Serdica J. Computing **12** (2018), No 4, 227–246

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

LAGRANGE'S BOUND ON THE VALUES OF THE POSITIVE ROOTS OF POLYNOMIALS

Alkiviadis G. Akritas, Adam W. Strzeboński, Panagiotis S. Vigklas

To the Memory of George E. Collins, 1928–2017

Abstract. In this paper we present Lagrange's[1] theorem of 1767 for computing a bound on the values of the positive roots of polynomials, along with its interesting history and a short proof of it dating back to 1842. Since the bound obtained by Lagrange's theorem is of linear complexity, in the sequel it is called "Lagrange Linear", or `LL` for short.

Despite its average good performance, `LL` is endowed with the weaknesses inherent in *all* bounds with linear complexity and, therefore, the values obtained by it can be much bigger than those obtained by our own bound "Local Max Quadratic", or `LMQ` for short.

To level the playing field, we incorporate Lagrange's theorem into our `LMQ` and we present the new bound "Lagrange Quadratic", or `LQ` for short, the quadratic complexity version of `LL`. It turns out that `LQ` is one of the most efficient bounds available since, at best, the values obtained by it are half of those obtained by `LMQ`.

Empirical results indicate that when `LQ` replaces `LMQ` in the Vincent–Akritas–Strzeboński Continued Fractions (`VAS-CF`) real root isolation method, the latter becomes measurably slower for some classes of polynomials.

ACM Computing Classification System (1998): F.2.0, G.1.5, I.1.2.

Key words: Langrange's theorem of 1767, positive roots of polynomials, bounds on the values of the positive roots, real root isolation.

[1]Joseph-Louis Lagrange, born Giuseppe Lodovico Lagrangia (25 January 1736 – 10 April 1813): Italian mathematician.

1. Introduction. In our earlier attempts to develop the most efficient bound on the values of the positive roots of polynomials $f \in \mathbb{Z}[x]$ – for use in the `VAS-CF` real root isolation method [2], [3] – we totally missed Lagrange's theorem of 1767 on the topic ([12], p. 553), [13], [14], ([16], pp. 2–3), ([19], VIII, p. 32). This omission was due to the fact that Lagrange's theorem was almost totally forgotten, having being overshadowed by what is often encountered in the literature as the Lagrange-Maclaurin theorem ([17], Theorem 11.1, p. 48), [22], ([23], p. 150, Exercise 6.2.3(i)).

The Lagrange-Maclaurin theorem also appears without any name associated with it; see for example the books by Burnside and Panton ([4], pp. 180–181) and Dickson ([7], p. 57), or the paper by Grinstein ([8], formula III.A). We present the "culprit" following Obreschkoff ([17], p. 48).

Theorem 1. (**Lagrange-Maclaurin**) *Consider the polynomial*

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_0, \tag{1}$$

and let α be the largest absolute value of the negative coefficients. If a_{n-m} is the first negative coefficient in (1)*, then an upper bound, b, on the real roots of $f(x)$ is given by*

$$b = 1 + \sqrt[m]{\alpha}. \tag{2}$$

Proof. See Obreschkoff ([17], p. 48). □

Historical Note: When $\alpha > 1$, the bound computed above (2) is certainly an improvement over the bound

$$b = 1 + \alpha. \tag{3}$$

which Lagrange ([19], VIII, p. 32) – and later Grinstein ([8], formula III:B:1) – believed to be due to Maclaurin. However, according to the excellent work by J. Stedall, ([20], pp. 69, 160) Michel Rolle was the first to introduce the latter (3) in his *Traité d'algebre*, published in 1690 in Paris. Namely, Rolle stated without proof that an upper bound on the real roots of a polynomial can be found if:[2]

> "One selects from the negative terms of the equation that with the greatest coefficient; one ignores the sign and the unknown in this term, one divides the result by the coefficient of the first term, and to the quotient one adds unity, or a positive number greater than unity."

[2]The translation from French is by Stedall.

For a proof that the last equation (3) gives an upper bound on the values of the real roots of the polynomial (1), in the case of cubics, Stedall ([20], pp. 69, 160) refers to Reyneau (1708) and Maclaurin (1748).

Therefore, one can say with certainty that Maclaurin was involved in the development of (3). Moreover, in case the polynomial (1) has only one negative coefficient then, as we will see in Lemma 1 below, Lagrange's bound (4) reduces to simply $\sqrt[m]{\alpha}$. Hence, the bound (2) in Theorem 1 can be considered a combination of the theorems by Lagrange and by (Rolle, which was partially proved by) Maclaurin, and voilà the name Lagrange-Maclaurin.

We believe Theorem 1 forced Lagrange's own theorem of 1767 into oblivion, because in the 19th century it was much more convenient to compute just one radical instead of the many more analogous computations required by the latter (see Theorem 2 below). The *only* time Lagrange's theorem appeared in the 20th century[3] – in the form of a formula – was in Grinstein's excellent review paper ([8], p. 613, formula IV:B).

The appearance of a recent paper on the subject [6] verifies our point. Indeed, looking at Theorem 3 of that paper, we see that the author – by requiring that the polynomial has "at least two negative coefficients" – has unnecessarily altered Lagrange's theorem and restricted its range of applications. What is even worse is the fact that, despite the otherwise very interesting "Literature review" section of the paper, the author missed – among other sources – the very short, clever and elegant proof by Pury dating back to 1842 [18]. In this respect see also Batra's proof who reduced "the technicalities in the Collins-Krandick proof to a single line, bringing out the essence of the proof" [5].

Outline of the paper. In Section 2 we present Lagrange's original theorem as found in Mignotte and Ştefănescu's paper ([16], pp. 2–3), along with Pury's proof of 1842 [18]. Based on this theorem we derive the algorithm "Lagrange Linear", or `LL`, which is of linear complexity and, hence, unsuitable for use in the `VAS-CF` real root isolation method.

In Section 3 we present our quadratic complexity method "Local Max Quadratic", or `LMQ` [1], [22]. This algorithm is currently used – in various Computer Algebra Systems[4] – in the implementation, among other things, of the `VAS-CF` real root isolation method.

In Section 4 we incorporate Langrange's theorem into `LMQ` and come up with the algorithm "Lagrange Quadratic", or `LQ`. It turns out that – at its best –

[3]Besides the references to *it* by Ostrowski as explicitly cited in [6].

[4]*Mathematica*, `Sympy` and `Xcas` to name a few. `Sage` on the other hand uses min(`FL`, `LM`), the minimum of our linear complexity methods "First Lambda" and "Local Max."

the values obtained by `LQ` are half of those obtained by `LMQ`, making it thus one of the best bounds available. Tests run on a large number of random (monic) polynomials indicate that it is hard to tell whether `LMQ` or `LQ` will compute the better bound.

In Section 5 we present two tables comparing the bounds obtained by `LMQ`, `LL` and `LQ`. Additionally, in three tables, we time the performance of the `VAS-CF` real root isolation method using `LMQ`, `LQ` and `LMQ+LQ`, where the latter uses the minimum value obtained by the two bounds.

Finally, in Section 6 we present our conclusions.

2. Lagrange's Theorem and the Linear Algorithm `LL`. On p. 553 of his original paper [12][5] – or on p. 32 of his famous book [14], which constitutes the 8th volume of *Œuvres de Lagrange*, edited by Joseph Alfred Serret [19] – Lagrange only states that given the polynomial F, where

$$-\mu y^{r-m} - \nu y^{r-n} - \bar{\omega} y^{r-p} - \cdots$$

are its "negative terms", an upper bound for the real roots of F is given by the sum of the first two largest of the quantities

$$\sqrt[m]{\mu},\ \sqrt[n]{\nu},\ \sqrt[p]{\bar{\omega}}, \ldots$$

or "a number larger than this sum".

Due to the importance of Lagrange's statement we also present a modern version of it as (partially) found in Mignotte and Ştefănescu's paper ([16], p. 3).

Note that Lagrange gave no proof of his statement; he simply mentioned that it is similar to the proof of (3) and moved on.

In 1842, Lagrange's statement appeared as an exercise to be proved in the section "Théorémes à démontrer. Problèmes" of *Nouvelles annales de mathématiques*, 1[e] série, tome 1 (1842), p. 57–59, Exercise 6.

The proof – given by A. Pury [18] – appeared the very same year in *Nouvelles annales de mathématiques*, 1[e] série, tome 1 (1842), p. 243–244 under the title "Solution du problème 6. Limite de Lagrange."

Theorem 2. (**Lagrange, 1767**) *Let f, as in (1), be a non constant monic polynomial of degree n over $\mathbb{R}$ and let $a_{n-j} : j \in J$ be the set of its negative coefficients. Then an upper bound for the positive real roots of f is given by the sum of the largest and the second largest members in the set $\left\{ \sqrt[j]{|a_{n-j}|} : j \in J \right\}$. That is,*

$$b = \max_{\{a_{n-i}, a_{n-k} \in J\}} \left(\sqrt[i]{|-a_{n-i}|} + \sqrt[k]{|-a_{n-k}|} \right). \tag{4}$$

[5] Presented to the Berlin Academy on April 20, 1769.

P r o o f. (**Pury, 1842**) The worst possible case to consider is obviously when, starting from the second term, all the coefficients of the polynomial are negative, of the form

$$(5)\quad x^n - A_1x^{n-1} - A_2x^{n-2} - A_3x^{n-3} - \cdots - A_px^{n-p} - \cdots - A_rx^{n-r} - \cdots - A_n = 0,$$

from which we conclude that

(6)

$$1 = \frac{A_1}{x} + \left(\frac{\sqrt{A_2}}{x}\right)^2 + \left(\frac{\sqrt[3]{A_3}}{x}\right)^3 + \cdots + \left(\frac{\sqrt[p]{A_p}}{x}\right)^p + \cdots + \left(\frac{\sqrt[r]{A_r}}{x}\right)^r + \cdots + \left(\frac{\sqrt[n]{A_n}}{x}\right)^n.$$

Let $\sqrt[p]{A_p}$ and $\sqrt[r]{A_r}$ be the two largest quantities of the sequence A_1, $\sqrt{A_2}$, $\sqrt[3]{A_3}$, $\ldots$, $\sqrt[n]{A_n}$ and let $\sqrt[p]{A_p} = \lambda\sqrt[r]{A_r}$, where λ is initially considered greater than 1.

Setting $x = \sqrt[p]{A_p} + \sqrt[r]{A_r} = (\lambda+1)\sqrt[r]{A_r}$ the fraction $\dfrac{\sqrt[p]{A_p}}{x}$ becomes equal to $\dfrac{\lambda}{\lambda+1}$, whereas *all* the other fractions in (6) become smaller than $\dfrac{1}{\lambda+1}$—since $\sqrt[r]{A_r}$ is greater than any of the other quantities A_1, $\sqrt{A_2}$, $\sqrt[3]{A_3}$, etc. Hence, if we replace x by this value, the right-hand side of (6) becomes smaller than the sum

(7)

$$\frac{1}{1+\lambda} + \frac{1}{(1+\lambda)^2} + \frac{1}{(1+\lambda)^3} + \cdots + \frac{1}{(1+\lambda)^p} + \cdots + \frac{1}{(1+\lambda)^n} + \left(\frac{\lambda}{1+\lambda}\right)^p - \frac{1}{(1+\lambda)^p},$$

and, since $\lambda + 1 > 1$, the sum in (7) is smaller than $\dfrac{1}{\lambda} + \left(\dfrac{\lambda}{1+\lambda}\right)^p - \dfrac{1}{(1+\lambda)^p}$.

However, for the trinomial, we have

$$\frac{1}{\lambda} + \left(\frac{\lambda}{1+\lambda}\right)^p - \frac{1}{(1+\lambda)^p} < 1,$$

which is easily seen from $\dfrac{\lambda^p - 1}{(1+\lambda)^p} < \dfrac{\lambda - 1}{\lambda}$ if we divide both sides by $\lambda - 1$.

Therefore, replacing x by the sum $\sqrt[p]{A_p} + \sqrt[r]{A_q}$ the first part of equation (6) becomes á fortiori greater than the second part, and the same is true for equation (5).

In case $\lambda = 1$, the sum in equation (7) is replaced by the geometric series

$$\frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^n} = 1 - \frac{1}{2^n} < 1. \qquad \square$$

Note in Theorem 2 that if the set J is empty then the polynomial f has no positive root and 0 is a bound. If, on the other hand, the set J is a singleton, then a bound is computed by the following lemma.

Lemma 1. *Let f, as in (1), be a non constant monic polynomial of degree n and let α be the absolute value of the single negative coefficient in the term of degree $n-m$. Then an upper bound on the positive root of f is given by*[6]

$$b = \sqrt[m]{\alpha}. \tag{8}$$

Proof. For $x > \sqrt[m]{\alpha}$ we have

$$f \geq x^n - a_{n-m}x^{n-m} = x^{n-m}(x^m - a_{n-m}) > 0. \qquad \square$$

Therefore, for the class of polynomials with *one* negative coefficient, Lagrange's bound (4), in Theorem 2, reduces to (8), which is better than the Lagrange-Maclaurin bound (2).

The requirement in Theorem 2 – and in Lemma 1 – that the polynomial be monic makes the proof easier but is not actually needed for its algorithmic implementation. As a matter of fact it helps the presentation in Section 2 – and in subsequent sections – if the polynomial is *not* monic. However, the leading coefficient has to be *positive*.

Algorithmic Implementation of Lagrange's Theorem. Consider the polynomial

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0, \tag{9}$$

with $a_n \geq 1$. To compute a bound on the positive roots of f using Lagrange's theorem we proceed as follows:

- each negative coefficient a_{n-j} of f is "paired" with the leading coefficient a_n and the radical $\sqrt[j]{-\frac{a_{n-j}}{a_n}}$ is computed,
- the bound is the sum of the largest two radicals.

An algorithmic description of the above is presented in Algorithm 1 below.

Obviously, `LL` is of linear complexity and, as discussed elsewhere [1], [22], it cannot be used in the `VAS-CF` real root isolation method [2], [3]. The following example explains the reason why.

[6] See also Ştefănescu's work [21].

Input: A univariate polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \in \mathbb{Z}[x]$, with $a_n > 0$.
Output: An upper bound on the values of the positive roots of $f(x)$.

`// form cl, the coefficients list`
$cl \leftarrow [a_n, a_{n-1}, a_{n-2}, \ldots, a_0]$; `/* list enumeration begins with 0 */`
`// form ncl, the negative coefficients list of pairs` $[\frac{a_{n-j}}{a_n}, j]$.
$ncl \leftarrow$ [for $1 \leq j \leq n$ if $cl[j] < 0$ form the pair $[\frac{cl[j]}{a_n}, j]]$; `/* OK to contain just one pair */`
if $ncl = []$ **then return** 0;
; `/* no positive roots */`
`// form rl, the list of radicals`
$rl \leftarrow$ [for each pair $[\frac{cl[j]}{a_n}, j] \in ncl$ evaluate $\sqrt[j]{-\frac{cl[j]}{a_n}}$]; `/* OK to contain just one radical */`
$rl \leftarrow \mathrm{sort}(rl)$; `/* sort rl in increasing order */`
return $\mathrm{sum}(rl[-2:\ ])$; `/* the sum of the largest two values. */`

Algorithm 1. `LL(f, x)`, Lagrange's Linear algorithm.

Example 1. Consider the third degree polynomials

$$\begin{aligned} f_1 &= x^3 + 1000000x^2 - 1000000x - 1, \\ f_2 &= x^3 - x^2 + 10000000x - 10000000, \end{aligned}$$

where, for both polynomials, the only positive root is 1. The bounds obtained by Lagrange's theorem are

$$\begin{aligned} \mathrm{LL}(f_1, x) &= 1001.00, \\ \mathrm{LL}(f_2, x) &= 216.44. \end{aligned}$$

To see why these bounds are unacceptable, we run ahead of ourselves and present the bounds obtained with: (a) `LMQ`, the "Local Max Quadratic" method, described in Section 3, and (b) `LQ`, the "Lagrange Quadratic" method, described in Section 4.

The bounds obtained by our `LMQ` are

$$\begin{aligned} \mathrm{LMQ}(f_1, x) &= 2.00, \\ \mathrm{LMQ}(f_2, x) &= 2.00, \end{aligned}$$

whereas, the bounds obtained by the newly developed method `LQ` are

$$\begin{aligned} \mathrm{LQ}(f_1, x) &= 1.001, \\ \mathrm{LQ}(f_2, x) &= 1.00. \end{aligned}$$

From Example 1, it becomes clear that we need to develop `LQ`, the quadratic complexity version of `LL`. However, to do so we need a thorough understanding of the `LMQ` algorithm, in which we will embed Lagrange's theorem.

3. Our Local Max Quadratic Algorithm `LMQ`. `LMQ` is one of the best algorithms with quadratic complexity for computing bounds on the values of positives roots of polynomials.[7] It is an implementation of Theorem 2 in [3] – which is the same as Theorem 5 in [1] – and has been used to improve the performance of the `VAS-CF` real root isolation method.

`LMQ` is a *max-min* algorithm, which – given f as in (9) – computes the bound as follows:[8]

- *each* negative coefficient a_{n-i} of the polynomial is "paired" with *each one* of the preceding positive coefficients a_{n-j}, $(i > j)$ and the minimum is taken of all the radicals of the form

  $$\sqrt[i-j]{\frac{-a_{n-i}}{\frac{a_{n-j}}{2^{t_{n-j}}}}}, \tag{10}$$

  where the exponent t_{n-j} is explained below, and next
- the maximum of all those minimums is taken as the estimate of the bound.

In other words we have:

$$b = \max_{\{a_{n-i}<0\}} \min_{\{a_{n-j}>0 : i>j\}} \sqrt[i-j]{\frac{-a_{n-i}}{\frac{a_{n-j}}{2^{t_{n-j}}}}}. \tag{11}$$

What is unusual about our method is the exponent t_{n-j} in the expression $2^{t_{n-j}}$ in (11); t_{n-j} counts the number of times the corresponding *positive* coefficient a_{n-j} has been "paired" with various negative coefficients to produce a minimum. That is, t is a list of length $n+1$, and it is through this list that we will embed Lagrange's theorem in `LMQ` (in Section 4), to develop `LQ`.

By comparison, Hong's method [9] – which is equivalent to `KQ`, Kioustelidis' quadratic complexity method [1], [10] – uses a formula almost identical to (11). The difference is that `KQ` uses the expression 2^{i-j} instead of $2^{t_{n-j}}$ and,

---

[7] It should be noted that time is not of importance in our case since, in the `VAS-CF` real root isolation method, these bounds are estimated *before* a translation of complexity at *least* $O(n^2)$ is executed.

[8] To make the transition to `LQ` easier, we present here a slight variation of the original version of `LMQ` [22].

hence, the bounds obtained by it are greater than or equal to those obtained by `LMQ`.[9] Details can be found elsewhere [1].

An algorithmic description of our method `LMQ` is presented in Algorithm 2.

```
Input: A univariate polynomial f(x) = a_n x^n + a_{n-1} x^{n-1} + ··· + a_0 ∈ Z[x], with a_n > 0.
Output: An upper bound on the values of the positive roots of f(x).

// at least one sign variation (v ≥ 1)?
cl ← [a_0, a_1, a_2, ..., a_{n-1}, a_n];                /* list of length n+1 */
v ← number of sign variations in cl;
if v = 0 then return 0;
;
// initialize variables
b ← 0;
m ← length(cl);
t ← [1, 1, 1, ..., 1, 1];                               /* list of length n+1 */
// loop within loop
for i = m − 1 to 1 step −1 do
    if cl(i − 1) < 0 then
        index ← m;
        tmp ← +∞;
        for j = m to i + 1 step −1 do
            if cl(j − 1) > 0 then
                q ← (2^{t[j−1]} (− cl[i − 1] / cl[j − 1]))^{1/(j−i)};
                if q < tmp then
                    tmp ← q;
                    index ← j − 1;
                end
            end
        end
        t[index] ← t[index] + 1;
        if b < tmp then
            b ← tmp;
        end
    end
end
return b
```

Algorithm 2. `LMQ(f, x)`, the "Local Max Quadratic" algorithm.

4. Lagrange's Quadratic Algorithm `LQ`. As we saw in the previous section, the Local Max Quadratic algorithm, `LMQ`, uses the list t of length $n+1$, in which initially all entries are 1. The entry $t_{n-j} = t[n-j]$ corresponds to the coefficient a_{n-j} of the polynomial and if $a_{n-j} > 0$, then $t[n-j]$ counts the number of times a_{n-j} has been "paired" with various negative coefficients a_{n-i}, with $i > j$,

[9] However, Hong's bound is better suited for the study of the complexity analysis of the `VAS-CF` real root isolation method. Indeed, using Hong's bound an improvement was achieved on the time bounds of `VAS-CF` [15].

to produce a minimum value.

In `LQ` we use the list (of lists) t of length $n+1$, in which initially each entry is [ ], the empty list. Again, the list $t_{n-j} = t[n-j]$ corresponds to the coefficient a_{n-j} of the polynomial but now, if $a_{n-j} > 0$, then the list $t[n-j]$ contains all the minimum values produced by $a_{n-j} > 0$ when "paired" with various negative coefficients a_{n-i}, with $i > j$.

Clearly, for any given $a_{n-j} > 0$, the list $t[n-j]$ may be the empty list, or a singleton, or it may contain more than one value.

Therefore, the Lagrange Quadratic algorithm `LQ` is a variation of `LMQ` and – given f as in (9) – the bound is computed as follows:

- *each* negative coefficient a_{n-i} of the polynomial is "paired" with *each one* the preceding positive coefficients a_{n-j}, $(i > j)$ and the minimum is taken of all the radicals of the form

$$\sqrt[i-j]{\frac{-a_{n-i}}{a_{n-j}}} \tag{12}$$

  as indicated in Lagrange's theorem (Theorem 2); each minimum is then appended to the corresponding list $t[n-j]$,

- we initialize a temporary bound to 0, and then for each non-empty list $t[n-j]$ we proceed as follows: (a) if the list $t[n-j]$ has a single element, and its value is greater than the temporary bound, then *it* (the single element) becomes the temporary bound and, (b) if the list $t[n-j]$ has more than one element, we sort them in increasing order and take the sum of the largest two; if the sum is greater than the temporary bound, then *it* (the sum) becomes the temporary bound; at the end the temporary bound is taken as the estimate of the bound.

An algorithmic description of Lagrange's quadratic method `LQ` is presented in Algorithm **??** below.

The following theorem establishes a relation between the bounds computed by the quadratic algorithms `LMQ` and `LQ`.

Theorem 3. *Let b_{LMQ} and b_{LQ} denote bounds computed using, respectively, algorithm `LMQ` and `LQ`. Then we have*

$$\frac{b_{LMQ}}{2} \leq b_{LQ} < 2 \cdot b_{LMQ}. \tag{13}$$

Input: A univariate polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \in \mathbb{Z}[x]$, with $a_n > 0$.
Output: An upper bound on the values of the positive roots of $f(x)$.

```
// at least one sign variation (v ≥ 1)?
cl ← [a_0, a_1, a_2, ..., a_{n-1}, a_n];                      /* list of length n+1 */
v ← number of sign variations in cl;
if v = 0 then return 0;
;
if v = 1 then return value computed by Lemma 1 ;
;
// initialize variables
m ← length(cl);
t ← [ [ ], [ ], [ ], ..., [ ], [ ] ];                          /* list of length n+1 */
// main loop, which is almost identical to the one in LMQ
for j = 0 to m − 1 step 1 do
    if cl(j) < 0 then
        b ← +∞;
        index ← m;
        for k = j + 1 to m − 1 step 1 do
            if cl(k) > 0 then
                q ← (−cl[j]/cl[k])^{1/(k−j)};
                if q < b then
                    b ← q;
                    index ← k;
                end
            end
        end
        t[index] ← append(t[index], b);
    end
end
// secondary loop to process the list of lists t
b ← 0;
for j = 0 to m − 1 step 1 do
    tp ← t[j];
    if tp ≠ [] then
        if length(tp) = 1 then
            tp ← tp[0];                                  /* enumeration starts from 0 */
        else
            tp ← sort(tp);                              /* sort tp in increasing order */
            tp ← sum(tp[−2 :]);                        /* sum of the largest two values */
        end
        if tp > b then
            b ← tp;
        end
    end
end
return b
```

Algorithm 3. `LQ(f, x)`, Lagrange's Quadratic Algorithm.

P r o o f. The value of t_{n-j} in (11) counts the number of times a_{n-j} has been "paired" with various negative coefficients a_{n-k} , with $j < k \leq i$, to produce a minimum value. Hence $t_{n-j} \leq i - j$. Therefore,

$$\begin{aligned} b_{LMQ} &= \max_{\{a_{n-i}<0\}} \min_{\{a_{n-j}>0\,:\,i>j\}} \sqrt[i-j]{\frac{-a_{n-i}}{\frac{a_{n-j}}{2^{t_{n-j}}}}} \\ &= \max_{\{a_{n-i}<0\}} \min_{\{a_{n-j}>0\,:\,i>j\}} \sqrt[i-j]{2^{t_{n-j}}} \sqrt[i-j]{\frac{-a_{n-i}}{a_{n-j}}} \\ &\leq \max_{\{a_{n-i}<0\}} \min_{\{a_{n-j}>0\,:\,i>j\}} 2 \sqrt[i-j]{\frac{-a_{n-i}}{a_{n-j}}}. \end{aligned}$$

Since for all $a_{n-i} < 0$ we have

$$b_{LQ} \geq \min_{\{a_{n-j}>0\,:\,i>j\}} \sqrt[i-j]{\frac{-a_{n-i}}{a_{n-j}}},$$

it follows that $b_{LMQ} \leq 2 \cdot b_{LQ}$.

On the other hand, let $a_{n-j_0} > 0$ be such that b_{LQ} is obtained as the sum of the *largest* two radicals in the list $t_{n-j_0} = t[n - j_0]$. Then,

$$b_{LQ} \leq \sqrt[i_1-j_0]{\frac{-a_{n-i_i}}{a_{n-j_0}}} + \sqrt[i_2-j_0]{\frac{-a_{n-i_2}}{a_{n-j_0}}} \tag{14}$$

where, for $k = 1, 2$, we hav

$$\sqrt[i_k-j_0]{\frac{-a_{n-i_k}}{a_{n-j_0}}} = \min_{\{a_{n-j}>0\,:\,i_k>j\}} \sqrt[i_k-j]{\frac{-a_{n-i_k}}{a_{n-j}}}.$$

If $t[n - j_0]$ is a singleton the inequality (14) is obtained by taking $i_1 = i_2$.[10] Since in (11) $t_{n-j} \geq 1$, it follows that $\sqrt[i-j]{2^{t_{n-j}}} > 1$. Hence, for $k = 1, 2$, we have

$$\begin{aligned} \sqrt[i_k-j_0]{\frac{-a_{n-i_k}}{a_{n-j_0}}} &\leq \max_{\{a_{n-i}<0\}} \min_{\{a_{n-j}>0\,:\,i>j\}} \sqrt[i-j]{\frac{-a_{n-i}}{a_{n-j}}} \\ &< \max_{\{a_{n-i}<0\}} \min_{\{a_{n-j}>0\,:\,i>j\}} \sqrt[i-j]{2^{t_{n-j}}} \sqrt[i-j]{\frac{-a_{n-i}}{a_{n-j}}} \quad = \quad b_{LMQ}. \end{aligned}$$

Therefore $b_{LQ} < 2 \cdot b_{LMQ}$. □

[10] In which case (14) is a strict inequality.

In other words, the advantage of `LQ` over `LMQ` is only in the strict inequality on the right vs. the weak inequality on the left. The second inequality in (13) is approached for $x^n - x - 1$, when $n \to \infty$.

To demonstrate Theorem 3 we run several experiments with a large number of random polynomials and random monic polynomials and the results were quite interesting. Namely, we found out that:

- for random polynomials `LMQ` gives – on average – better bounds than `LQ`, whereas,
- for random monic polynomials `LQ` gives – on average – better bounds than `LMQ`.

In Tables 1 and 2 that follow, each result is based on a sample of 1000 polynomials. As in Theorem 3, b_{LMQ} (b_{LQ}) denote bounds computed using the algorithm `LMQ` (`LQ`). The column marked Mean gives the *geometric* mean of the ratio b_{LMQ}/b_{LQ}. The columns marked Min and Max give the minimal and the maximal value of the ratio b_{LMQ}/b_{LQ}. The column marked `LMQ better` (resp. `LQ better`) gives the number of polynomials (out of 1000) for which $b_{LMQ} < b_{LQ}$ (resp. $b_{LMQ} < b_{LQ}$).

In Table 1 we used dense polynomials of degree n with uniformly distributed randomly generated coefficients with s decimal digits.

Table 1. For random polynomials `LMQ` gives better bounds. Notice that the mean value of the ratio b_{LMQ}/b_{LQ} is (almost) always smaller than 1.

		Bounds computed by `LMQ` and `LQ`				
n	s	Mean	Min	Max	`LMQ better`	`LQ better`
10	3	1.04568	0.611428	2.	492	507
10	100	1.0495	0.577506	2.	486	514
100	3	0.941566	0.650993	2.	644	356
100	100	0.938535	0.637656	2.	666	334
1000	3	0.933784	0.67628	2.	663	337
1000	100	0.920236	0.644298	2.	697	303

In Table 2 we used dense monic polynomials of degree n with uniformly distributed randomly generated coefficients with s decimal digits.

5. Empirical results. In this section we compare the bounds obtained by the three methods `LMQ`, `LL` and `LQ` described above. Along with the bounds we

Table 2. For random monic polynomials LQ gives better bounds. Notice that the mean value of the ratio b_{LMQ}/b_{LQ} is always greater than 1.

		Bounds computed by LMQ and LQ				
n	s	Mean	Min	Max	LMQ better	LQ better
10	3	1.4382	0.59386	2.	215	784
10	100	1.48369	0.577506	2.	211	788
100	3	1.34263	0.652529	2.	323	677
100	100	1.39559	0.646699	2.	316	684
1000	3	1.35768	0.670705	2.	329	671
1000	100	1.35888	0.664029	2.	343	657

also compute numerically the maximum positive root, *MaxRoot*, of each polynomial. Moreover, in Subsection 5 we time the performance of the VAS-CF real root isolation method, implemented with LMQ, LQ and LMQ+LQ $=$ min(LMQ, LQ) to compute the bounds.

In Table 3 we follow the standard practice and use as benchmark the Laguerre[11], Chebyshev (first[12] kind), Wilkinson[13] and Mignotte[14] polynomials, of degrees $\{10, 100, 1000\}$. Notice how well both LL and LQ are doing.

In Table 4 we test two extreme-case polynomials. The weakness of LL is revealed by the first polynomial. This table shows clearly the inequalities of Theorem 3.

VAS-CF implemented with LMQ, LQ and LMQ+LQ. In this subsection we time the performance of the VAS-CF real root isolation method, implemented with three different quadratic algorithms for computing bounds: LMQ, LQ and LMQ+LQ $=$ min(LMQ, LQ).

The tests were run on a Linux virtual machine with 8 GB of RAM on a laptop with Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz processor (it has 4 physical/8 virtual cores, but the implementation of VAS-CF is sequential). The

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

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

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

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

Table 3. Special Polynomials: 1. Laguerre (L_{10}), 2. Laguerre (L_{100}), 3. Laguerre (L_{1000}), 4. Tchebyshev (T_{10}), 5. Tchebyshev (T_{100}), 6. Tchebyshev (T_{1000}), 7. Wilkinson (W_{10}), 8. Wilkinson (W_{100}), 9. Wilkinson (W_{1000}), 10. Mignotte (M_{10}), 11. Mignotte (M_{100}), 12. Mignotte (M_{1000})

	Bounds			
Polynomials	`LMQ`	`LL`	`LQ`	*MaxRoot*
1	200.	144.208	100.	29.9207
2	20000.	15393.3	10000.	374.984
3	2.0×10^6	$1.549\,22 \times 10^6$	1.0×10^6	3943.25
4	2.23607	2.54083	1.58114	0.987688
5	7.07107	8.65267	5.	0.999877
6	22.3607	27.5232	15.8114	0.999999
7	110.	81.28	55.	10.
8	10100.	7792.13	5050.	100.
9	1.001×10^6	775569.	500500.	1000.
10	1.77828	2.70246	1.63069	1.5763
11	1.04811	2.04768	1.04073	1.03618
12	1.00463	2.00462	1.00393	1.00348

Table 4. Extreme-case polynomials: $1. x^3 + 10^{100}x^2 - 10^{100}x - 1$, $2. x^{100} - x - 1$

	Bounds			
Polynomials	`LMQ`	`LL`	`LQ`	*MaxRoot*
1	2.	1.0×10^{50}	1.	1.
2	1.01396	2.	2.	1.00699.

times are all in milliseconds.

In Table 5 we use the benchmark polynomials Laguerre, Chebyshev (first kind), Wilkinson and Mignotte, of degrees $\{100, 500, 1000\}$. As we see, in most cases, `VAS-CF(LMQ)` is measurably faster than `VAS-CF(LQ)`.

Table 5. Special Polynomials: 1. Laguerre (L_{100}), 2. Laguerre (L_{500}), 3. Laguerre (L_{1000}), 4. Tchebyshev (T_{100}), 5. Tchebyshev (T_{500}), 6. Tchebyshev (T_{1000}), 7. Wilkinson (W_{100}), 8. Wilkinson (W_{500}), 9. Wilkinson (W_{1000}), 10. Mignotte (M_{100}), 11. Mignotte (M_{500}), 12. Mignotte (M_{1000})

	Timing `VAS-CF` in ms		
Degree	`VAS-CF(LMQ)`	`VAS-CF(LQ)`	`VAS-CF(LMQ+LQ)`
100	56.991	52.992	44.993
500	200.00	153.933	100.00
1000	7675.83	9682.53	8061.78
100	26.997	18.997	12.998
500	700.894	948.856	727.888
1000	5828.11	8324.73	5922.1
100	6.998	7.999	6.999
500	855.87	876.866	886.866
1000	9976.48	10162.5	10214.4
100	5	3.999	3
500	54.991	47.993	43.993
1000	207.969	219.966	205.969

In Table 6 we test random polynomials of degrees $\{10, 100, 1000\}$ and coefficient sizes $\{3, 100\}$. The entries are averages of 1000 runs.

In Table 7 we test random monic polynomials of degrees $\{10, 100, 1000\}$ and coefficient sizes $\{3, 100\}$. The entries are averages of 1000 runs.

6. Conclusions. In this paper we have presented Lagrange's original theorem of 1767 for computing upper bounds on the positive roots of polynomials and have provided a simple, short proof to it by Pury, dating back to 1842.

The bounds computed by Lagrange's Linear complexity algorithm `LL` are, on average, very good but the algorithm itself cannot be recommended for use in the `VAS-CF` real root isolation method.

Based on our previous experience, we have developed `LQ`, the quadratic

Table 6. Here again, as the degree of the polynomials gets bigger and bigger, `VAS-CF(LMQ)` becomes measurably faster than `VAS-CF(LQ)`.

		Timing `VAS-CF` in ms		
Degree	Size	`VAS-CF(LMQ)`	`VAS-CF(LQ)`	`VAS-CF(LMQ+LQ)`
10	3	0.036994	0.037994	0.036995
10	100	0.027996	0.026996	0.029995
100	3	2.42563	2.80457	2.6056
100	100	1.08084	1.3468	1.08184
1000	3	373.016	507.339	376.122
1000	100	146.753	271.506	147.312

Table 7. Contrary to expectations – founded on Table 2 – for random monic polynomials, `VAS-CF(LMQ)` becomes measurably faster than `VAS-CF(LQ)`, as the degree of the polynomials get bigger and bigger.

		Timing `VAS-CF` in ms		
Degree	Size	`VAS-CF(LMQ)`	`VAS-CF(LQ)`	`VAS-CF(LMQ+LQ)`
10	3	0.040993	0.040994	0.040994
10	100	0.031995	0.032995	0.033995
100	3	2.49662	2.80557	2.53062
100	100	1.07584	1.39479	1.10283
1000	3	371.947	503.066	374.647
1000	100	152.933	297.876	154.951

version of `LL`, by embedding Lagrange's theorem into our "Local Max Quadratic" (`LMQ`) method.

The bounds computed by Lagrange's Quadratic complexity algorithm `LQ`, are comparable to – and at best half of – those computed by `LMQ`.

Despite the overall very good performance of `LQ`, empirical results have demonstrated that we have *nothing* to gain by implementing it in the `VAS-CF` real root isolation method. However, as we have shown – both theoretically and empirically – the combined method `LQ+LMQ` is today the best method to obtain a bound on the values of the positive roots of polynomials. We plan to implement this method in the future in a computer algebra package.

REFERENCES

[1] Akritas A. G. Linear and Quadratic Complexity Bounds on the Values of the Positive Roots of Polynomials. *Journal of Universal Computer Science*, **15** (2009), No 3, 523–537.

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

[3] Akritas A. G., A. W. Strzeboński, P. S. Vigklas. Improving the Performance of the Continued Fractions Method Using new Bounds of Positive Roots. *Nonlinear Analysis: Modelling and Control*, **13** (2008), No 3, 265–379.

[4] Burnside W. S., A. W. Panton. The theory of equations: with an introduction to the theory of binary algebraic forms. Hodges, Figgis, & Co., Dublin, 1886.

[5] Batra P. On a proof of Lagrange's bound for real roots. Preprint, September 2, 2015.

[6] Collins G. E. Krandick's proof of Lagrange's real root bound claim. *Journal of Symbolic Computation*, **70** (2015), 106–111.

[7] Dickson L. E. Elementary theory of equations. New York, Wiley, 1914.

[8] Grinstein L. S. Upper Limits to the Real Roots of Polynomial Equations. *The American Mathematical Monthly*, **60** (1953), No 9, 608–615.

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

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

[11] Lagrange J.-L. Liste des Ouvrages de Lagrange. In: M. J. Bertrand (ed.). Mécanique Analytique, Vol. 2, Paris, 1855. `https://gdz.sub.uni-goettingen.de/dms/load/img/?PPN=PPN622996614&DMDID=DMDLOG_0007&LOGID=LOG_0011&PHYSID=PHYS_0391`, 10 February 2020.

[12] Lagrange J.-L. Sur la résolution des équations numériques (1767). *Mémoires de l'Académie Royale des Sciences et des Belle-Lettres de Berlin*, **23** (1769), 539–578. This is in volume 2 of [19].[15]

[13] Lagrange J.-L. Additions au mémoire sur la résolution des équations numériques. *Mémoires de l'Académie Royale des Sciences et des Belle-Lettres de Berlin*, **24** (1770), 581–652. This is in volume 2 of [19].

[14] Lagrange J.-L. Traité de la résolution des équations numériques de tous les degrés. Paris, 1808. This is volume 8 of [19].

[15] Mehlhorn K., S. Ray. Faster algorithms for computing Hong's bound on absolute positiveness. *Journal of Symbolic Computation*, **45** (2010), 677–683.

[16] Mignotte M., D. Ştefănescu. On an estimation of polynomial roots by Lagrange. HAL, 2002. `https://hal.archives-ouvertes.fr/hal-00129675`, 10 February 2020.

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

[18] Pury A. Solution du problème 6. Limite de Lagrange. *Nouvelles annales de mathématiques, journal des candidats aux ecoles polytechnique et normale*, 1^{e} série, **1** (1842), 243–244.

[19] Serret J. A. (ed.). Œuvres de Lagrange. Gauthier-Villars, Paris, 1879.

[15] The digitized form of *Œuvres de Lagrange* [19], Lagrange's collected work, can be found on the sites `http://sites.mathdoc.fr/OEUVRES/`, and `https://gdz.sub.uni-goettingen.de/`. Following Stedall's remark ([20], p. 209), Lagrange's article is listed here with two dates: the first one is the year when the paper is known to have been written – as recorded by Lagrange himself ([11], p. 384) – and the other one is the year in which the volume of papers was published.

[20] STEDALL J. From Cardano's great art to Lagrange's reflections: filling a gap in the history of algebra. European Mathematical Society, 2011.

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

[22] VIGKLAS P. S. Upper bounds on the values of the positive roots of polynomials. Ph. D. Thesis, University of Thessaly, Volos, Greece, 2010.

[23] YAP C. Fundamental Problems in Algorithmic Algebra. Oxford University Press, 2000.

Alkiviadis G. Akritas, Panagiotis S. Vigklas
Department of Electrical and Computer Engineering
University of Thessaly
GR-38221, Volos, Greece
e-mail: `{akritas, pviglas}@uth.gr`

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

Received 23 April 2018
Final Accepted 25 July 2018