JOSAI MATHEMATICAL MONOGRAPHS
Vol. **2** *(2000) pp. 69 - 76*

Finding Real Roots of Polynomials

–Toward High Quality & High Reliability Computing–

Teluhiko HILANO*

Kanagawa Institue of Technology

Abstract

We propose a simple and fast implimentation to find real zeros of polynomials of integer coefficients up to required places. If the placeses is not sufficient to seperate the real zeros, their accuracy will be enlarged automatically.

1 Introduction

There are many method to find real zeros of polynomials of one variable with integer coefficients. In [9], they classify them 29 kinds including methods to get complex zeros and list up many papers according to the principle:

> we present a comprehensive bibliography on roots of polynomials, covering (hopefully) most published work between the "Dawn of History" and 1994 .

In this short paper, we propose a program to give the zeros of polynomials of one variable with precision in advance, using recent Computer Algebraic systems. If the precision given at start is not sufficient, we change precisions of zeros automatically.

One may suppose that an algorithm using computer algebraic systems to solve such problems is according to Sturm's theorem. But, it takes long time to calculate so-called Sturm sequence. Our algorithm is much simpler but faster when the degree of polynomial is large.

2 Algorithm

Our algorithm proposed here is following:

*hilano@gen.kanagawa-it.ac.jp

1. Let $f(x)$ be a polynomial to be solved. We make a polynomial sequence $f_1(x), f_2(x), \ldots f_k(x)$ by following procedure. [2, Proof of Proposition 2.86]
   (a) Let $f_1(x) = f(x)/\gcd(f(x), f'(x))$.
   (b) If $f_j(x)$ is defined, we put $f_{j+1}(x) = f_j'(x)/\gcd(f_j'(x), f_j''(x))$.
   (c) If the degree of $f_j(x)$ is greater 1, repeat the previous step.
2. For $j = k-1, k-2, \ldots, 1$, we get real zeros of $f_{j+1}(x) = 0$ using the zeros of $f_{j+1}(x) = 0$ in order.
3. Determine the multiplicity of zeros of $f(x)$.
4. Refine the zeros upto required accuracy.

The Step 2 is based on the following fact:

Lemma 1
The above polynomial sequence $f_1(x), f_2(x), \ldots f_k(x)$ has the following properties.

- *Every $f_j(x)$ is squre free.*
- *Equations $f_j(x) = 0$ and $f_{j+1}(x) = 0$ have no common zeros.*
- *For $j = 1, 2, \ldots, k-1$, let α, β be consecutive zeros of $f_{j+1}(x)$. There exist at most one real zero of $f_j(x) = 0$ in (α, β). If there exists real zeros of $f_j(x) = 0$, the sign of $f_j(x)$ must change near the zeros.*

proof) For a polynomial $g(x)$, let

$$g(x) = g_1(x)g_2^2(x)\cdots g_k^k(x)$$

be the square free decomposition, i.e. every $g_j(x)(j = 1, 2, \ldots k)$ is squarefree. Then

$$g(x)/\gcd(g(x), g'(x)) = g_1(x)g_2(x)\cdots g_k(x)$$

This implies $g(x)/\gcd(g(x), g'(x))$ is squere free and the same zeros as $g(x)$. Let α, β be consecutive real zeros of the derivative of a squre free polynomial $g^*(x)$. Because $g^*(x)$ is monotone in $[\alpha, \beta]$, $g^*(x)$ has at most one real zero and must change its sign near the zero, if exists.

Applying this fact to above polynomial sequence, Lemma follows.

3 The Implimentation

3.1 Fundamental principle of Implimentation

We impliment above algrithm to Risa/Asir under following principles.

- In Computer Algebraic System, rational arithmetic run slower than integer arithmetic. See, [4, p.428, the second paragraph] or [8, p.68, explanation of `ptozp`]. Hence, we use only integer arithmetic. To calcurate some number in n decimal places, we use $f\left(\frac{x}{10^N}\right)$ insted of the given polynomial $f(x)$.
- Real zeros of $f_j(x)$ are given of the intervals which include them and do not intersects each other(exclude their end points).

3.2 Details of separation of zeros

The method to separate the real zeros of $f_j(x)$ is as follows:

1. Let S_L be the sign of the left end point of $f_j(x)$ in the interval in consideration.
2. Let $[\alpha, \beta]$ be the interval in which contains only one real zeros $f_{j+1}(x)$. If the sign $f_{j+1}(\alpha)$ or $f_{j+1}(\beta)$ is different to S_L, there exists only one zero of $f_j(x)$.
3. If above condition fails, we check the sign of $f_{j+1}(\alpha + \beta 2)$ and the length of the interval to exist the zero of $f_j(x)$ is halved in integer. Then we repeat the previous step.
4. If the length of the above interval cannnot be halved in integer, we calcurate the range of value of $f_j(x)$ by interval analysis, using Honers' rule. If the range(interval number) does not contain 0, we have the sign of $f_j(x)$ at the above interval and conclude not to exist the zero of $f_j(x)$. [1)]
5. Othewise, let $g(x) = f_j(\alpha + x)$ and determine the sign of $g(x)$ in the interval $[0, \beta - \alpha]$ by the interval analysis as the previous step.($\beta - \alpha = 1$ in this situation.)
6. If the previous steps fails, we replace $f(x)$ to $10^n f_j(x/10)$, where n is the degree of $f_j(x)$.

The step 5 is necesary. If the step is ommited, there exist polynomials which it takes much computation time to determine the zeros.

3.3 Algorithm to refine zeros upto required places

We adopt Newton Method in integer to refine zeros upto required places. In the process of Newton method, the sequence of approximation of the zero is not always monotone and goes outside of the given interval. So we modify Newton method when the value goes outside or one end point of the interval.

Now, we asuume that there exists only one simple zero of $f(x) = 0$ in an interval $[\alpha, \beta]$ (and α, β are integers.) We use only integer arithmetic appearing in the following steps.

[1)] About interval analysis, see [4, p.240–241, 4.2.2 C.].

1. We set $\gamma = \alpha$.
2. When $f'(\gamma) = 0$, we cannot apply to Newton method. So, we set $\gamma^* = \dfrac{\alpha+\beta}{2}$.
3. Otherwise, we set $\gamma^* = \gamma - \dfrac{f(\gamma)}{f'(\gamma)}$.
4. When $\gamma^* < \alpha$ or $\gamma^* > \beta$, Newton method fails and we set $\gamma^* = \dfrac{\alpha+\beta}{2}$.
5. In the case $\gamma^* = \alpha$, we set $\gamma^* = \alpha+1$ because we can expect the calucuration converges.
6. In the case $\gamma^* = \beta$, we set $\gamma^* = \beta - 1$ by the same reason of 5.
7. If the sign of $f(\gamma^*)$ and $f(\alpha)$ are same, we set $\alpha = \gamma^*$, otherwise $\beta = \gamma^*$.
8. New we set $\gamma = \gamma^*$.
9. Repeat step 2 while $\beta - \alpha > 1$.

Since the length of interval decrease in these steps, the algorithm stops. Steps 5,6 are important because the sequence of the approximation of zeros in Newton method is monotone very near to the exact zeros. See, [3, p.79 Theorem 4.8].

4 Examples and time of execution

The functions we impliment on Risa/Asir are follows:

```
findrealzero(Poly, Low, High, Prec)
findrealzeroall(Poly, Prec)
```

`Poly` is the polinomial to be solved. The arguments `Low` and `High` are the left and right end points of given interval to find zeros in, respectively. In the case function `findrealzeroall`, it is not necessary to give an inteval to be find zeros in because the bound of the existence of roots are determined automatically. The variable `Prec` is the required decimal places of zeros. For example,

```
findrealzero(x^2-2, -4, 4, 10)
```

requests to find the all real zeros of $x^2 - 2$ in $[-4, 4]$ upto 10 decimal places, i.e get $\pm\sqrt{2}$ upto 10 decimal places. This produces [2)]

```
[[[-14142135624,10],[-14142135623,10],1],
  [[14142135623,10], [14142135624,10],1]]
```

[2)] We add carrage returns and spaces to understand the results easily.

This means that there exist the zeros of x^2-2 with multiplicity 1 in the two intervals

$$[-14142135624\times10^{-10}, -14142135623\times10^{-10}], [14142135623\times10^{-10}, 14142135624\times10^{-10}]$$

respectively. Hence we have

$$1.4142135623 \leq \sqrt{2} \leq 1.4142135624.$$

The next example is that auatomatic refinment is done when there exist very closer zeros than required precision.

Example 1 $_{20}$
Let $P(x) = \prod_{k=1}(x-k)$. *The derivative coefficient at its zero* $x = 20$ *is*

$$19! = 121645100408832000 \asymp 1.2 \times 10^{17}.$$

Thus, the zero of $P(x) = 1$ *near to* $x = 20$ *is as close as by the inverse of this value.*
The calucuration `findrealzero(P*(P-1),-1,30,10,0);`
gives (in part)

```
[[100000000000007,13],[100000000000008,13],1],
[[109999999999992,13],[109999999999993,13],1],
[[109999999999999,13],[110000000000000,13],1],
[[119999999999999,13],[120000000000000,13],1],
[[120000000000006,13],[120000000000007,13],1],
[[129999999999995,13],[129999999999996,13],1],
[[129999999999999,13],[130000000000000,13],1],
[[139999999999999,13],[140000000000000,13],1],
[[140000000000002,13],[140000000000003,13],1],
[[1499999999999990,14],[1499999999999991,14],1],
[[1499999999999999,14],[1500000000000000,14],1],
[[1599999999999999,14],[1600000000000000,14],1],
[[1600000000000003,14],[1600000000000004,14],1],
[[16999999999999992,15],[16999999999999993,15],1],
[[16999999999999999,15],[17000000000000000,15],1],
[[17999999999999999,15],[18000000000000000,15],1],
[[18000000000000001,15],[18000000000000002,15],1],
[[189999999999999998,16],[189999999999999999,16],1],
[[189999999999999999,16],[190000000000000000,16],1],
[[1999999999999999999,18],[2000000000000000000,18],1],
[[2000000000000000008,18],[2000000000000000009,18],1]]
```

Refinemets of zeros are according to the differences of close two zeros which are different in there positions.

Example 2

The next table shows CPU times and total times within garbage collections to separate, to refine zeros and both when we solve the zeros of Legendre polynomials in interval $[-1, 1]$

$$P_n(x)\frac{1}{n!}\frac{d^n}{dx^n}(x^2-1)^n (n = 100, 105, \ldots, 300)$$

in 30 decimal places using Risa/Asir Version 990618 on FreeBSD 3.2/ K6-III 400MHz.

K6 III 400MHzMemory 128MB FreeBSD 3.2

degree	Separation(sec)		Refinement(sec)		Total(sec)		heap	pari
	CPU	Total	CPU	Total	CPU	Total	(byte)	2.0.16
100	5.753	7.9537	16.813	23.924	22.580	31.905	2576384	15.984
105	6.668	9.3699	15.989	21.147	22.671	30.533	3223552	18.703
110	7.647	10.432	18.750	25.403	26.412	35.852	4030464	22.719
115	8.903	11.687	19.914	26.524	28.833	38.228	4030464	24.117
120	10.187	13.491	30.193	40.696	40.396	54.207	4030464	28.055
125	11.762	15.855	24.563	34.012	36.342	49.887	4030464	31.656
130	13.416	18.270	29.826	41.623	43.260	59.914	4030464	41.688
135	14.864	20.522	31.012	44.202	45.896	64.747	4030464	39.758
140	16.801	23.647	41.981	56.030	58.803	79.704	5042176	52.492
145	19.033	25.679	38.375	52.067	57.430	77.773	5042176	52.055
150	21.213	29.215	45.207	58.320	66.443	87.563	6303744	138.195
155	23.659	31.109	46.040	60.546	69.725	91.684	6303744	60.789
160	26.403	35.260	73.308	98.112	99.737	133.403	6303744	166.648
165	28.798	38.947	53.862	73.116	82.688	112.096	6303744	167.742
170	31.828	43.452	63.139	87.088	94.997	130.576	6303744	199.281
175	35.033	48.333	60.486	84.209	95.548	132.577	6303744	194.758
180	38.711	53.806	80.401	113.398	119.143	167.242	6303744	118.070
185	42.591	56.327	70.793	93.084	113.418	149.451	7880704	243.102
190	46.549	63.108	83.483	114.005	130.066	177.154	7880704	274.992
195	50.846	69.695	86.703	119.465	137.584	189.203	7880704	287.109
200	55.112	71.992	122.844	157.651	177.993	229.687	9854976	355.156
205	59.195	78.766	93.054	123.161	152.289	201.973	9854976	336.258
210	64.423	86.264	110.529	147.020	174.993	233.333	9854976	381.711
215	70.855	91.042	113.179	142.092	184.081	233.218	12320768	394.703
220	77.020	100.277	142.511	182.509	219.576	282.837	12320768	482.320
225	83.906	110.367	121.764	157.654	205.715	268.076	12320768	451.672

K6 III 400MHzMemory 128MB FreeBSD 3.2

degree	Separation(sec)		Refinement(sec)		Total(sec)		heap	pari
	CPU	Total	CPU	Total	CPU	Total	(byte)	2.0.16
230	90.774	120.226	140.585	183.465	231.405	303.748	12320768	518.477
235	97.707	130.588	136.807	179.878	234.563	310.525	12320768	498.328
240	104.934	141.952	219.674	293.103	324.665	435.156	12320768	603.562
245	112.930	153.892	154.410	208.260	267.399	362.255	12320768	591.367
250	120.052	156.156	177.156	223.943	297.268	380.205	15405056	767.344
255	128.513	169.738	170.574	219.989	299.142	389.793	15405056	688.797
260	135.767	181.850	221.160	288.579	356.986	470.500	15405056	874.406
265	144.638	194.380	190.311	249.805	335.012	444.259	15405056	775.875
270	153.574	208.222	222.974	294.809	376.613	503.107	15405056	981.609
275	163.567	212.022	224.422	280.462	388.057	492.563	19259392	896.828
280	173.759	227.804	306.393	389.279	480.221	617.163	19259392	—
285	182.913	240.988	243.073	309.908	426.057	550.978	19259392	—
290	195.376	259.892	275.773	355.273	471.222	615.250	19259392	—
295	206.216	276.012	273.851	353.026	480.139	629.126	19259392	—
300	219.311	296.347	337.184	443.798	556.573	740.237	19259392	—

5 Comparison to other sysems

There are very few implimentation to give (real) zeros of polynomials of very high order. A function `polroots` in the package pari-2.0 is an example of which algorithm is depend on Algorithm of Shönhage with modification by Gourdon[1]. It can find complex zeros at all. The previous table contains the execution times of this function. It requires more memory(pari stack) and more times when the degree increases.

The function `DSolve` in Mathematica[5] gives complex zeros of polinomial which has no complex zeros(for example $(x-1)(x-2)\cdots(x-20)$) So, we do not compare to the execution time.

Acknowlegement

The author express his great thanks to T. Saito(Sophia Univ.), T. Takeshima(Fujitsu Laboratory) and Y. Kondoh(Takuma National Coll. of Tech.) Especially, Takeshma informed him the polynomial of degree 103 in [7, p.18–19] and the execution time to solve it by Sturm Method, which helps to improve the algorithm and implimentation of this paper.

References

[1] Batut, C., Belabas, K., Bernardi, D., Cohen H., Olivier, M, *User's Guide to PARI-GP*, at `ftp://megrez.math.u-bordeaux.fr/pub/pari/`

[2] Becker, T., Weispfenning, V.,*Gröbner Bases, A Computational Approach to Commutative Algebra*, Graduate Texts in Math. 141, Springer, 1993

[3] Henrici, P., *Elements of Numerical Analysis*, John Wiley & Sons, 1963

[4] Knuth, D. E., *The Art of Computer Programming*, Vol. 2 *Seminumerical Algorithms*, 3rd Ed., Addison Welsley, 1998

[5] Wolfram, S., *The Mathematica book*, 3rd ed., Wolfram Media, Cambridge Univ. Press, 1996

[6] Masao Iri, Quality of Calcuration(In Japanese), bit **28**(1996), 9,p.52–55

[7] Tomokatsu SAito, Yuji Kondoh,, Yoshihiko Miyoshi, Tak Takeshima, Displaying Real Solution of Mathematical Equations, Suusikishori, Vol.6, No. 2, pp.2-21 (1998)

[8] Tomokatsu Saito, Tak Takeshima, Teluhiko Hilano: Computer alegraic System prodeced in Japan: — Guide Bokk on Risa/Asir, SEG shuppan(1998)

[9] J. M. McNamee *A bibliography on roots of polynomials* `http://www.elsevier.co.jp/homepage/sac/cam/mcnamee/`