A New Method for Real Root Isolation of Univariate Polynomials

Ting Zhang and Bican Xia

Abstract. A new algorithm for real root isolation of univariate polynomials is proposed, which is mainly based on exact interval arithmetic and bisection method. Although exact interval arithmetic is usually supposed to be inefficient, our algorithm is surprisingly fast because the termination condition of our algorithm is different from those of existing algorithms which are mostly based on Descartes' rule of signs or Vincent's theorem and we decrease the times of Taylor Shifts by exact interval arithmetic.

Keywords. Real root isolation, interval arithmetic, interval Newton operator, bisection method, continued fractions method.

Extended Abstract

Real root isolation of univariate polynomials with integer coefficients plays a significant role in many algorithms concerning computational real algebra and real algebraic geometry. In many computer algebra systems (CAS), one can find implementations of algorithms based on different principles for real root isolation. The `realroot` function in Maple and the `RealRootIntervals` function in Mathematica are such examples.

Many algorithms for real root isolation have been proposed in literature, see, for example, [4, 6, 5, 1, 9, 12, 2], which can be classified into two kinds of methods: bisection and non-bisection methods. For example, algorithms proposed in [4, 6, 5, 9, 12] are essentially based on bisection strategy, taking use of different principles to rule out intervals with no roots and to pick out intervals containing exactly one root and the algorithm proposed in [1, 2] is based on Vincent's theorem which does not bisect intervals.

This work is supported by NKBRPC-2004CB318003, NKBRPC-2005CB321902 and NSFC-60573007 in China.

The idea of bisection method is natural and easy to understand. Suppose B is the root bound of $f(x)$, a univariate polynomial with integral coefficients, and we have an effective rule $\mathcal{M}$ to determine the number of roots of $f(x)$ in an interval, then we may bisect $(-B, B)$ (or $(0, B)$) repeatedly and apply $\mathcal{M}$ each time to rule out intervals not containing roots. We will call such rules as *root-rules* in this abstract. Some examples of root-rules are Sturm's theorem, Budan-Fourier's theorem and Descartes' rule of signs. It is well known that bisection algorithms based on Descartes's rule of signs are most efficient in general [6, 8, 12]. All the known algorithms for real root isolation based on Descartes' rule of signs and the bisection strategy are described in [12] in a unified framework, and a new algorithm REL is presented in [12], which is proved to be very efficient and can works with huge polynomials, including orthogonal polynomials of degree 1000 and more.

The CF method proposed in [1] does not perform bisection. It is directly based on Vincent's theorem and is more tricky. A comparative study of the CF method and the REL method through timings on some special and randomly generated polynomials is reported in [2].

One main computation involved in algorithms based on Descartes' rule of signs and bisection is computing Taylor shifts. This is also true for the CF algorithm. Because computing Taylor shifts may be very costly especially for polynomials of high degrees, fast algorithms for Taylor shifts [7] are employed in the REL and CF methods. Moreover, some special technics are implemented in REL for fast Taylor shifts [11] while CF computes lower bounds of positive roots of polynomials to decrease the number of Taylor shifts [2].

In this extended abstract, we briefly describe a new method for real root isolation, which is also a bisection method using however a modified *interval Newton iteration* as root-rule.

The set of all intervals is denoted by $\mathrm{I}(\mathbb{R})$. Let f be an arithmetic expression of a polynomial in $\mathbb{R}[x_1, ..., x_n]$. We replace all operands of f as intervals and replace all operations of f as interval operations and denote the result by F. Then, $F : \mathrm{I}(\mathbb{R})^n \to \mathrm{I}(\mathbb{R})$ is called an *interval evaluation* of f. Let $f(x)$ be a polynomial in $\mathbb{R}[x]$ and X an interval, *the interval Newton operator* [3] is defined as

$$N(X) = \mathrm{m}(X) - \frac{f(\mathrm{m}(X))}{F'(X)}$$

where F' is an interval evaluation of f' and $\mathrm{m}(X)$ is the midpoint of X.

The above defined Newton operator satisfies the following properties [3, 10].

1. If $x^* \in X$ is a zero of $f(x)$, $x^* \in N(X)$.
2. If $X \cap N(X) = \emptyset$, $f(x)$ does not have zeros in X.
3. If $N(X) \subset X$, $f(x)$ has zeros in X.
4. If $N(X)$ is contained in the interior of X, $f(x)$ has a unique zero in X.

Let an initial interval $X = X_0$ be given, setting $X = N(X) \cap X$, repeating this *interval Newton iteration* for all resulting intervals and discarding empty sets, we will obtain a set of intervals possibly containing zeros of $f(x)$ on X_0. Naturally,

one can propose an algorithm for real root isolation using interval Newton iteration and exact interval arithmetic. However, direct use of the original interval Newton iteration with exact interval arithmetic leads to inefficiency. One main reason is that, for a polynomial $f(x)$ and an interval X, $F(X)$ (the interval evaluation of $f(x)$ on X) is often much more wider than $f(X)$. So, those intervals not containing zeros cannot be recognized quickly and thus a great number of redundant iterations occur. We found that the interval evaluation of a polynomial $g(x)$ on $(0,1)$ is easy to be computed and has little difference to $g((0,1))$. Therefore, we modify the Newton iteration and combine the interval Newton operator with the bisection method. As a result, the efficiency has been improved dramatically.

The sketch of our algorithm can be described as follows. For a squarefree polynomial $f(x) \in \mathbb{Z}[x]$, we only consider its positive roots.

Algorithm: `TRealroot`
Input: a squarefree polynomial $f(x) \in \mathbb{Z}[x]$.
Output: L, a list of isolating intervals for the positive zeros of $f(x)$.

1. Let $B \geq 1$ be a positive root bound of $f(x)$. Set $g(x) = f(Bx)$.
2. By the subalgorithm described below, compute the list of isolating intervals of real zeros of $g(x)$ in $(0,1)$ and denote it by L'.
3. Replace every interval (a_i, b_i) in L' by (Ba_i, Bb_i) and call the resulting list L.

In the following description of subalgorithm `Kiteflying`, we use a data structure $[h, u, v, r, s, k]$ where h is a polynomial and u, v, r, s, k are integers. The zeros of $h(x)$ in (u, v) correspond to the zeros of $g(x)$ in $(r/2^k, s/2^k)$ one to one. In the subalgorithm, $(u, v) = (0, 1)$ and $s = r + 1$.

Subalgorithm: `Kiteflying`
Input: a squarefree polynomial $g(x) \in \mathbb{Z}[x]$ with all positive zeros in $(0,1)$ and $g(0)g(1) \neq 0$.
Output: $OutL$, a list of isolating intervals for the zeros of $g(x)$ in $(0,1)$.

1. Set $OutL = [\ ]; L = [\ [g, 0, 1, 0, 1, 0]\ ]$. Repeat the following steps until L is empty.
2. Fetch the first element of L, say $[h, u, v, r, s, k]$, and delete it in L.
3. If $h((u+v)/2) = 0$, include $[(r+s)/2^{k+1}, (r+s)/2^{k+1}]$ in $OutL$ and replace $h(x)$ by $h(x)/(x - (u+v)/2)$. Include $[2^m h(x/2), 2u, u+v, 2r, r+s, k+1]$ and $[2^m h(x/2), u+v, 2v, r+s, 2s, k+1]$ in L as the first two elements where m is the degree of $h(x)$, then goto Step 2.
4. If $H'((u, v))$ (the interval evaluation of $h'(x)$ on (u, v)) does not contain 0, check the signs of $h(u)$ and $h(v)$. If $h(u)h(v) < 0$, include $(r/2^k, s/2^k)$ in $OutL$.
5. If $H'((u, v))$ contains 0, apply interval Newton operator to h and (u, v), *i.e.*, set
$$N(h, u, v) = (u+v)/2 - \frac{h((u+v)/2)}{H'((u, v))}.$$

5.1 If $N(h,u,v)\cap((u+v)/2,v)$ is not empty, set $h_2(x)=2^m h((x+v)/2)$ where m is the degree of h and include $[h_2,u,v,r+s,2s,k+1]$ in L as the first element.

5.2 If $N(h,u,v)\cap(u,(u+v)/2)$ is not empty, set $h_1(x)=2^m h((x+u)/2)$ where m is the degree of h and include $[h_1,u,v,2r,r+s,k+1]$ in L as the first element.

Remark 1. The above algorithm is much more efficient than the original interval Newton iteration. However, as stated above, Taylor shifts in Step 5.1 and Step 5.2 may be very costly. So, we use some technics to modify the subalgorithm `Kiteflying` for decreasing the number of Taylor shifts. For example, we may compute an upper bound for the difference of $H'((u,v))$ and $h'((u,v))$. If the bound is less than an empirical value given before, we do not perform Taylor shift in Step 5.1 (or Step 5.2) but include $[2^m h(x/2),u+v,2v,r+s,2s,k+1]$ (or $[2^m h(x/2),2u,u+v,2r,r+s,k+1]$) in L as the first element. Moreover, we may use Budan-Fourier's theorem to accelerate the subalgorithm. We would like to explain the details in our full paper later.

Remark 2. We do not analyze the complexity of our algorithm in this abstract. Thus, it is difficult to compare our algorithm with others theoretically. Instead, we have implemented our algorithm TRealroot by Maple and done experiments on many examples in literature. We hope the timings can illustrate the behavior of our algorithm.

Randomly generated polynomials

coefficients (digits)	terms	degrees	timings	loops	shifts	roots
10	10	100	0.184	17.40	3.8	3.40
10	10	500	0.513	32.80	0.6	2.40
10	10	1000	0.187	49.20	0	3.60
10	10	2000	0.279	34.40	0	2.20
10	100	100	0.785	27.40	7.8	3.60
10	500	500	4.281	52.40	1.4	3.60
10	1000	1000	11.380	61.80	0	4.80
10	2000	2000	47.718	77.40	0	6.00
1000	10	100	0.191	23.80	2.4	2.60
1000	10	500	0.153	24.00	0.2	2.80
1000	10	1000	0.166	39.40	0	3.20
1000	10	2000	0.350	46.80	0	3.20
1000	100	100	0.837	26.80	7.4	2.60
1000	500	500	4.713	52.00	1	3.60
1000	1000	1000	13.316	65.80	0	4.80
1000	2000	2000	54.697	80.20	0	5.20

The three tables report our experiments. The columns entitled "loops" gives the numbers of loops executed (*i.e.,* the numbers of bisections) and the "shifts" columns show the numbers of Taylor shifts performed. We think these two numbers

are the most essential data for bisection based methods. The "roots" columns are the average numbers of roots.

Randomly generated monic polynomials

coefficients (digits)	terms	degrees	timings	loops	shifts	roots
10	10	100	0.240	22.40	4.2	4.00
10	10	500	0.094	30.20	0	2.80
10	10	1000	0.159	35.80	0	3.00
10	10	2000	0.431	60.80	0	4.80
10	100	100	0.719	24.80	7.8	4.00
10	500	500	5.288	60.40	2.2	5.20
10	1000	1000	9.734	59.40	0	4.40
10	2000	2000	46.646	82.60	0	7.60
1000	10	100	0.372	25.80	4.2	3.00
1000	10	500	1.528	39.40	2.4	4.20
1000	10	1000	3.516	48.80	0.8	3.20
1000	10	2000	0.378	42.00	0	3.40
1000	100	100	0.822	31.00	8.2	5.20
1000	200	200	2.250	37.80	7.2	3.60
1000	500	500	5.594	58.20	1.4	5.60
1000	1000	1000	15.343	73.80	0	6.00

Polynomials	degrees	timings	loops	shifts	roots
ChebyShevT	100	1.656	154	60	50
ChebyShevT	500	149.299	977	249	250
ChebyShevT	1000	1467.589	2365	437	500
ChebyShevT	1200	3082.505	3189	534	600
ChebyShevU	100	1.578	148	59	50
ChebyShevU	500	140.171	920	240	250
ChebyShevU	1000	1461.933	2272	446	500
ChebyShevU	1200	2902.301	3270	508	600
Laguerre	100	2.891	223	104	100
Laguerre	500	417.743	1258	493	500
Laguerre	900	4281.166	2768	874	900
Laguerre	1000	6652.326	3071	958	1000
Mignotte	100	0.25	242	1	4
Mignotte	300	2.921	708	0	4
Mignotte	400	8.204	940	0	4
Mignotte	600	37.811	1405	0	4
Wilkinson	100	2.407	205	96	100
Wilkinson	500	402.113	1266	468	500
Wilkinson	800	2636.266	2289	761	800
Wilkinson	900	4378.753	2763	853	900
Wilkinson	1000	7130.022	3158	946	1000

All examples are computed on a PC (Pentium IV/3.0GHz CPU, 1 G memory, Windows XP) with Maple 10 and the timings are obtained by using the `time` function in Maple. For randomly generated polynomials, the timings are average timings on 5 polynomials [1]. For ChebyShev polynomials (which are generated by the functions `ChebyShevT` and `ChebyShevU` in Maple), only positive roots are isolated. We take $a = 5$ for the Mignotte polynomials $(x^n - 2(ax-1)^2)$ in the third table.

We try to compare our timings with those of the REL [12] and CF [2] methods but such comparison may be not suitable since the three implementations are realized by different languages and the timings are obtained on different machines (AMD Athlon 1GHz CPU and 1.5GB memory for REL and AMD Athlon 850MHz CPU and 256M memory for CF). The following table gives such comparison on some special polynomials.

polynomials	degree	CF	REL	TRealroot
Chebyshev	1000	2172	1183	1467
Laguerre	900	3790	2116	4281
Laguerre	1000	6210	3055	6652
Wilkinson	1000	256	840	7130
Mignotte	300	0.12	33	3
Mignotte	400	0.22	122	8
Mignotte	600	0.54	428	38

For randomly generated polynomials, TRealroot is much faster than CF and we do not have corresponding data of REL. Timings of TRealroot increase smoothly as the degrees of polynomials increase. Generally speaking, the current implementation of TRealroot is slower than CF and REL for polynomials with low degrees but is faster for those with high degrees. Moreover, TRealroot is super efficient for sparse polynomials with large degrees. For example, our program isolates the three real roots of

$$f(x) = -10x^{93925} + 62x^{82660} - 82x^{76886} + 80x^{69549} - 44x^{68273} + 71x^{55578} \\ -17x^{53739} - 75x^{30731} - 10x^{22679} - 7$$

within 318 seconds.

References

[1] A. G. Akritas, A. V. Bocharov, A. W. Strzeboński, *Implementation of real root isolation algorithms in Mathematica.* In: Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (Interval94), St. Petersburg, Russia, March 7C10, 23–27, 1994.

[2] A. G. Akritas, A. W. Strzeboński, *A Comparative Study of Two Real Root Isolation Methods.* Nonlinear Analysis: Modelling and Control **10** (2005), 297–304.

[1] In order to compare our results with others, we use only 5 randomly generated polynomials as in literature.

[3] G. Alefeld, J. Herzberger, *Introduction to interval analysis.* Academic Press, 1983.

[4] 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., 272–275, 1976.

[5] G. E. Collins, J. R. Johnson, *Quantifier elimination and the sign variation method for real root isolation.* In: Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation, ACM Press, 264–271, 1989.

[6] G. E. Collins, R. Loos, *Real Zeros of Polynomials.* In: Computer Algebra: Symbolic and Algebraic Computation (Buchberger, B., Collins, G. E., Loos, R., eds.), Springer-Verlag, Wien New York, 83–94, 1983.

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

[8] J. R. Johnson, *Algorithms for polynomial real root isolation.* Technical Report OSU-CISRC-8/91-TR21, Ohio State University, 1991.

[9] J. R. Johnson, W. Krandick, *Polynomial real root isolation using approximate arithmetic.* In: Proceedings of ISSAC'97, Maui, Hawaii, 225–232, 1997.

[10] G. Mayer, *Epsilon-inflation in verification algorithms.* J. of Computational and Applied Mathematics **60** (1995), 147–169.

[11] F. Rouillier, personal communication, 2006.

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

Ting Zhang
Accounting Centre of China Aviation
Beijing, China
e-mail: ztfriend@tom.com

Bican Xia
LMAM & School of Mathematical Sciences
Peking University
Beijing 100871, China
e-mail: xbc@math.pku.edu.cn