_2 \leftarrow c + d$.

7. If $s_2 > 1$, compute $p_2(x) \leftarrow (x+1)^m p(\frac{1}{x+1})$, where m is the degree of p. If $p_2(0) = 0$, set $p_2(x) \leftarrow p_2(x)/x$. Compute $s_2 \leftarrow sgc(p_2)$.
8. If $s_1 < s_2$, swap $\{a_1, b_1, c_1, d_1, p_1, s_1\}$ with $\{a_2, b_2, c_2, d_2, p_2, s_2\}$.
9. If $s_1 = 0$ goto Step 2. If $s_1 = 1$ add $intrv(a_1, b_1, c_1, d_1)$ to $rootlist$, else put interval data $\{a_1, b_1, c_1, d_1, p_1, s_1\}$ on $intervalstack$.
10. If $s_2 = 0$ goto Step 2. If $s_2 = 1$ add $intrv(a_2, b_2, c_2, d_2)$ to $rootlist$, else put interval data $\{a_2, b_2, c_2, d_2, p_2, s_2\}$on $intervalstack$. Go to Step 2.

Please note that the lower bound, α, on the positive roots of $p(x)$ is computed in Step 3, and used in Step 5.

As mentioned in the introduction, Cauchy's bound was the first one to be used in *VAS-CF*. However, new bounds on the values of the positive roots were developed by us recently; for details see [9], [10], and [11]. The important thing that emerged from our work is that *all* bounds on the values of the positive roots are derived from the following [11]:

Theorem 2. *Let* $p(x)$

$$p(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \ldots + \alpha_0 \quad (\alpha_n > 0) \tag{1}$$

be a polynomial with real coefficients and let $d(p)$ *and* $t(p)$ *denote the degree and the number of its terms, respectively.*

Moreover, assume that $p(x)$ *can be written as*

$$p(x) = q_1(x) - q_2(x) + q_3(x) - q_4(x) + \ldots + q_{2m-1}(x) - q_{2m}(x) + g(x), \tag{2}$$

where all the polynomials $q_i(x)$, $i = 1, 2, \ldots, 2m$ *and* $g(x)$ *have only positive coefficients. In addition, assume that for* $i = 1, 2, \ldots, m$ *we have*

$$q_{2i-1}(x) = c_{2i-1,1} x^{e_{2i-1,1}} + \ldots + c_{2i-1,t(q_{2i-1})} x^{e_{2i-1,t(q_{2i-1})}}$$

and

$$q_{2i}(x) = b_{2i,1} x^{e_{2i,1}} + \ldots + b_{2i,t(q_{2i})} x^{e_{2i,t(q_{2i})}},$$

where $e_{2i-1,1} = d(q_{2i-1})$ *and* $e_{2i,1} = d(q_{2i})$ *and the exponent of each term in* $q_{2i-1}(x)$ *is greater than the exponent of each term in* $q_{2i}(x)$. *If for all indices* $i = 1, 2, \ldots, m$, *we have*

$$t(q_{2i-1}) \geq t(q_{2i}),$$

then an upper bound of the values of the positive roots of $p(x)$ *is given by*

$$ub = \max_{\{i=1,2,\ldots,m\}} \left\{ \left(\frac{b_{2i,1}}{c_{2i-1,1}} \right)^{\frac{1}{e_{2i-1,1} - e_{2i,1}}}, \ldots, \left(\frac{b_{2i,t(q_{2i})}}{c_{2i-1,t(q_{2i})}} \right)^{\frac{1}{e_{2i-1,t(q_{2i})} - e_{2i,t(q_{2i})}}} \right\}, \tag{3}$$

for any permutation of the positive coefficients $c_{2i-1,j}$, $j = 1, 2, \ldots, t(q_{2i-1})$. *Otherwise, for each of the indices* i *for which we have*

$$t(q_{2i-1}) < t(q_{2i}),$$

we ***break up*** *one of the coefficients of* $q_{2i-1}(x)$ *into* $t(q_{2i}) - t(q_{2i-1}) + 1$ *parts, so that now* $t(q_{2i}) = t(q_{2i-1})$ *and apply the same formula* (3) *given above.*

For a proof of this theorem see [11]. Among others, the following linear and quadratic complexity bounds on the values of the positive roots of polynomials are derived from it.

2.2 Linear complexity bounds derived from Theorem 2

Various linear complexity bounds can be obtained from the above theorem; the ones described below have been presented elsewhere, [11], but *not* in the context of complexity. We present them here again, briefly, for completeness:

C. Cauchy's "leading-coefficient" implementation of Theorem 2. For a polynomial $p(x)$, as in equation (1), with λ negative coefficients, Cauchy's method first breaks up its leading coefficient, α_n, into λ *equal* parts and then pairs each part with the first unmatched negative coefficient, [12].

That is, we have:

$$ub_C = \max_{\{1 \le k \le n:\ \alpha_{n-k} < 0\}} \sqrt[k]{-\frac{\lambda \alpha_{n-k}}{\alpha_n}}$$

or, equivalently,

$$ub_C = \max_{\{1 \le k \le n:\ \alpha_{n-k} < 0\}} \sqrt[k]{-\frac{\alpha_{n-k}}{\frac{\alpha_n}{\lambda}}}.$$

K. Kioustelidis' "leading-coefficient" implementation of Theorem 2. For a polynomial $p(x)$, as in (1), Kioustelidis' method matches the coefficient $-\alpha_{n-k}$ of the term $-\alpha_{n-k}x^{n-k}$ in $p(x)$ with $\frac{\alpha_n}{2^k}$, the leading coefficient divided by 2^k. Kioustelidis' "leading-coefficient" implementation of Theorem 2 differs from that of Cauchy's only in that the leading coefficient is now broken up in *unequal* parts, by dividing it with different powers of 2, [6],

$$ub_K = 2 \max_{\{1 \le k \le n:\ \alpha_{n-k} < 0\}} \sqrt[k]{-\frac{\alpha_{n-k}}{\alpha_n}}$$

or, equivalently,

$$ub_K = \max_{\{1 \le k \le n:\ \alpha_{n-k} < 0\}} \sqrt[k]{-\frac{\alpha_{n-k}}{\frac{\alpha_n}{2^k}}}.$$

In addition to the above we present our own two linear complexity implementations of Theorem 2, the combination of which (that is, taking the minimum) yields the best linear

complexity upper bound on the values of the positive roots of a polynomial, [11]. Please note that in general we can obtain better bounds if in Theorem 2 we pair coefficients from non adjacent polynomials $q_{2\ell-1}(x)$ and $q_{2i}(x)$, $1 \leq \ell < i$ — and this is done in the following definitions:

FL. "**First–λ**" implementation of Theorem 2. For a polynomial $p(x)$, as in (2), with λ negative coefficients we first take care of all cases for which $t(q_{2i}) > t(q_{2i-1})$, by breaking up the last coefficient $c_{2i-1,t(q_{2i})}$, of $q_{2i-1}(x)$, into $t(q_{2i}) - t(q_{2i-1}) + 1$ *equal* parts. We then pair each of the first λ positive coefficients of $p(x)$, encountered as we move in non-increasing order of exponents, with the first unmatched negative coefficient.

LM. "**Local-Max**" implementation of Theorem 2. For a polynomial $p(x)$, as in (1), the coefficient $-\alpha_k$ of the term $-\alpha_k x^k$ in $p(x)$ — as given in (1) — is paired with the coefficient $\frac{\alpha_m}{2^t}$, of the term $\alpha_m x^m$, where α_m is the largest positive coefficient with $n \geq m > k$ and t indicates the number of times the coefficient α_m has been used.

We have tested extensively — on various classes of specific and random polynomials — all four linear complexity bounds mentioned above and the following is a summary of our findings, [11]:

- Kioustelidis' bound is, in general, better (or much better) than Cauchy's; this happens because the former breaks up the leading coefficient in *unequal* parts, whereas the latter breaks it up in *equal* parts.

- Our *first-λ* bound, as the name indicates, uses additional coefficients and, therefore, it is not surprising that it is, in general, better (or much better) than both previous bounds. In the few cases where Kioustelidis' bound is better than *first-λ*, our *local-max* bound takes again the lead.

Therefore, given their linear cost of execution, $\min(FL, LM)$ is the best among the linear complexity bounds, [11], and we use it in Section 3 to determine the cutoff point between linear and quadratic complexity bounds. Additionally, in Section 3 we see that implementing $\min(FL, LM)$ in the continued fractions real root isolation method, *VAS-CF*/ $\min(FL, LM)$, we obtain a speed-up of 15 % — when compared with $VAS - CF/Cauchy$.

2.3 Quadratic complexity bounds derived from Theorem 2

In this subsection we present the two quadratic complexity bounds which will be used in our experimentation. See the literature, [13], for a more complete discussion of quadratic complexity bounds.

We begin with *KQ*, the quadratic version of Kioustelidis' bound on the values of the positive roots of a polynomial. To our knowledge this was first presented by Hong, [14, p. 574], and can be stated as follows:

KQ. Kioustelidis' Quadratic complexity implementation of Theorem 2. For a polynomial $p(x)$, as in equation (1), each negative coefficient $a_i < 0$ is "paired" with *each* one of the preceding positive coefficients a_j divided by 2^{j-i} — that is, *each* positive coefficient a_j is "broken up" into unequal parts, as is done with *just* the leading coefficient in Kioustelidis' bound — and the minimum is taken over all j; subsequently, the maximum is taken over all i.

That is, we have:

$$ub_{KQ} = 2 \max_{\{a_i<0\}} \min_{\{a_j>0:\ j>i\}} \sqrt[j-i]{-\frac{a_i}{a_j}},$$

or, equivalently,

$$ub_{KQ} = \max_{\{a_i<0\}} \min_{\{a_j>0:\ j>i\}} \sqrt[j-i]{-\frac{a_i}{\frac{a_j}{2^{j-i}}}}.$$

Hong's Theorem 2.2, if examined carefully, reveals that for polynomials in one variable it is the quadratic version of Kioustelidis' bound, *KQ*; moreover, the relation with our Theorem 2 is quite obvious.

This observation lead us to our own quadratic complexity bound, *LMQ*, based also on our Theorem 2, [13], which can be stated as follows:

LMQ. "Local-Max" Quadratic complexity implementation of Theorem 2. For a polynomial $p(x)$, as in (1), each negative coefficient $a_i < 0$ is "paired" with *each* one of the preceding positive coefficients a_j divided by 2^{t_j} — that is, *each* positive coefficient a_j is "broken up" into unequal parts, as is done with *just* the locally maximum coefficient in the local max bound; t_j is initially set to 1 and is incremented each time the positive coefficient a_j is used — and the minimum is taken over all j; subsequently, the maximum is taken over all i

That is, we have:

$$ub_{LMQ} = \max_{\{a_i<0\}} \min_{\{a_j>0:\ j>i\}} \sqrt[j-i]{-\frac{a_i}{\frac{a_j}{2^{t_j}}}}.$$

Since $2^{t_j} \leq 2^{j-i}$ — where i and j are the indices realizing the max of min; equality holds when there are *no* missing terms in the polynomial — it is clear that the bounds computed by *LMQ* are sharper by the factor $2^{\frac{j-i-t_j}{j-i}}$ than those computed by Kioustelidis' *KQ*; nonetheless, *VAS-CF* is implemented with both quadratic bounds.

We have also implemented the fastest quadratic complexity bound, FLQ, but we omit it since its behavior is similar to that of *LMQ* and it cannot be directly compared with *KQ*; details can be found elsewhere, [15].

3 Empirical results

In this section we compare four implementations of the *VAS-CF* real root isolation method using two linear and two quadratic complexity bounds on the values of the positive roots of polynomials.

The two linear complexity bounds are: Cauchy's and $\min(FL, LM)$, the minimum of our *F*irst *L*ambda and *L*ocal *M*ax bounds, [11], whereas the two quadratic complexity ones are: *KQ*, the *Q*uadratic complexity variant of *K*ioustelidis' bound, studied by Hong, [14], and *LMQ*, the *Q*uadratic complexity version of our *L*ocal *M*ax bound, [13].

Our choice of the various bounds in the implementations of *VAS-CF* is justified as follows:

1. From the linear complexity bounds we included:
   (a) Cauchy's bound, C, to be used as a point of reference, since it has been in use for the past 30 years, and
   (b) our $\min(FL, LM)$ bound, [11], which is the best among the linear complexity bounds, in order to see when it's implementation will outperform that of the two quadratic complexity bounds.
2. From the quadratic complexity bounds we included:
   (a) Kioustelidis' *KQ*, and
   (b) our own *LMQ*, in order to compare their performance; as explained in the previous section *LMQ* computes sharper estimates than *KQ*.

We have implemented all versions of *VAS-CF* as a part of *Mathematica* kernel. They all use the same implementation of Shaw and Traub's algorithm for Taylor shifts (see [16]). We followed the standard practice and used as benchmark the Laguerre[3], Chebyshev (first[4] and second[5] kind), Wilkinson[6] and Mignotte[7] polynomials, as well as several types of randomly generated polynomials of degrees $\{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000\}$. For the random polynomials the size of the coefficients ranges from -2^{20} to 2^{20}.

All computations were done on a Windows XP laptop computer with with 1.8 Ghz Pentium M processor, and 2 GB of RAM. The change in memory use was negligible in every case, and hence, it is not included in the following tables. The average speed-up was calculated using the formula: *Speed-up* $= 100 \cdot (\textit{LMQ} - \textit{Cauchy})/\textit{Cauchy}$, from which we omitted the minus sign.

[3] recursively defined as: $L_0(x) = 1$, $L_1(x) = 1 - x$, and $L_{n+1}(x) = \frac{1}{n+1}((2n + 1 - x)L_n(x) - nL_{n-1}(x))$

[4] recursively defined as: $T_0(x) = 1$, $T_1(x) = x$, and $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$

[5] recursively defined as: $U_0(x) = 1$, $U_1(x) = 2x$, and $U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)$

[6] defined as: $W(x) = \prod_{i=1}^{n}(x - i)$

[7] defined as: $M_n(x) = x^n - 2(5x - 1)^2$

Table 1. Special Polynomials. Some indicative results are given for degrees 100, 1000, 1500 and 2000. The average speed-up for this table is about 32 %

Polynomial Class	Degree	Time (s)			
		Cauchy	*FL + LM*	*KQ (Hong)*	*LMQ*
Laguerre	100	0.23	0.19	0.19	0.17
Laguerre	1000	979	665	729	633
Laguerre	1500	7194	4903	5356	4569
Laguerre	2000	27602	21007	22712	19277
Chebyshev I	100	0.19	0.17	0.16	0.11
Chebyshev I	1000	517	460	496	299
Chebyshev I	1500	3681	3333	3381	2188
Chebyshev I	2000	16697	15010	14571	10473
Chebyshev II	100	0.42	0.17	0.15	0.10
Chebyshev II	1000	529	437	443	296
Chebyshev II	1500	3772	3198	3190	2166
Chebyshev II	2000	16559	14492	14370	10184
Wilkinson	100	0.03	0.03	0.03	0.03
Wilkinson	1000	54.6	44.5	43.7	43.3
Wilkinson	1500	339	295	270	265
Wilkinson	2000	1361	1305	1241	1242
Mignotte	100	0.008	0.004	0.008	0.004
Mignotte	1000	0.79	0.78	0.81	0.66
Mignotte	1500	2.05	2.12	2.06	1.77
Mignotte	2000	4.52	4.37	4.47	3.69

Table 2. Polynomials with random 10-bit coefficients. The average speed-up for this table is about 39 %

Degree	No. of roots Avg (Min/Max)	Time (s), Avg (Min/Max)			
		Cauchy	*FL + LM*	*KQ (Hong)*	*LMQ*
100	4.4 (2/6)	0.01 (0.00/0.01)	0.01 (0.01/0.02)	0.01 (0.01/0.02)	0.01 (0.01/0.01)
200	4.0 (2/8)	0.06 (0.02/0.18)	0.06 (0.03/0.16)	0.05 (0.03/0.14)	0.04 (0.03/0.09)
300	4.8 (4/6)	0.14 (0.07/0.24)	0.12 (0.06/0.22)	0.13 (0.07/0.19)	0.09 (0.07/0.13)
400	4.4 (4/6)	0.17 (0.12/0.21)	0.18 (0.12/0.25)	0.17 (0.12/0.20)	0.16 (0.12/0.20)
500	4.8 (2/8)	0.70 (0.21/1.96)	0.54 (0.20/1.22)	0.35 (0.21/0.56)	0.32 (0.20/0.50)
600	5.2 (4/6)	0.96 (0.46/1.41)	0.86 (0.51/1.25)	0.60 (0.42/0.84)	0.53 (0.42/0.72)

700	4.0 (2/6)	0.95 (0.45/1.68)	0.81 (0.44/1.33)	0.82 (0.44/1.25)	0.69 (0.50/0.91)
800	5.2 (4/8)	1.97 (0.67/4.09)	1.68 (0.74/3.33)	1.22 (0.71/2.25)	1.02 (0.72/1.70)
900	3.6 (2/6)	2.56 (0.68/7.15)	2.27 (0.72/6.13)	1.44 (0.71/2.55)	1.19 (0.67/1.87)
1000	6.4 (4/8)	4.07 (1.63/9.02)	3.56 (1.54/7.64)	2.86 (1.57/4.51)	2.06 (1.38/3.18)
1500	4.0 (2/6)	10.6 (2.73/26.1)	7.51 (2.33/13.9)	5.78 (2.35/10.1)	5.24 (2.43/7.77)
2000	6.8 (4/12)	53.8 (7.54/137)	45.5 (7.90/118)	23.3 (7.67/53.9)	19.1 (7.61/40.2)

Table 3. Polynomials with random 1000-bit coefficients. The average speed-up for this table is about 48 %

Degree	No. of roots	Time (s), $\mathrm{Avg}\,(\mathrm{Min}/\mathrm{Max})$			
	$\mathrm{Avg}\,(\mathrm{Min}/\mathrm{Max})$	*Cauchy*	*FL + LM*	*KQ (Hong)*	*LMQ*
100	4.0 (4/4)	0.01 (0.00/0.02)	0.01 (0.00/0.02)	0.01 (0.00/0.02)	0.01 (0.00/0.02)
200	3.6 (2/6)	0.06) (0.03/0.12)	0.05 (0.02/0.10)	0.04 (0.02/0.06)	0.03 (0.01/0.06)
300	4.8 (2/8)	0.12 (0.04/0.32)	0.11 (0.04/0.28)	0.10 (0.04/0.23)	0.09 (0.04/0.17)
400	4.4 (2/6)	0.29 (0.06/0.54)	0.25 (0.06/0.44)	0.24 (0.06/0.44)	0.16 (0.06/0.25)
500	5.2 (4/8)	0.68 (0.16/1.20)	0.55 (0.17/0.95)	0.45 (0.21/0.92)	0.32 (0.21/0.48)
600	3.6 (2/4)	0.76 (0.19/2.09)	0.54 (0.18/0.96)	0.43 (0.19/0.66)	0.39 (0.18/0.52)
700	3.6 (0/6)	1.26 (0.25/2.82)	1.28 (0.19/2.51)	0.85 (0.19/1.54)	0.68 (0.19/1.29)
800	4.4(2/6)	3.03 (0.29/5.50)	2.53 (0.26/4.76)	1.08 (0.34/1.68)	0.93 (0.27/1.53)
900	5.6 (4/8)	4.55 (1.05/9.32)	3.72 (1.02/7.53)	2.23 (1.00/3.09)	1.59 (0.76/2.68)
1000	3.6 (2/6)	2.42 (0.46/4.62)	2.06 (0.44/3.92)	1.27 (0.40/2.00)	1.04 (0.42/1.68)
1500	5.6 (4/8)	16.1 (2.30/40.2)	9.41 (1.99/18.2)	7.17 (2.10/11.9)	5.63 (1.96/10.8)
2000	5.2(4/6)	23.3 (4.12/79.8)	19.4 (4.08/65.4)	13.2 (4.33/33.4)	10.4 (4.11/20.2)

Table 4. Monic polynomials with random 10-bit coefficients. The average speed-up for this table is about 31 %

Degree	No. of roots Avg (Min/Max)	Time (s), Avg (Min/Max)			
		Cauchy	*FL + LM*	*KQ (Hong)*	*LMQ*
100	4.8 (2/8)	0.01 (0.01/0.02)	0.01 (0.00/0.02)	0.01 (0.01/0.02)	0.01 (0.00/0.02)
200	5.6 (4/6)	0.08 (0.03/0.16)	0.06 (0.03/0.13)	0.06 (0.03/0.09)	0.05 (0.03/0.08)
300	4.8(4/6)	0.12 (0.08/0.22)	0.12 (0.08/0.21)	0.12 (0.08/0.15)	0.10 (0.08/0.15)
400	4.8 (4/6)	0.19 (0.19/0.29)	0.19 (0.16/0.26)	0.18 (0.15/0.26)	0.17 (0.16/0.20)
500	5.2 (4/10)	0.44 (0.18/1.38)	0.42 (0.18/1.19)	0.33 (0.18/0.74)	0.32 (0.19/0.63)
600	5.6 (4/8)	0.99 (0.30/2.04)	0.76 (0.30/1.21)	0.65 (0.31/0.94)	0.49 (0.30/0.72)
700	5.2 (4/8)	1.14 (0.43/1.63)	0.99 (0.42/1.47)	0.92 (0.46/1.30)	0.73 (0.49/0.94)
800	5.6(4/8)	1.45 (0.66/1.99)	1.29 (0.64/1.62)	1.22 (0.65/1.42)	0.90 (0.69/1.03)
900	4.4 (2/6)	1.01 (0.74/1.18)	1.01 (0.71/1.31)	1.05 (0.69/1.36)	1.09 (0.72/1.33)
1000	5.6 (4/8)	3.40 (1.18/7.09)	3.02 (1.03/5.94)	(1.10/4.28) (1.10/4.28)	(1.10/2.70) (1.10/2.70)
1500	6.8 (6/8)	14.8 (6.06/27.3)	11.8 (5.86/17.1)	8.43 (5.98/12.9)	6.80 (4.90/9.24)
2000	7.6(4/14)	54.8 (6.12/137)	47.6 (6.09/120)	23.9 (6.15/56.0)	19.4 (6.03/42.2)

Table 5. Monic polynomials with random 1000-bit coefficients. The average speed-up for this table is about 60 %

Degree	No. of roots Avg (Min/Max)	Time (s), Avg (Min/Max)			
		Cauchy	*FL + LM*	*KQ (Hong)*	*LMQ*
100	6.0 (4/8)	0.03 (0.02/0.04)	0.01 (0.01/0.03)	0.01 (0.00/0.02)	0.01 (0.00/0.02)
200	5.2 (4/8)	0.09 (0.02/0.22)	0.07 (0.02/0.19)	0.04 (0.02/0.11)	0.04 (0.03/0.06)
300	5.6(4/8)	0.19 (0.06/0.46)	0.14 (0.07/0.28)	0.12 (0.06/0.24)	0.14 (0.07/0.26)
400	5.2 (4/8)	0.41 (0.08/1.00)	0.24 (0.06/0.54)	0.21 (0.06/0.44)	0.15 (0.06/0.28)
500	5.6 (4/8)	0.62 (0.18/1.00)	0.39 (0.12/0.68)	0.45 (0.12/0.74)	0.26 (0.12/0.37)

600	4.8 (4/6)	1.03 (0.24/3.09)	0.52 (0.17/1.21)	0.37 (0.17/0.68)	0.32 (0.17/0.59)
700	5.2 (2/10)	1.27 (0.20/2.67)	1.02 (0.21/1.84)	0.86 (0.19/1.43)	0.65 (0.19/1.09)
800	5.6(4/8)	2.92 (0.43/5.41)	2.40 (0.39/4.46)	1.02 (0.38/2.02)	0.79 (0.38/1.38)
900	6.0 (4/8)	4.22 (0.84/9.86)	2.67 (0.80/5.78)	1.84 (0.88/2.47)	1.43 (0.74/2.22)
1000	5.6 (4/6)	4.23 (2.21/5.86)	2.90 (1.34/4.21)	2.23 (1.34/3.52)	1.44 (1.15/1.84)
1500	6.8 (6/8)	17.1 (26.06/41.8)	11.4 (5.25/28.2)	8.28 (4.86/15.7)	5.44 (3.30/10.4)
2000	6.4(6/8)	30.6 (4.80/102)	24.0 (4.59/80.9)	16.7 (4.60/47.4)	12.6 (5.02/35.9)

Table 6. Products of terms $x^{20} - r$, with random 20-bit r. The average speed-up for this table is about 48 %

Degree	No. of roots	Time (s), $\mathrm{Avg}\,(\mathrm{Min}/\mathrm{Max})$			
	$\mathrm{Avg}\,(\mathrm{Min}/\mathrm{Max})$	*Cauchy*	*FL + LM*	*KQ (Hong)*	*LMQ*
100	10	0.05 (0.02/0.09)	0.05 (0.02/0.09)	0.03 (0.02/0.04)	0.02 (0.02/0.02)
200	20	0.31 (0.18/0.39)	0.27 (0.16/0.38)	0.24 (0.16/0.32)	0.15 (0.12/0.20)
300	30	1.07 (0.58/1.37)	0.89 (0.60/1.11)	0.87 (0.60/1.04)	0.57 (0.56/0.60)
400	40	2.22 (1.92/2.58)	1.97 (1.86/2.27)	1.94 (1.86/2.08)	1.50 (1.35/1.70)
500	50	8.51 (6.03/11.5)	5.32 (4.24/7.28)	4.55 (4.25/5.32)	3.24 (2.87/3.74)
600	60	13.9 (11.7/17.0)	9.15 (8.28/10.2)	8.96 (8.43/9.35)	6.43 (5.96/6.84)
700	70	24.6 (21.7/29.1)	17.2 (13.7/21.2)	16.5 (13.3/19.7)	12.1 (10.6/14.0)
800	80	38.0 (33.7/44.2)	26.3 (23.6/30.4)	24.4 (19.4/30.3)	17.2 (15.1/19.3)
900	90	53.7 (40.4/63.8)	43.5 (37.0/51.5)	37.0 (28.8/45.8)	29.5 (23.1/36.6)
1000	100	89.6 (70.9/103)	69.1 (52.2/78.5)	63.9 (45.4/76.5)	50.0 (42.1/58.9)
1500	150	577 (468/696)	456 (378/533)	429 (360/473)	353 (3.11/402)
2000	200	2228 (1917/2711)	1907 (1674/2342)	1808 (1614/2279)	1464 (1204/1767)

Table 7. Products of terms $x^{20} - r$, with random 1000-bit r. The average speed-up for this table is about 25 %

Degree	No. of roots Avg (Min/Max)	Time (s), Avg (Min/Max)			
		Cauchy	*FL + LM*	*KQ (Hong)*	*LMQ*
100	10	0.08 (0.05/0.10)	0.08 (0.05/0.12)	0.11 (0.06/0.31)	0.09 (0.03/0.23)
200	20	1.65 (0.96/2.14)	1.42 (0.97/2.09)	1.28 (1.02/0.1.45)	1.31 (1.10/1.50)
300	30	7.54 (5.08/10.8)	5.20 (4.46/5.65)	4.88 (3.67/5.49)	4.24 (3.92/4.69)
400	40	15.7 (10.8/19.7)	15.7 (13.3/17.5)	14.7 (12.7/17.3)	12.7 (11.0/14.1)
500	50	42.4 (29.2/64.7)	44.5 (35.2/48.7)	35.5 (32.8/40.5)	35.0 (27.5/49.7)
600	60	117 (91.9/154)	106 (82.6/134)	103 (90.0/121)	92.0 (86.5/97.0)
700	70	248 (208/332)	252 (221/282)	240 (205/264)	189 (168/205)
800	80	549 (351/753)	481 (410/590)	474 (412/542)	382 (364/432)
900	90	1138 (971/1271)	855 (721/967)	834 (718/931)	670 (646/723)
1000	100	1661 (1513/1913)	1335 (1123/1673)	1265 (1066/1440)	1065 (947/1146)
1500	150	9004 (8233/9705)	8360 (7281/8999)	8230 (7357/9652)	6141 (5659/6470)

Fig. 8. Products of terms $x - r$ with random integer r. The average speed-up for this table is about 35 %

Degree	No. of roots Avg (Min/Max)	Time (s), Avg (Min/Max)			
		Cauchy	*FL + LM*	*KQ (Hong)*	*LMQ*
10	100	0.46 (0.28/0.94)	0.24 (0.18/0.28)	0.34 (0.27/0.41)	0.34 (0.30/0.41)
10	200	1.46 (1.24/1.85)	1.40 (1.28/1.69)	1.41 (1.26/1.71)	1.40 (1.20/1.69)
10	500	18.1 (16.5/18.9)	18.1 (16.6/18.8)	21.2 (17.5/24.4)	22.1 (18.7/24.2)
1000	20	0.07 (0.04/0.14)	0.02 (0.02/0.03)	0.03 (0.02/0.04)	0.03 (0.02/0.04)
1000	50	3.69 (2.38/6.26)	0.81 (0.60/1.28)	0.88 (0.52/1.28)	0.81 (0.52/1.11)
1000	100	47.8 (37.6/56.9)	13.8 (10.3/19.2)	17.6 (12.4/25.9)	15.8 (11.3/21.3)

4 Conclusions

Notice the following general remarks:

- the only inequality between bounds that is always satisfied is that the *KQ* bound is always greater than or equal to *LMQ*,
- for an individual polynomial a better bound may lead to worse performance in *VAS-CF*,
- timing results are subject to measurement error, which especially affects small timings.

Taking these observations into consideration, one could foresee that using *LMQ* should give the best performance of *VAS-CF*, except for individual data points for which this is not true.

Indeed, from the tables in Section 3, it is obvious that the *VAS-CF* implementation using our *LMQ* bound is fastest for all classes of polynomials tested, except for the case of very many very large roots, Table 8. For all cases, the average overall improvement in computation time is about $40\,\%$. In the case of very many very large roots *VAS-CF* using *LMQ* is a very close second to *VAS-CF* using our linear complexity bound $\min(FL, LM)$.

Moreover, as was shown elsewhere, [17], using just our $\min(FL, LM)$ bound the *VAS-CF* algorithm is *always* faster than that by Vincent-Collins-Akritas[8], [19], or any of its variants, [20]. Therefore, our current results widen the gap between these two methods.

References

1. A. J. H. Vincent, Sur la résolution des équations numériques, *J. Math. Pure Appl.*, **1**, pp. 341–372, 1936.
2. A. Alesina, M. Galuzzi, A new proof of Vincent's theorem, *L'Enseignement Mathemathique*, **44**, pp. 219–256, 1998.
3. A. G. Akritas, *Elements of Computer Algebra with Applications*, John Wiley Interscience, New York, 1989.
4. A. G. Akritas, A. W. Strzebonski, A comparative study of two real root isolation methods, *Nonlinear Anal. Model. Control*, **10**(4), pp. 297–304, 2005.
5. E. P. Tsigaridas, I. Z. Emiris, Univariate polynomial real root isolation: Continued fractions revisited, in: *ESA 2006*, Y. Azar, T. Erlebach (Eds.), LNCS 4168, pp. 817–828, 2006.
6. B. Kioustelidis, Bounds for positive roots of polynomials, *J. Comput. Appl. Math.*, **16**(2), pp. 241–244, 1986.
7. V. Sharma, Complexity of Real Root Isolation Using Continued Fractions, in: *Proc. Annual ACM ISSAC, Waterloo, Canada*, C. W. Brown (Ed.), pp. 339–346, 2007.

[8] misleadingly referred to (by several authors) first as "modified Uspensky's method" and recently as "Descartes' method"; see [18]

8. V. Sharma, *Complexity Analysis of Algorithms in Algebraic Computation*, Ph.D. Thesis, Department of Computer Sciences, Courant Institute of Mathematical Sciences, New York University, 2007.

9. D. Ştefănescu, New bounds for positive roots of polynomials, *J. Univers. Comput. Sci.*, **11**(12), pp. 2132–2141, 2005.

10. A. G. Akritas, P. S. Vigklas, A comparison of various methods for computing bounds for positive roots of polynomials, *J. Univers. Comput. Sci.*, **13**(4), pp. 455–467, 2007.

11. A. G. Akritas, A. W. Strzebonski, P. S. Vigklas, Implementations of a new theorem for computing bounds for positive roots of polynomials, *Computing*, **78**, pp. 355–367, 2006.

12. N. Obreschkoff, *Verteilung und Berechnung der Nullstellen reeller Polynome*, VEB Deutscher Verlag der Wissenschaften, Berlin, 1963.

13. A. G. Akritas, A. W. Strzebonski, P. S. Vigklas, Quadratic complexity bounds on the values of the positive roots of polynomials (submitted).

14. H. Hong Bounds for absolute positiveness of multivariate polynomials, *J. Symb. Comput.*, **25**(5), pp. 571–585, 1998.

15. A. G. Akritas, A. I. Argyris, A. W. Strzeboński, FLQ, the Fastest Quadratic Complexity Bound on the Values of Positive Roots of Polynomials (to appear).

16. J. von zur Gathen, J. Gerhard, Fast algorithms for Taylor shifts and certain difference equations, in: *Proceedings of ISSAC'97, Maui, Hawaii, U.S.A.*, pp. 40–47, 1997.

17. A. G. Akritas, A. W. Strzebonski, P. S. Vigklas, Advances on the continued fractions method using better estimations of positive root bounds, in: *Proceedings of the 10th International Workshop on Computer Algebra in Scientific Computing, CASC 2007, Bonn, Germany, September 16–20, 2007*, V. G. Ganzha, E. W. Mayr, E. V. Vorozhtsov (Eds.), LNCS 4770, Springer Verlag, Berlin, pp. 24–30, 2007.

18. F. Boulier, *Systèmes polynomiaux : que signifie "résoudre"?*, Université Lille 1, UE INFO 251, 2007.

19. G. E. Collins, A. G. Akritas, Polynomial real root isolation using Descartes' rule of signs, in: *Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computations*, Yorktown Heights, N.Y., pp. 272–275, 1976.

20. F. Rouillier, P. Zimmermann, Efficient isolation of polynomial's real roots, *J. Comput. Appl. Math.*, **162**, pp. 33–50, 2004.