Accelerating the Numerical Computation of Positive Roots of Polynomials using Improved Bounds

Kinji Kimura[1], Takuto Akiyama[2], Hiroyuki Ishigami[3], Masami Takata[4], and Yoshimasa Nakamura[5]

[1,2,3,5]Graduate School of Informatics, Kyoto University,

[4]Academic Group of Information and Computer Sciences, Nara Women's University

Aim

Accelerating
the numerical computation of positive roots of polynomials using improved bounds

Download site

You can download our program from the following website:
URL:
http://www-is.amp.i.kyoto-u.ac.jp/
kkimur/REALROOT.html

Strategy

Based on exact computation using arbitrary precision arithmetic(GMP:GNU Multiple Precision Arithmetic Library) and

adoption of the continued fraction (CF) method

Motivation(1)

Nonlinear polynomial equations:

$$c_0 + c_1 + c_2 = 0, c_0c_1 + c_0c_2 + c_1c_2 = 0,$$
$$c_0c_1c_2 - 2 = 0$$

By virtue of Gröbner basis, we can get

$$c_0 + c_1 + c_2 = 0, c_1^2 + c_2^2 + c_2c_1 = 0,$$
$$c_2^3 - 2 = 0.$$

After some computations, c_2 can be regarded as a separable variable.

Motivation(2)

"separable" means,

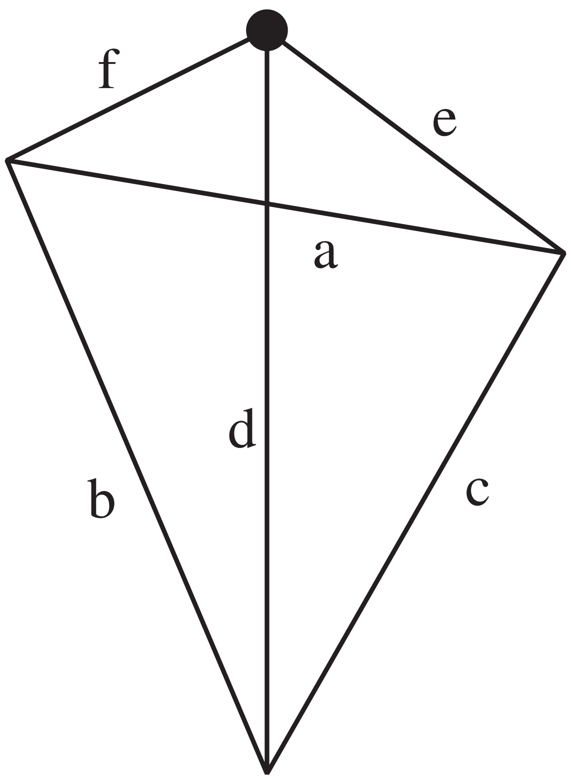
if the root of c_2 is real,

then roots of all variables(c_0, c_1, c_2) are real.

For the univariate polynomial:$c_2^3 - 2 = 0$, we can apply the CF method. We can get the real root of c_2 which is isolated into a specific interval $(0, 2]$.

Motivation(3)

A famous problem in a history of Japanese mathematics,



$$
\begin{aligned}
p_1 &= a^2d^2(b^2+c^2+e^2+f^2)-a^2d^4-a^4d^2 \\
&\quad +b^2e^2(a^2+c^2+d^2+f^2)-b^2e^4-b^4e^2 \\
&\quad +c^2f^2(a^2+b^2+e^2+d^2)-c^2f^4-c^4f^2 \\
&\quad -a^2b^2c^2-a^2e^2f^2-b^2f^2d^2-c^2d^2e^2 \\
p_2 &= d^3-b^3-271 \\
p_3 &= b^3-c^3-217 \\
p_4 &= c^3-a^3-608/10 \\
p_5 &= a^3-e^3-3262/10 \\
p_6 &= e^3-f^3-61
\end{aligned}
$$

Motivation(4)

- By virtue of Gröbner basis, we can regard a variable f as a separable variable.
- The degree of the univariate polynomial of f is 1458. (The coefficients of the degree of multiples of 3 are not zero and the other coefficients are zero.)
- **In order to solve nonlinear polynomial equations with Gröbner basis, we have to isolate positive roots of a higher-degree univariate polynomial.**

Demo

From the feature that the coefficients of the degree of multiples of 3 are not zero and the other coefficients are zero, we can transform the univariate polynomial of the degree 1458 of f into the univariate polynomial of the degree 486 with $x = f^3$.

We just treat the univariate polynomial of the degree 486.

Continued fraction method(1)

Theorem 1(Descartes' rule of signs)

For a polynomial equation

$$f(x) = a_0 x^n + \cdots + a_{n-1}x + a_n = 0,\ x \in \mathbb{R},\ a_i \in \mathbb{R},$$

$W =$ the number of "changes of sign" in the list of coefficients $\{a_0,\ a_1,\ \ldots,\ a_n\}$, except for $a_i = 0$

$N =$ the number of positive roots in $(0, \infty)$

Under these definitions, the following relation holds:

$$N = W - 2h,$$

where h is a non-negative integer.

Continued fraction method(2)

Using Theorem 1, the number of positive roots of $f(x) = 0$ is determined as the following conditional branch:

- Case where $W = 0$: $f(x) = 0$ does not have any positive roots in the interval $x \in (0, \infty)$.
- Case where $W = 1$: $f(x) = 0$ has only one positive root in the interval $x \in (0, \infty)$.
- Case where $W \geq 2$: the number of positive roots of $f(x) = 0$ cannot be determined.

Continued fraction method(3)

In the case that $W = 1$, the root is included in $(0, u\_b]$, where u_b denotes the upper bound of the positive roots of $f(x) = 0$.

In the case that $W \geq 2$, the interval $(0, \infty)$ should first be divided into two intervals.

This division is performed by $x \rightarrow x + 1$ and $x \rightarrow \frac{1}{x+1}$.

replacement	the interval of the original poly.	the interval the replaced poly.
$x \rightarrow x + 1$	$(1, \infty)$	$(0, \infty)$
$x \rightarrow \frac{1}{x+1}$	$(0, 1)$	$(0, \infty)$

Then, Descartes' rule of signs can be applied to each interval.

Synthetic division and the cost

The replacements needs synthetic division. As an example, the following table shows the calculation of the coefficients of

$$\begin{aligned} g_5(x) &= a_0(x+1)^3 + a_1(x+1)^2 + a_3(x+1) + a_4 \\ &= a_0 x^3 + (3a_0 + a_1)x^2 + (3a_0 + 2a_1 + a_2) + (a_0 + a_1 + a_2 + a_3). \end{aligned}$$

a_0	a_1	a_2	a_3
	a_0	$a_0 + a_1$	$a_0 + a_1 + a_2$
a_0	$a_0 + a_1$	$a_0 + a_1 + a_2$	$a_0 + a_1 + a_2 + a_3$
	a_0	$2a_0 + a_1$	
a_0	$2a_0 + a_1$	$3a_0 + 2a_1 + a_2$	
	a_0		
a_0	$3a_0 + a_1$		

Clearly, the cost is $O(n^2)$, where n is the highest order of the polynomial equation.

Acceleration of the CF method using a lower bound

In order to speed up the CF method, we need an origin shift.

A lower bound l_b of $f(x) = 0$ can be computed in the following procedure:

1. Replace x with $1/x$ in $f(x)$.
2. Compute u_b, which means the **upper bound** of the positive roots of the replaced polynomial equation.
3. Obtain l_b as $l\_b = 1/u\_b$.

The lower bound l_b can be used as the shift amount.

Computation of the upper bound of positive roots(1)

Theorem 2(Akritas, 2006)

Let $f(x)$ be a polynomial with real coefficients, and assume $a_0 > 0$. Let $d(f)$ and $t(f)$ denote its degree and number of terms, respectively.

In addition, assume that $f(x)$ can be reshaped as follows:

$$f(x) = q_1(x) - q_2(x) + \cdots - q_{2m}(x) + g_6(x),$$

where the polynomials $q_i(x)$, $i = 1, \ldots, 2m$, and $g_6(x)$ have only positive coefficients. Moreover, assume that, for $i = 1, 2, \ldots, m$, we obtain

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

and

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

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 $q_{2i}(x)$.

Computation of the upper bound of positive roots(2)

If $t(q_{2i-1}) \geq t(q_{2i})$ for all indices $i = 1, 2, \cdots, m$, then the upper bound of the positive roots of $f(x) = 0$ is defined by

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

for any permutation of the positive coefficients $c_{2i-1,j}$, $j = 1, 2, \cdots, t(q_{2i-1})$. Otherwise, for each of the indices i for which we obtain $t(q_{2i-1}) < t(q_{2i})$, we break up one of the coefficients of $q_{2i-1}(x)$ into $t(q_{2i}) - t(q_{2i-1}) + 1$ parts, so that $t(q_{2i}) = t(q_{2i-1})$. We can then apply the formula defined in Eq. (1).

Computation of the upper bound of positive roots(3)

The sharpness of the upper bound is dependent on pairing.

$3x^3 - 5x^2 + 4x + 7 \to$ creating the pair $\left\{3x^3, -5x^2\right\}$ easily
$3x^3 - 5x^2 - 4x + 7 \to$ not creating the pair immediately
In this case, since

$$3x^3 = \frac{3}{2}x^3 + \frac{3}{2}x^3 = x^3 + 2x^3 = \cdots ,$$

we can create the pair as

$\left\{\frac{3}{2}x^3, -5x^2\right\}, \left\{\frac{3}{2}x^3, -4x\right\}$ or $\left\{x^3, -5x^2\right\}, \left\{2x^3, -4x\right\}$ or $\cdots$.

“Local-max” bound and “first-λ” bound

Using Theorem 2, Akritas et al. proposed the following bounds.

“Local-max” bound

Target polynomial : $x^3 + 10^{100}x^2 - x - 10^{100}$

pairing : $\left\{\frac{10^{100}}{2}x^2, -x\right\}$ and $\left\{\frac{10^{100}}{2^2}x^2, -10^{100}\right\}$

bound : 2

“First-λ” bound

Target polynomial : $x^5 + 2x^4 - 3x^3 + 4x^2 - 5x - 10^{10}$

$$= x^5 + 2x^4 - 3x^3 + 2x^2 + 2x^2 - 5x - 10^{10}$$

pairing : $\left\{x^5, -3x^3\right\}, \left\{2x^4, -5x\right\}$, and $\left\{2x^2, -10^{10}\right\}$

bound : $\sqrt{10^{10}/2} = 50000\sqrt{2}$

New upper bounds(1)

We propose "local-max2" bound, tail-pairing "first-λ" type-I bound, and tail-pairing "first-λ" type-II bound.

"Local-max2" bound

Target polynomial; $x^3 + 10^{100}x^2 - x - 10^{100}$

pairing : $\left\{\frac{10^{100}}{2}x^2, -x\right\}$ and $\left\{\frac{10^{100}}{2}x^2, -10^{100}\right\}$

bound : $\sqrt{2}$

It can be proven that the local-max2 bound is better than or equal to the local-max bound for all polynomials.

cf. "Local-max" bound proposed by Akritas et al.

We pair the terms $\left\{\frac{10^{100}}{2}x^2, -x\right\}$ and $\left\{\frac{10^{100}}{2^2}x^2, -10^{100}\right\}$, and obtain a bound estimate of 2.

New upper bounds(2)

Target polynomial : $x^5 + 2x^4 - 3x^3 + 4x^2 - 5x - 10^{10}$

Tail-pairing "first-λ" type-I bound

pairing : $\{x^5, -3x^3\}, \{2x^4, -10^{10}\}$, and $\{4x^2, -5x\}$

bound : $\sqrt[4]{10^{10}/2} = 100\sqrt[4]{50}$

Tail-pairing "first-λ" type-II bound

pairing : $\{x^5, -10^{10}\}, \{2x^4, -3x^3\}$, and $\{4x^2, -5x\}$

bound : $\sqrt[5]{10^{10}} = 100$

cf. "First-λ" bound proposed by Akritas et al.

We pair the terms $\{x^5, -3x^3\}$, $\{2x^4, -5x\}$, and $\{2x^2, -10^{10}\}$, and obtain a bound estimate of $\sqrt{10^{10}/2} = 50000\sqrt{2}$.

Numerical experiment(1)

To evaluate the effect of the proposed bounds, we implement the CF method with the following bounds:

- FL+LM: ($\mathrm{max}(\mathrm{FL}, \mathrm{LM})$,introduced by Akritas et al.)
- LMQ: local-max quadratic bound (introduced by Akritas et al.)
- **TPFL-I+LM2: ($\mathrm{max}(\mathbf{TPFL\text{-}I}, \mathbf{LM2})$,our proposed bound)**
- **TPFL-II+LM2: ($\mathrm{max}(\mathbf{TPFL\text{-}II}, \mathbf{LM2})$,our proposed bound)**

Note that FL, LM, TPFL, and LM2 denote the first-λ bound, local-max bound, tail-pairing first-λ bound, and local-max2 bound, respectively.

Numerical experiment(2)

As test polynomial equations, the following were used:

- Laguerre: $L_0(x) = 1$, $L_1(x) = 1 - x$, and $L_{n+1}(x) = \frac{1}{n+1}((2n + 1 - x)L_n(x) - nL_{n-1}(x))$
- Chebyshev-I: $T_0(x) = 1$, $T_1(x) = x$, and $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$
- Chebyshev-II: $U_0(x) = 1$, $U_1(x) = 2x$, and $U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)$
- Wilkinson: $W_n(x) = \Pi_{i=1}^{n}(x - i)$
- Mignotte: $M_n(x) = x^n - 2(5x - 1)^2$
- Randomized polynomial

Numerical experiment(3)

Polynomial Class	Degree	Time (s)			
		FL +LM	LMQ	TPFL-I +LM2	TPFL-II +LM2
Laguerre	100	0.01	0.01	0.01	0.01
Laguerre	1000	43.51	48.20	41.77	**36.57**
Laguerre	1500	221.10	242.69	217.21	**189.34**
Laguerre	2000	704.95	755.48	683.57	**617.01**
Chebyshev-I	100	0.01	0.01	0.01	0.01
Chebyshev-I	1000	40.22	41.11	**36.30**	**36.48**
Chebyshev-I	1500	206.87	210.86	**184.45**	**185.61**
Chebyshev-I	2000	650.85	638.67	**590.36**	**590.36**

Chebyshev-II	100	0.01	0.01	0.01	0.01
Chebyshev-II	1000	40.48	40.88	**35.74**	**35.56**
Chebyshev-II	1500	203.53	210.73	**182.73**	**182.67**
Chebyshev-II	2000	652.94	636.42	599.48	**579.28**
Wilkinson	100	0.00	0.00	0.00	0.00
Wilkinson	1000	**4.53**	4.92	**4.52**	**4.54**
Wilkinson	1500	**22.45**	23.82	**22.46**	**22.46**
Wilkinson	2000	**70.46**	73.97	**70.59**	**70.60**
Mignotte	100	0.00	0.00	0.00	0.00
Mignotte	1000	0.04	0.04	0.04	0.04
Mignotte	1500	0.12	0.12	0.12	0.12

Numerical experiment(4)

Execution time for random polynomials defined as

$$f(x) = \prod_{i=0}^{r}(x - x_i)\prod_{j=0}^{s}(x - \alpha_j + i\beta_j)(x - \alpha_j - i\beta_j), -10^9 \le x_i, \alpha_j, \beta_j \le 10^9.$$

Parameters	Degree	Time (s), Avg (Min/Max)	
		FL+LM	LMQ
$s = 40$ $r = 20$	100	0.015(0.01/0.02)	0.0188(0.01/0.03)
$s = 490$ $r = 20$	1000	29.046(19.15/43.61)	30.161(17.47/49.39)
$s = 740$ $r = 20$	1500	135.59(94.78/203.07)	139.06(92.1/211.72)
$s = 990$ $r = 20$	2000	415.37(296.62/645.55)	425.47(270.36/835.35)

Execution time for random polynomials defined as

$$f(x) = \prod_{i=0}^{r}(x - x_i)\prod_{j=0}^{s}(x - \alpha_j + i\beta_j)(x - \alpha_j - i\beta_j), -10^9 \le x_i, \alpha_j, \beta_j \le 10^9.$$

Parameters	Degree	Time (s), Avg (Min/Max)	
		TPFL-I+LM2	TPFL-II+LM2
$s = 40$ $r = 20$	100	0.0145(0.01/0.02)	0.0127(0.01/0.02)
$s = 490$ $r = 20$	1000	27.325(19.05/38.39)	**26.88(17.22/39.38)**
$s = 740$ $r = 20$	1500	128.07(91.69/179.71)	**123.84(86.17/176.16)**
$s = 990$ $r = 20$	2000	384.11(266.41/617.17)	**368.36(271.71/603.31)**

Computing environment

CPU:Intel Core i7 3770K

Mem:32Gbyte, CC:gcc 4.6.3

LIB:GNU Multiple Precision Arithmetic Library, because the CF method needs multiple-precision arithmetic to compute the coefficients in the replaced polynomial equations.

Conclusions

- We have proposed new lower bounds(local-max2 bound and tail-pairing first-λ bound).
- The numerical results show that the average execution time of the CF method with both the local-max2 bound and the tail-pairing first-λ bound is **faster than or nearly equal to** that with the local-max bound, first-λ bound, and local-max quadratic bound proposed by Akritas et al. for all polynomial equations.

Thank you for your kind attention!