Serdica J. Computing **2** (2008), 145–162

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

FLQ, THE FASTEST QUADRATIC COMPLEXITY BOUND ON THE VALUES OF POSITIVE ROOTS OF POLYNOMIALS

Alkiviadis G. Akritas, Andreas I. Argyris, Adam W. Strzeboński

On the 20th Anniversary of the University of Thessaly

Abstract. In this paper we present FLQ, a *quadratic* complexity bound on the values of the positive roots of polynomials. This bound is an extension of $FirstLambda$, the corresponding *linear* complexity bound and, consequently, it is derived from Theorem 3 below. We have implemented FLQ in the Vincent-Akritas-Strzeboński Continued Fractions method (*VAS-CF*) for the isolation of real roots of polynomials and compared its behavior with that of the theoretically proven best bound, LMQ. Experimental results indicate that whereas FLQ runs on average faster (or quite faster) than LMQ, nonetheless the quality of the bounds computed by both is about the same; moreover, it was revealed that when *VAS-CF* is run on our benchmark polynomials using FLQ, LMQ and $min(FLQ, LMQ)$ all three versions run equally well and, hence, it is inconclusive which one should be used in the *VAS-CF* method.

ACM Computing Classification System (1998): G.1.5, F.2.1, I.1.2.

Key words: Vincent's theorem, real root isolation methods, linear and qua-dratic complexity bounds on the values of the positive roots.

1. Introduction. Computing an upper bound, ub, on the values of the (real) positive roots of a polynomial $p(x)$ is a very important operation because it can be used to isolate these roots — that is, to find intervals on the positive axis each containing exactly one positive root.

As an example, suppose that the positive roots of $p(x)$ lie in the open interval $]0, ub[$ and that we have a test for determining the number of roots in any interval $]a, b[$. Then, we can isolate these roots by repeatedly subdividing the interval $]0, ub[$ until each resulting interval contains exactly one root and every real root is contained in some interval. The bound, ub, is of practical use because we now work with a definite interval $]0, ub[$, instead of $]0, +\infty[$.

Obviously, the sharper the upper bound, ub, the more efficient the real root isolation method becomes, since fewer bisections will be performed. Please note that the bisection method uses the upper bound only once and imagine the savings in time that would occur if an isolation method depends heavily on repeated computations of such bounds!

Such is the case with the Vincent-Akritas-Strzeboński Continued Fractions (*VAS-CF*) method for isolating the positive roots of polynomial equations. This method is based on Vincent's theorem of 1836, [25], 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.

Note that if we represent by $\frac{ax+b}{cx+d}$ the continued fraction that leads to a transformed polynomial $f(x) = (cx+d)^n p\left(\frac{ax+b}{cx+d}\right)$, with one sign variation, then the single positive root of $f(x)$ — in the interval $]0, \infty[$ — corresponds to *that* positive root of $p(x)$ which is located in the open interval with endpoints

$\frac{b}{d}$ and $\frac{a}{c}$. These endpoints are *not* ordered and are obtained from $\frac{ax+b}{cx+d}$ by replacing x with 0 and ∞, respectively. See the literature [1], [2], Chapter 7 in [3], and the papers by Alesina & Galuzzi, [10], and [4] for a complete historical survey of the subject and implementation details respectively.

Therefore, with Vincent's theorem we can isolate the (positive) roots of a given polynomial $p(x)$. The negative roots are isolated — as suggested by Sturm — after we transform them to positive with the replacement $x \leftarrow -x$ performed on $p(x)$. The requirement that $p(x)$ have no multiple roots does not restrict the generality of the theorem because in the opposite case we first apply square-free factorization and then isolate the roots of each one of the square-free factors.

In 1978, [1], [2], it was found that each partial quotient α_i is the integer part of a real number — i.e. $\alpha_i = \lfloor \alpha_s \rfloor$, where α_s is the smallest positive root of some polynomial $f(x)$ — and, hence, that it can be computed as the lower bound, ℓb, on the values of the positive roots of a polynomial. So assuming that $\ell b = \lfloor \alpha_s \rfloor$ (*ideal* lower bound) we now set $\alpha_i \leftarrow \ell b$, $\ell b \geq 1$, and perform the replacement $x \leftarrow x + \ell b$, $\ell b \geq 1$ — which takes about the same time as the replacement $x \leftarrow x + 1$. Later, the assumption of the *ideal* lower bound was abandoned, [4].

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 $x^n f\left(\frac{1}{x}\right)$ and then setting $\ell b = \frac{1}{ub}$. So what is needed is an efficient method for computing upper bounds on the values of *just* the positive roots of polynomial equations[1].

It should be emphasized that bounds on the values of just the *positive* roots of polynomials are scarce in the literature. Cauchy's 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, [24], Emiris and Tsigaridas used Kioustelidis' bound, [16] and independently verified the results obtained by Akritas and Strzeboński[2], [4]. Please note that both bounds mentioned above, Cauchy's and Kioustelidis', are *linear* in complexity.

---

[1]With suitable transformations $p(x) \equiv p(-x) = 0$ and $p(x) \equiv x^n p\left(-\frac{1}{x}\right) = 0$ one can find the lower $-ub$ and upper $-\frac{1}{ub}$ bounds of the negative roots x_- of $p(x)$ respectively, $-ub \leq x_- \leq -\frac{1}{ub}$.

[2]See also Sharma's work, [20] and [21], where he used the worst possible positive lower bound to prove that the *VAS-CF* method is still polynomial in time!

Especially Kioustelidis' bound appeared in 1986, [16], but went rather unnoticed by Hong, [15], when he developed the *first* quadratic complexity bound on the values of the positive roots of polynomials.

In recent work, [9], [5], a theorem by Ştefănescu of 2005, [22], was extended and generalized in such a way that *all* the — then existing — linear complexity methods for computing bounds on the values of the positive roots of a polynomial are derived from it. Based on Theorem 3, FL and LM, two new linear complexity bounds were developed; and using their minimum in *VAS-CF* not only was the isolation of real roots speeded up 15% — when compared with the version of *VAS-CF* implementing Cauchy's bound, [7] — but it also became always faster than the Vincent-Collins-Akritas[3] bisection method (*VCA-Bisect*), [6].

Recently, motivated by Hong's work, [15], new quadratic complexity methods were developed for computing bounds on the values of the positive roots of polynomials. These methods — as well as the one developed by Hong — are also derived from Theorem 3 and are presented elsewhere [7]. It has been demonstrated that — except for FLQ — among the quadratic complexity bounds the estimate obtained by LMQ is always the best.

In section 2 we present Theorem 3 from which *all* methods for computing bounds on the values of the positive roots of a polynomial are derived. We then present the linear complexity bounds — First Lambda, (FL) and Local Max, (LM) — along with their corresponding quadratic complexity bounds — FLQ and LMQ. Please note that LMQ was first presented elsewhere, [7].

In section 3 we present the code for the quadratic complexity bounds FLQ and LMQ.

Finally, in section 4 we compare the estimates of FLQ and LMQ along with the time needed to compute them; moreover, we compare their performance in the *VAS-CF* real root isolation method.

2. Theoretical Background.

In the literature there are bounds on the *absolute* values of the roots, [13] , [17], [26], and bounds on just the *positive* roots of polynomials, [16], [18]. Although of limited use, the most recent addition to the latter type of bounds has been by Ştefănescu, [22]. He proved the following theorem:

Theorem 2 (Ştefănescu, 2005)**.** *Let $p(x) \in R[x]$ be such that the number of variations of signs of its coefficients is* ***even****. If*

$$p(x) = c_1x^{d_1} - b_1x^{m_1} + c_2x^{d_2} - b_2x^{m_2} + \ldots + c_kx^{d_k} - b_kx^{m_k} + g(x), \tag{1}$$

[3]Misleadingly referred to in the literature as "modified Uspenskys" or "Descartes" method

with $g(x) \in R_+[x], c_i > 0, b_i > 0, d_i > m_i > d_{i+1}$ *for all* i*, the number*

$$B_3(p) = \max\left\{ \left(\frac{b_1}{c_1}\right)^{1/(d_1-m_1)}, \ldots, \left(\frac{b_k}{c_k}\right)^{1/(d_k-m_k)} \right\} \tag{2}$$

is an upper bound for the positive roots of the polynomial p *for any* ***choice*** *of* $c_1, \ldots, c_k$.

Ştefănescu's theorem introduces the concept of *matching* or *pairing* a positive coefficient with the negative coefficient of a lower order term; however, Ştefănescu's theorem worked *only* for polynomials with an even number of sign variations.

Ştefănescu's theorem was generalized in the sense that Theorem 3 below applies to polynomials with any number of sign variations, [9]. To accomplish this, the concept of *breaking up* a positive coefficient was introduced, whereby each of the several parts of the coefficient is paired with negative coefficients of lower order terms[4], [5].

Theorem 3. *Let* $p(x)$

$$p(x) = \alpha_n x^n + \alpha_{n-1}x^{n-1} + \ldots + \alpha_0, \quad (\alpha_n > 0) \tag{3}$$

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*

$$p(x) = q_1(x) - q_2(x) + q_3(x) - q_4(x) + \ldots + q_{2m-1}(x) - q_{2m}(x) + g(x), \tag{4}$$

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*

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

[4]After our work, [5], Ştefănescu also extended his Theorem 2, [23].

then an upper bound of the values of the positive roots of $p(x)$ is given by
(5)

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

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 (5) given above.*

For a proof of this theorem see [5]. Please note that the partial extension of Theorem 2 presented in [9] does not treat the case $t(q_{2i-1}) < t(q_{2i})$.

Crucial Observation. Pairing up positive with negative coefficients and breaking up a positive coefficient into the required number of parts — to match the corresponding number of negative coefficients — are the key ideas of this theorem. In general, formulae analogous to (5) hold for the cases where: (a) we pair coefficients from the non-adjacent polynomials $q_{2l-1}(x)$ and $q_{2i}(x)$, for $1 \le l < i$, and (b) we break up one or more positive coefficients into several parts to be paired with the negative coefficients of lower order terms.

Among others, the following linear and quadratic complexity bounds on the values of the positive roots of polynomials are derived from Theorem 3.

2.1. Two Linear Complexity Bounds Derived from Theorem 3. Various linear complexity bounds can be obtained from Theorem 3; the ones described below have been presented elsewhere, [5], but *not* in the context of complexity. We present them here again, briefly, for completeness:

FL. "**first–λ**" implementation of Theorem 3. For a polynomial $p(x)$, as in (4), 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 3. For a polynomial $p(x)$, as in (3), the coefficient $-\alpha_k$ of the term $-\alpha_k x^k$ in $p(x)$ — as given in Eq. (3) — 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 $n \ge m > k$ and t indicates the number of times the coefficient α_m has been used.

These two bounds have been extensively tested — on various classes of specific and random polynomials — and it was found that their combination, $\min(FL, LM)$, is the best among the linear complexity bounds, [5]; moreover, a speed-up of 15% was achieved with *VAS-CF*/$\min(FL, LM)$, that is, the continued fractions real root isolation method using the bound $\min(FL, LM)$ — when compared with *VAS-CF/Cauchy*, the continued fractions method implementing *Cauchy*'s bound, [7].

2.2. Two Quadratic Complexity Bounds Derived from Theorem 3. In this subsection we present FLQ and LMQ the two quadratic complexity implementations of FL and LM respectively, which are also derived from Theorem 3; other quadratic complexity bounds are described elsewhere [7]. In general, the estimates obtained from the quadratic complexity bounds are better than those obtained from their linear complexity counterparts, as they are computed after much greater effort.

FLQ. "First-Lambda" Quadratic complexity implementation of Theorem 3. For a polynomial $p(x)$, as in (4), 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 $d_{2i-1,t(q_{2i})} = t(q_{2i}) - t(q_{2i-1}) + 1$ *equal* parts. Then each negative coefficient $a_\mu < 0$ is "paired" with *each* one of the preceding $min(\mu, \lambda)$ positive coefficients a_ν divided by d_ν — that is, *each* of the preceding $min(\mu, \lambda)$ positive coefficient a_ν is "broken up" into d_ν *equal* parts, where d_ν is initially set to 1 and its value changes *only* if the positive coefficient a_ν is broken up into equal parts, as stated in Theorem **??**; $u(\nu)$ indicates the number of times a_ν can be used to calculate the minimum, it is originally set equal to d_ν and its value decreases each time a_ν is used in the computation of the minimum — and the minimum is taken over all ν; subsequently, the maximum is taken over all μ.

That is, we have:

$$ub_{FLQ} = \max_{\{a_\mu<0\}} \min_{\{a_\nu>0:\nu>min(\mu,\lambda):u(\nu)\neq 0\}} \sqrt[\nu-\mu]{-\frac{a_\mu}{\frac{a_\nu}{d_\nu}}}.$$

LMQ. "Local-Max"" Quadratic complexity implementation of Theorem 3. For a polynomial $p(x)$, as in (3), each negative coefficient $a_\mu < 0$ is "paired" with *each* one of the preceding positive coefficients a_ν divided by 2^{t_ν} — that is, *each* positive coefficient a_ν is "broken up" into *unequal* parts, as is done with *just* the locally maximum coefficient in the local max bound; t_ν is initially set to 1 and is incremented each time the positive coefficient a_ν is used — and the minimum is taken over all ν; subsequently, the maximum is taken over all μ.

That is, we have:

$$ub_{LMQ} = \max_{\{a_\mu<0\}} \min_{\{a_\nu>0:\nu>\mu\}} \sqrt[\nu-\mu]{-\frac{a_\mu}{\frac{a_\nu}{2^{t_\nu}}}}.$$

From the above two descriptions it is clear that FLQ tests *just* the first $\min(\mu,\lambda)$ positive coefficients, whereas LMQ tests *all* the preceding positive coefficients. Hence, FLQ is faster (or quite faster) than LMQ. In addition, since the other quadratic complexity bounds described in [7] work as the LMQ bound, it is obvious that FLQ is the fastest quadratic complexity bound.

3. Algorithmic Implementation of *FLQ* and *LMQ*.

In this section we present the code for LMQ and FLQ, the latter in two parts.

Input : A univariate polynomial $p(x) = a_k x^k + a_{k-1}x^{k-1} + \ldots + a_0, (a_k > 0)$
Output: An upper bound $tempmax$, on the values of the positive roots of the polynomial

initializations;
$cl \longleftarrow \{a_0, a_1, a_2, \ldots, a_{k-1}, a_k\}$;
$timesused \longleftarrow \{1, 1, 1, \ldots, 1\}$;
$tempmax = 0$;
if $k + 1 \leq 1$ **then return** $tempmax = 0$;
for $m \longleftarrow k$ **to** 1 **do**
  if $cl(m) < 0$ **then**
    $tempmin = \infty$;
    for $n \longleftarrow k + 1$ **to** $m + 1$ **do**
      $temp = (\frac{-cl(m)}{\frac{cl(n)}{2^{timesused(n)}}})^{\frac{1}{n-m}}$;
      $timesused(n) + +$;
      if $tempmin > temp$ **then** $tempmin = temp$;
    end
    if $tempmax < tempmin$ **then** $tempmax = tempmin$;
  end
end
return $tempmax$;

Algorithm 3.1. LMQ implementation

Input : A univariate polynomial $p(x) = a_k x^k + a_{k-1}x^{k-1} + \ldots + a_0, (a_k > 0)$

Output: An upper bound $tempmax$, on the values of the positive roots of the polynomial

initializations;
$cl \longleftarrow \{a_0, a_1, a_2, \ldots, a_{k-1}, a_k\}$;
$\lambda \longleftarrow$ number of negative elements of cl;
$usedVector \longleftarrow \{0, 0, 0, \ldots, 0\}$;
for $i \longleftarrow 1$ **to** $k+1$ **do**
    if $cl(i) > 0$ **then** $usedVector(i) = 1$;
end
if $k+1 \leq 1$ **or** $\lambda = 0$ **then return** $tempmax = 0$;
$i = k+1$;
$templamda = 0$;
$flag = 0$;
while $templamda < \lambda$ **do** `// make sure` $t(q_{2i-1}) \geq t(q_{2i})$ `holds for all` i
    if $cl(i) > 0$ **then**
        if $flag = 0$ **then** $posCounter++$;
        else if $flag = 1$ **then**
            if $negCounter > posCounter$ **then**
                $usedVector(positionLastPositiveCoef) = negCounter - posCounter + 1$;
            end
            $negCounter = 0$;
            $posCounter = 1$;
            $flag = 0$;
        end
        $positionLastPositiveCoef = i$;
    else if $cl(i) < 0$ **then**
        $flag = 1$;
        $negCounter++$;
        $templamda++$;
    end
    $i--$;
end
if $negCounter > posCounter$ **then**
    $usedVector(positionLastPositiveCoef) = negCounter - posCounter + 1$;
end

Algorithm 3.2. FLQ implementation part 1

```
sumPosCoeff = 0;
i = k + 1;
// Last of the first-λ coefficients
while sumPosCoeff < λ do
    if usedVector(i) ≠ 0 then
        sumPosCoeff+ = usedVector(i);
        flPos = i;
    end
    i − −;
end
/* If the last of the first-λ coefficients is a broken one
   (usedVector(flPos) > 1), there might be a chance that the
   sum of the positive coefficients (including broken ones) is
   more than λ.For Example:
   Let the signs of p be + + + - + + - - - - + + + -
   the 5^th positive coefficient will be broken into 2 pieces
   (usedVector(8) = 2). However, the sum of the first-λ (5 non
   broken)positive coefficients is 6(incl. broken). As a
   result we are going to use the last of the positive first-λ
   coefficients timesToUse(8) − (sum − λ) = 1 time only.     */
timesToUse(flpos)− = (sumPosCoeff − λ);
denomVector ←− usedVector;
m = k;
tempmax = 0;
while λ > 0 do
    if cl(m) < 0 then
        tempmin = ∞;
        for n = k + 1 to max(m + 1, flPos) do
            if usedVector(n) > 0 then
                tempB = ( −cl(m) / (cl(n) / denomVector(n)) )^(1/(n−m));
                if tempmin > tempB then
                    tempmin = tempB;
                    tempN = n;
                end
            end
        end
        usedVector(tempN) − −;
        λ − −;
        if tempmax < tempmin then tempmax = tempmin;
    end
    m − −;
end
return tempmax;
```

Algorithm 3.3. FLQ implementation part 2

4. Empirical Results. The experimental results presented in this section are divided in two groups: Tables 1–2 and Tables 3–7.

In Tables 1 and 2 we present the estimates computed by FLQ and LMQ for various classes of specific and random/custom polynomials. Moreover, the time needed for each estimate is recorded in parentheses. These computations were performed on a P4 Northwood 2.4GHz @ 2.7GHz, 1GB RAM computer. The following random/custom polynomials were used:

- **sRand**:
$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$
with random $\{a_n, a_{n-1}, \ldots, a_0\} \in [-2^{20}, 2^{20}]$ and $seed = 1001$.

- **usRand**:
$$p(x) = x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$
with random $\{a_{n-1}, \ldots, a_0\} \in [-2^{20}, 2^{20}]$ and $seed = 1001$.

- **pRand I**:
$$p(x) = \prod_{degree} (x - n)$$
with random $n \in [-2^{10}, 2^{10}]$ and $seed = 1001$.

- **pRand II**:
$$p(x) = \prod_{degree} (x - n)$$
with random $n \in [-2^{1000}, 2^{1000}]$ and $seed = 1001$.

- **Custom Poly I**:
$$p(x) = x^3 + (10^{100})x^2 - (10^{100})x - 1$$

- **Custom Poly II**:
$$p(x) = x^9 + 3x^8 + 2x^7 + x^6 - 4x^4 + x^3 - 4x^2 - 3$$

Table. 1. Bounds for positive roots of various types of polynomials. *MPR* stands for the maximum positive root, computed numerically.

		Degrees									
Polynomial	Bounds	10	100	200	300	400	500	600	700	800	900
Laguerre	LMQ	200	2×10^4	8×10^4	18×10^4	32×10^4	50×10^4	72×10^4	98×10^4	128×10^4	162×10^4
		(0.)	(0.563)	(3.562)	(11.187)	(25.594)	(49.782)	(87.344)	(142.453)	(220.766)	(329.719)
	FLQ	100	10^4	4×10^4	9×10^4	16×10^4	25×10^4	36×10^4	49×10^4	64×10^4	81×10^4
		(0.)	(0.015)	(0.031)	(0.078)	(0.109)	(0.172)	(0.25)	(0.328)	(0.406)	(0.5)
	MPR	29.92	374.98	767.82	1162.8	1558.81	1955.44	2352.5	2749.87	3147.48	3545.29
ChecyshevI	LMQ	2.23607	7.07107	10	12.2474	14.1421	15.8114	17.3205	18.7083	20	21.2132
		(0.)	(0.078)	(0.515)	(1.547)	(3.453)	(6.453)	(10.688)	(15.891)	(23.406)	(33.75)
	FLQ	1.58114	5	7.07107	8.66025	10	11.1803	12.2474	13.288	14.1421	15
		(0.)	(0.)	(0.015)	(0.047)	(0.62)	(0.11)	(0.141)	(0.172)	(0.234)	(0.281)
	MPR	0.987688	0.999877	0.999969	0.999989	0.999992	0.999995	0.999997	0.999997	0.999998	0.999998
ChecyshevII	LMQ	2.12132	7.03562	9.97497	12.227	14.1244	15.7956	17.3061	18.6949	19.9875	21.2014
		(0.015)	(0.079)	(0.531)	(1.563)	(3.453)	(6.468)	(10.719)	(16.687)	(24.656)	(34.734)
	FLQ	1.5	4.97494	7.05337	8.64581	9.98749	11.1692	12.2372	13.2193	14.1333	14.9917
		(0.)	(0.016)	(0.015)	(0.047)	(0.078)	(0.109)	(0.141)	(0.187)	(0.219)	(0.265)
	MPR	0.959493	0.999516	0.999878	0.999945	0.999969	0.99998	0.999986	0.99999	0.999992	0.999994
Wilkinson	LMQ	110	10100	40200	90300	160400	250500	360600	490700	640800	810900
		(0.)	(6.391)	(52.234)	(179.75)	(438.516)	(878.)	(1549.89)	(2508.52)	(3833.52)	(5569.42)
	FLQ	55	5050	20100	45150	80200	125250	180300	245350	320400	405450
		(0.)	(0.016)	(0.047)	(0.078)	(0.014)	(0.203)	(0.282)	(0.359)	(0.516)	(0.656)
	MPR	10	100	200	300	400	500	600	700	800	900
Mignotte	LMQ	1.77828	1.04811	1.02353	1.01557	1.01164	1.00929	1.00773	1.00662	1.00579	1.00514
		(0.)	(0.)	(0.015)	(0.)	(0.)	(0.016)	(0.)	(0.)	(0.)	(0.015)
	FLQ	1.63069	1.04073	1.01995	1.01321	1.00988	1.00789	1.00656	1.00562	1.00491	1.00437
		(0.)	(0.016)	(0.)	(0.016)	(0.15)	(0.016)	(0.016)	(0.015)	(0.016)	(0.015)
	MPR	1.5763	1.0362	1.0177	1.0117	1.0088	1.0070	1.0058	1.0050	1.0044	1.0039
sRand	LMQ	2.01011	14.3673	1.39904	3.54546	3.65744	7.75602	2.5257	1.53975	1.70317	1.65478
		(0.)	(0.219)	(1.016)	(2.11)	(3.828)	(6.)	(8.625)	(11.735)	(14.765)	(18.734)
	FLQ	2.45417	25.1062	1.04472	3.54546	2.5862	7.75602	2.03158	1.6409	1.48873	1.17328
		(0.)	(0.031)	(0.157)	(0.281)	(0.547)	(0.843)	(1.157)	(1.656)	(1.5)	(2.375)
	MPR	1.6173	8.15106	0.982276	1.1221	1.71921	1.012339	0.983633	0.983628	1.010844	0.983628
usRand	LMQ	1.57532	1916790	1.40849	1.78915	105264	1272940	1803160	12.6533	1197500	790432
		(0.)	(0.25)	(1.032)	(2.125)	(3.797)	(5.891)	(8.5)	(11.531)	(14.875)	(18.765)
	FLQ	3.16629	1916790	1.06293	15.6504	105264	1272940	901580	6.32665	598750	790432
		(0.016)	(0.047)	(0.156)	(0.266)	(0.531)	(0.828)	(1.156)	(1.641)	(2.046)	(2.328)
	MPR	0.925381	1.018871	0.982509	0.9841	1.026689	1.011621	1.331235	1.755769	1.013423	1.228396

Table. 2. cont. Bounds for positive roots of various types of polynomials. *MPR* stands for the maximum positive root, computed numerically.

		Degrees		
Polynomial	Bounds	100	200	500
pRand I	LMQ	5984.55	8818.63	14435.4
		(10.297)	(75.453)	(1053.55)
	FLQ	4231.72	5555.39	9093.75
		(0.562)	(1.453)	(21.156)
	MPR	998	998	1019
		20	50	100
pRand II	LMQ	$2.478575900498678 \times 10^{301}$	$4.381225845096125 \times 10^{301}$	4.62669×10^{301}
		(7.812)	(138.86)	(1359.34)
	FLQ	$1.925678288070229 \times 10^{301}$	$3.210297938962179 \times 10^{301}$	3.51814×10^{301}
		(1.844)	(12.578)	(169.594)
	MPR	0.99777×10^{301}	1.05601×10^{301}	1.05601×10^{301}
Custom Poly I	LMQ	2		
		(0.)		
	FLQ	1		
		(0.)		
	MPR	1		
Custom Poly II	LMQ	1.3218		
		(0.)		
	FLQ	1.1487		
		(0.)		
	MPR	1.06815		

Tables 3–7 show the time needed for the *VAS-CF* method to isolate the real roots of various classes of specific and random polynomials — described in the header of each table — when it uses the bounds FLQ, LMQ and $\min(FLQ, LMQ)$. These computations were done on Windows XP laptop computer with with 1.8 Ghz Pentium M processor, and 2 GB of RAM.

Table 3. Special Polynomials

Polynomial	Degree	No. of Roots	FLQ *T(s)*	LMQ *T(s)*	$FLQ + LMQ$ *T(s)*
Laguerre	100	100	0.191	0.19	0.18
Laguerre	500	500	34.69	36.192	33.589
Laguerre	1000	1000	610.047	703.772	662.152
Chebychev I	100	100	0.15	0.161	0.17
Chebychev I	500	500	29.773	27.79	26.528
Chebychev I	1000	1000	422.577	384.073	386.285
Chebychev II	100	100	0.14	0.171	0.16
Chebychev II	500	500	27.76	30.303	27.149
Chebychev II	1000	1000	408.628	381.839	382.57
Wilkinson	100	100	0.03	0.05	0.03
Wilkinson	500	500	4.717	4.777	4.817
Wilkinson	1000	1000	56.341	57.343	57.392
Mignotte	100	100	0.011	0.01	0.01
Mignotte	500	500	0.24	0.19	0.17
Mignotte	1000	1000	1.022	0.811	0.821

Table 4. Polynomial with randomly generated coefficients

Coefficients (bit length)	Degree	No. of Roots (average)	FLQ *T(s)*	LMQ *T(s)*	$FLQ + LMQ$ *T(s)*
10	500	5.2	0.5628	0.4228	0.4244
10	1000	6.4	3.7152	2.4256	2.3254
10	2000	4	41.498	20.9762	21.08
1000	500	3.6	0.4384	0.2428	0.2442
1000	1000	5.2	2.8442	1.566	1.5804
1000	2000	4.8	20.161	10.6712	10.291

Table 5. Monic polynomials with randomly generated coefficients

Coefficients (bit length)	Degree	No. of Roots (average)	FLQ *T(s)*	LMQ *T(s)*	$FLQ + LMQ$ *T(s)*
10	500	5.6	0.4586	0.3788	0.3864
10	1000	6.8	4.0158	2.736	2.7358
10	2000	7.2	60.453	25.0124	23.6416
1000	500	5.2	0.6028	0.4166	0.4146
1000	1000	5.2	1.943	1.3658	1.38
1000	2000	5.6	20.6296	13.7298	13.2852

Table 6. Products of factors ($x^{20}-$randomly generated integer root)

Coefficients (bit length)	Degree	No. of Roots	FLQ *T(s)*	LMQ *T(s)*	$FLQ + LMQ$ *T(s)*
20	500	50	4.178	4.6408	3.8814
20	700	70	14.4728	13.5738	14.1442
20	1000	100	60.251	52.5754	51.6798
1000	300	30	7.05	5.5642	6.371
1000	400	40	18.933	15.9492	16.65
1000	500	50	47.5842	37.6782	41.3212

Table 7. Products of factors ($x-$randomly generated integer root)

Coefficients (bit length)	Degree	No. of Roots	FLQ *T(s)*	LMQ *T(s)*	$FLQ + LMQ$ *T(s)*
10	100	100	0.2682	0.3528	0.276
10	200	200	1.3278	1.2878	1.2918
10	500	500	19.7542	21.3006	19.2678
1000	20	20	0.032	0.03	0.028
1000	50	50	0.8732	1.0298	0.8552
1000	100	100	14.529	17.321	14.1182

5. Conclusions. From the data presented in the previous section it becomes obvious that the quality of the estimates of both FLQ and LMQ is about the same, but FLQ runs faster (or quite faster) than LMQ. However, when the bounds FLQ, LMQ and $\min(FLQ, LMQ)$ are implemented in the *VAS-CF* real root isolation method it is inconclusive which one should be used.

Extensive testing of *VAS-CF* implementing various linear and quadratic complexity bounds has revealed that $VAS - CF/LMQ$ is fastest for all classes of polynomials, except when there are very many very large roots; in that case, $VAS - CF/min(FL, LM)$ is the fastest by a very small difference; in fact, a speed-up of 40% was attained when $VAS-CF/LMQ$ was compared with $VAS-CF/Cauchy$, the original implementation, [8].

Moreover, as was shown elsewhere, [6], $VAS - CF/min(FL, LM)$ is *always* faster than the Vincent-Collins-Akritas bisection real root isolation method (*VCA-bisec*)[5], [12], or any of its variants, [19]. Therefore, our current results widen the gap between *VAS-CF* and *VCA-bisec*.

REFERENCES

[1] Akritas A. G. Vincent's Theorem in Algebraic Manipulation. Ph.D. Thesis, Department of Operations Research, North Carolina State University, Raleigh, NC, 1978.

[2] Akritas A. G. The fastest exact algorithms for the isolation of the real roots of a polynomial equation. *Computing*, **24** (1980), 299–313.

[3] Akritas A. G. Elements of Computer Algebra with Applications. John Wiley Interscience, New York, 1989.

[4] Akritas A. G., A. Strzebonski. A comparative study of two real root isolation methods. *Nonlinear Analysis: Modelling and Control*, **10**, No 4 (2005), 297–304.

[5] Akritas A. G., A. Strzebonski, P. Vigklas. Implementations of a New Theorem for Computing Bounds for Positive Roots of Polynomials. *Computing*, **78**, (2006), 355–367.

[6] Akritas A. G., A. Strzebonski, P. Vigklas. Advances on the Continued Fractions Method Using Better Estimations of Positive Root Bounds. In:

[5] Misleadingly referred to (by several authors) first as "modified Uspensky's method" and recently as "Descartes' method". See [11]

Proceedings of the 10th International Workshop on Computer Algebra in Scientific Computing, CASC'2007 (Eds V. G. Ganzha, E. W. Mayr, E. V. Vorozhtsov), Bonn, Germany, September 16-20, 2007. LNCS 4770, Springer Verlag, Berlin, Heidelberg, 24–30.

[7] Akritas A. G., A. Strzebonski, P. Vigklas. 'Quadratic Complexity Bounds on the Values of Positive Roots of Polynomials (submitted).

[8] Akritas A. G., A. Strzebonski, P. Vigklas. Improving the Performance of the Continued Fractions Method Using New Bounds of Positive Roots (submitted).

[9] Akritas A. G., P. Vigklas. A Comparison of Various Methods for Computing Bounds for Positive Roots of Polynomials. *Journal of Universal Computer Science*, **13**, No 4 (2007), 455–467

[10] Alesina A., M. Galuzzi. A new proof of Vincent's theorem. *L'Enseignement Mathemathique*, **44**, (1998), 219–256.

[11] Boulier F. Systèmes polynomiaux : que signifie "résoudre"?. Université Lille 1, UE INFO 251, 2007.

[12] Collins,G. E., 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., 1976, 272–275.

[13] Demidovich P., A. Maron. Computational Mathematics. Mir Publishers, Moscow, 1987.

[14] von zur Gathen J., J. Gerhard. Fast Algorithms for Taylor Shifts and Certain Difference Equations. In: Proceedings of ISSAC'97, Maui, Hawaii, U.S.A., 1997, 40–47.

[15] Hong H. Bounds for absolute positiveness of multivariate polynomials. *J. Symb. Comput.*, **25**, No 5 (1998), 571–585.

[16] Kioustelidis B. Bounds for positive roots of polynomials. *J. Comput. Appl. Math.*, **16**, No 2 (1986), 241–244.

[17] Mignotte M. Mathematics for Computer Algebra. Springer-Verlag, New York, 1992.

[18] Obreschkoff N. Verteilung und Berechnung der Nullstellen reeller Polynome. VEB Deutscher Verlag der Wissenschaften, Berlin, 1963.

[19] Rouillier F., P. Zimmermann. Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics*, **162**, (2004), 33–50.

[20] Sharma V. Complexity of Real Root Isolation Using Continued Fractions. In: ISAAC07, preprint, 2007.

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

[22] Ştefănescu D. New bounds for positive roots of polynomials. *Journal of Universal Computer Science*, **11**, No 12 (2005), 2132–2141.

[23] Ştefănescu D. Bounds for Real Roots and Applications to Orthogonal Polynomials. In: Proceedings of the 10th International Workshop on Computer Algebra in Scientific Computing, CASC 2007 (Eds V. G. Ganzha, E. W. Mayr, E. V. Vorozhtsov), Bonn, Germany, September 16–20, 2007, LNCS 4770, Springer Verlag, Berlin, Heidelberg, 377–391.

[24] Tsigaridas E. P., I. Z. Emiris. Univariate polynomial real root isolation: Continued fractions revisited. In: ESA 2006 (Eds Y. Azar and T. Erlebach), LNCS 4168, 2006, 817–828.

[25] Vincent A. J. H. Sur la resolution des équations numériques. *Journal de Mathématiques Pures et Appliquées*, **1**, (1836), 341–372.

[26] Yap C-K. Fundamental problems of Algorithmic Algebra. Oxford University Press, 2000.

Alkiviadis G. Akritas
Andreas I. Argyris
University of Thessaly
Department of Computer and
Communication Engineering
GR-38221 Volos, Greece
e-mails:{`akritas, anargiri`}`@uth.gr`

Adam W. Strzeboński
Wolfram Research, Inc.
100 Trade Center Drive
Champaign, IL 61820, USA
e-mail: `adams@wolfram.com`

Received April 3, 2008
Final Accepted May 28, 2008