Polynomial Real Root Isolation Using Descarte's Rule of Signs *

George E. Collins
and
Alkiviadis G. Akritas
Computer Sciences Department, University of Wisconsin
1210 W. Dayton Street, Madison, Wisconsin 53706

Abstract

Uspensky's 1948 book on the theory of equations presents an algorithm, based on Descartes' rule of signs, for isolating the real roots of a squarefree polynomial with real coefficients. Programmed in SAC-1 and applied to several classes of polynomials with integer coefficients, Uspensky's method proves to be a strong competitor of the recently discovered algorithm of Collins and Loos. It is shown, however, that it's maximum computing time is exponential in the coefficient length. This motivates a modification of the Uspensky algorithm which is quadratic in the coefficient length and which also performs well in the practical test cases.

1. Introduction

A polynomial real root isolation algorithm is an algorithm which, given a polynomial A(x) with real coefficients, computes a sequence of disjoint intervals each containing exactly one real root of A, and together containing all real roots of A. Heindel, in a 1970 Ph.D. thesis, [4], (see [5] for a compact presentation of the main results), provided the first implementation using exact arithmetic within a computer algebra system of such an algorithm, together with a thorough analysis of its computing time. Heindel's algorithm was based on the methodical application of Sturm's theorem.

Just recently, Collins and Loos, [3], have given a similar treatment of a new polynomial real root isolation algorithm, which is based on recursive application of Rolle's theorem together with a certain tangent construction. They show that this new algorithm is significantly superior to the Sturm algorithm in most practical cases.

In the following we present another new algorithm, this one based on the use of Descarte's rule of signs together with certain linear-fractional transformations. We find our algorithm to be a strong competitor of the Collins-Loos algorithm; applied to the same examples which were used in [3], it is from two to eight times faster in most cases. Furthermore, we are able to establish a better theoretical upper bound for its computing time; for a squarefree integral polynomial A of degree n with $|A|_1 = d$, we obtain a bound of $n^6 L(d)^2$ in place of the bound $n^7 L(d)^3$ for the previous algorithms. Here $|\Sigma_{i=0}^{n} a_i x^i|_1 = \Sigma_{i=0}^{n} |a_i|$ and L(d) is the length of the integer d in radix representation with unspecified integer radix $\beta \geq 2$.

*Research supported by National Science Foundation Grant DCR74-13278.

The new algorithm to be described in this paper is dedicated to J. V. Uspensky, to whose book, [6], its discovery is directly attributable. Our algorithm is a modification of one which Uspensky (in 1948) claims is most efficient. We quote from page 127 of his book: "The theorem of Rolle and the rule of signs, though they may often help in separating the roots of an equation, do not provide by themselves a complete and exhaustive solution of this important problem. The application of Rolle's theorem for this purpose requires a knowledge of the real roots of the derivative, which is, for the most part, lacking. The rule of signs is a rather weak proposition and, applied to an equation the nature of whose roots is not known, does not give the exact number of positive (or negative) roots except when the number of variations is zero or one. But exactly these two particular cases, when combined with a remarkable theorem published by Vincent in 1836 and hinted at earlier by Fourier, supply the most efficient method, not only to determine the exact number of positive and negative roots but also to effect their separation, in case the proposed equation has no multiple real roots."

We programmed Uspensky's algorithm and found that indeed, on the average, it was faster than the Sturm algorithm. But, as we will show, its maximum computing time is exponential in L(d) and, for the same reason, the observed computing times are occasionally quite unsatisfactory. Our modification of Uspensky's algorithm, which will be described explicitly below, may be regarded an application of the algorithm design principle of balance (see [1], page 65). The modification tends to increase the computing time when the roots are well separated, but decreases the computing time when some roots are very close.

2. Uspensky's Algorithm

Let $b = (b_1,\ldots,b_r)$ be a sequence of real numbers. Let $b' = (b'_1,\ldots,b'_s)$ be the subsequence

of non-zero elements of b. Then the number of variations in b, var(b), is the number of i's, $1 \le i < r$, such that $\text{sign}(b'_i) = -\text{sign}(b'_{i+1})$. If $A(x) = \Sigma_{i=0}^{n} a_i x^i$ is a real polynomial, then we identify A with its sequence of coefficients $a = (a_0,\ldots,a_n)$ and write var(A). Descartes' rule of signs is a theorem which asserts that the number of positive real roots (multiplicities counted) of a real polynomial A is equal to var(A) - 2k for some non-negative integer k. The exact number of positive real roots is determined just in case var(A) = 0 or var(A) = 1. The theorem can be applied to negative roots by replacing A(x) by A(-x), and zero is a root of A just in case $a_0 = 0$.

If var(A) ≥ 2 and $A(0) = a_0 \neq 0$, let $A_1(x) = A(x+1)$ and $A_2(x) = (x+1)^n A(1/(x+1))$. If $\alpha_1,\ldots,\alpha_m$ are the roots of A greater than or equal to one, then $\alpha_1-1,\ldots,\alpha_m-1$ are the roots of A_1 greater than or equal to zero. If $\alpha_1,\ldots,\alpha_m$ are the roots of A between zero and one, then $\alpha_1^{-1}-1,\ldots,\alpha_m^{-1}-1$ are the positive real roots of A_2. Let $A_1(x) = x^h \bar{A}_1(x)$ where $\bar{A}_1(0) \neq 0$. If h > 0, then 1 is a root of A and {1} = [1,1] is an isolating interval for A. If $\text{var}(\bar{A}_1) = 0$, then A has no roots in $(1,\infty)$, while if $\text{var}(\bar{A}_1) = 1$ then $(1,\infty)$ is an isolating interval. If $\text{var}(\bar{A}_1) > 1$, then $\bar{A}_1$ is subjected to the same two substitutions, x + 1 and 1/(x+1). Similarly, if $\text{var}(A_2) = 0$, then A has no roots in (0,1); if $\text{var}(A_2) = 1$, then (0,1) is an isolating interval for A; and if $\text{var}(A_2) > 1$, then A_2 is subjected to the same two substitutions.

This is Uspensky's algorithm. Its computations can be represented in the form of a binary tree in which there is associated with each node a polynomial and a transformation. Each path from a node to a successor corresponds to a substitution x + 1 or 1/(x+1), which substitution is applied to both the polynomial and the transformation. Polynomials at terminal nodes have at most one variation; the associated transformations of those with one variation, when applied to $(0,\infty)$, produce isolating intervals.

Uspensky applies a theorem of Vincent ([7], 1836) to show that, when applied to a squarefree polynomial, the algorithm always terminates, i.e., the binary tree is finite. Vincent's theorem asserts that if a squarefree polynomial is successively transformed by successive substitutions $a_i + \frac{1}{x}$, where the a_i are positive integers, one eventually obtains a polynomial with at most one variation. In fact, Uspensky's strengthened version of Vincent's theorem bounds the number of substitutions 1/(x+1) as a function of n = deg(A) and the minimum root separation of A.

If m is the number of substitutions 1/(x+1) in any path from the root of the tree to any node, then the Vincent-Uspensky theorem shows that m is dominated by $L(n) + L([\lambda^{-1}])$ where $\lambda = \text{sep}(A)$ is the minimum root separation of A. But it is known, [2], that $L([\lambda^{-1}]) \le nL(d)$ where $d = |A|_1$, so $m \lesssim nL(d)$. However, the number of substitutions x + 1 is not dominated by any power of L(d), as may be verified with the example $A(x) = \{ax-(a-1)\}\{(a+1)x-a\} = (a^2+a)x^2 - (2a^2-1)x + (a^2-a)$, with a a positive integer, a > 2. Here the maximum value of m is a - 2, while $d = 4a^2 - 1$. It is possible to construct a similar example for every n ≥ 2.

In the implementation of Uspensky's algorithm, one may replace the (partially-constructed) binary tree by a list of pairs corresponding to its terminal nodes, each pair consisting of the associated polynomial and transformation. Moreover, the list may contain pairs only for those nodes for which the polynomial has at least two variations. The main loop of the algorithm then removes a pair from the list and produces two new pairs corresponding to the two substitutions. The new pairs with two or more variations are then returned to the pair list while the new pairs with one variation contribute their intervals to an output list. The loop is repeated until the pair list becomes empty. Some observed times for application of this algorithm are presented in Section 4.

3. The Modified Uspensky Algorithm

Table 1 shows the intervals which may occur at the first few levels of the binary tree in Uspensky's algorithm. At level 4, the lengths of the finite intervals fluctuate between 1/15 and 1; at higher levels the disparity in lengths grows rapidly. Clearly there is a lack of optimal balance in this algorithm.

Table 1. Interval's in Uspensky's Algorithm

Level 0:	$(0,\infty)$
Level 1:	$(0,1),(1,\infty)$
Level 2:	$(0,1/2),(1/2,1),(1,2),(2,\infty)$
Level 3:	$(0,1/3),(1/3,1/2),(1/2,2/3),(2/3,1),$ $(1,3/2),(3/2,2),(2,3),(3,\infty)$
Level 4:	$(0,1/4),(1/4,1/3),(1/3,2/5),2/5,1/2),$ $(1/2,3/5),(3/5,2/3),(2/3,3/4),(3/4,1),$ $(1,4/3),(4/3,3/2),(3/2,5/3),(5/3,2),$ $(2,5/2),(5/2,3),(3,4),(4,\infty)$

Let b be a root bound for A, that is, $|\alpha| < b$ for every root α of A. We compute a root bound as in [3], so that $b = 2^k$ for some (positive, negative or zero) ineger k. Let $A^*(x) = A(2^k x)$ if $k \ge 0$, $A^*(x) = 2^{-kn} A(2^k x)$ if k < 0. Now A^* is an integral polynomial whose positive real roots are all in the interval (0,1); the roots of A^* in (0,1) correspond to the positive real roots of A. We associate with A^* the interval (0,b). If $\text{var}(A^*) = 1$, then (0,b) is an isolating interval for A; and if $\text{var}(A^*) = 0$, then A has no positive real roots. If $\text{var}(A^*) \ge 2$, then we put the pair $(A^*,(0,b))$ in a pair list, P. Each pair in P consists of a polynomial A_i and an interval,

(a_i,b_i), such that the roots of A in (a_i,b_i) correspond to the roots of A_i in (0,1) and the intervals (a_i,b_i) are disjoint. The roots of A_i in (0,1) correspond to the positive real roots of $A_i^*(x) = (x+1)^n A_i(1/(x+1))$ and so we assume that $\mathrm{var}(A_i^*) \geq 2$ for each i. Let $A_i'(x) = 2^n A_i(x/2)$ and $A_i''(x) = A_i'(x+1) = 2^n A_i(\frac{x+1}{2})$. The roots of A_i' in (0,1) correspond to the roots of A_i in (0,1/2) and hence to the roots of A in (a_i,c_i) where c_i is the midpoint $(a_i+b_i)/2$. Similarly, the roots of A_i'' in (0,1) correspond to those of A in (c_i,b_i). And c_i is a root of A just in case $A_i''(0) = 0$. Thus if $\mathrm{var}(A_i'^*) = 1$, then (a_i,c_i) is an isolating interval for A, and if $\mathrm{var}(A_i'^*) \geq 2$ then $(A_i',(a_i,c_i)$ is put in the list P. Similarly for $A_i'',A_i''^*$ and (c_i,b_i), but with due regard for the case $A_i''(0) = 0$ as in Uspensky's algorithm. This bisection process is repeated until the list P becomes empty. This is the modified Uspensky algorithm.

Vincent's theorem can be modified to obtain a proof that the modified algorithm terminates. The main result is the following.

Theorem 1. Let A(x) be a real polynomial of degree n. Let $\epsilon_n = (1+1/n)^{1/(n-1)} - 1$. Let α be a real root of A with $0 \leq a < \alpha < b$. Let C be the circle with center (a+b)/2 and radius $(\epsilon_n^{-1}+1/2)(b-a)$. Let $A_1(x) = A((b-a)x+a)$. If all roots, real and complex, of A other than α are outside C, then $\mathrm{var}(A_1^*) = 1$.

One also uses the known result that if all roots of a real polynomial A have negative real parts, then var(A) = 0. Combining Theorem 1 with the observation that $\epsilon_n^{-1} = O(n^2)$ and the bound of [2] for minimum root separation, one obtains the following result.

Theorem 2. Let A(x) be an integral square-free polynomial of degree $n \geq 1$ and $|A|_1 = d$. The computing time of the modified Uspensky real root isolation algorithm (using classical arithmetic algorithms) is dominated by $n^6 L(d)^2$.

4. Empirical Comparisons

In the following we present several tables showing the observed computing times, for several classes of polynomials, for the Collins-Loos differentiation algorithm (D), the Uspensky algorithm (U), and the modified Uspensky algorithm (M). The polynomials used in these tests are the same as those used in [3], and the times for Algorithm D are taken from [3]. All times are given in seconds and were obtained using SAC-1 on the University of Wisconsin UNIVAC 1110 computer.

We describe the several classes of polynomials very briefly; full descriptions are given in [3]. Tables 2 and 3 pertain to polynomials with random coefficients which are b bits long. n denotes the degree in all tables. The polynomials of Table 4 are resultants of random bivariate polynomials; each entry is the average for three resultants with coefficients from 10 to 30 decimal digits in length. Table 5 is for products of random linear polynomials, with the coefficients of the product truncated to a maximum of 33 bits. Table 6 pertains to the Chebyshev polynomials $T_n(x)$.

Each table has an additional column labeled B. This is the number of bisections performed in Algorithm M. Most of the work of Algorithm M consists of performing translations $A^*(x) = A(x+1)$, and the number of translations performed is 3B. It appears that the average value of B is approximately linear in n, with the constant of proportionality dependent on the class of polynomials. Using this observation (or conjecture) one can argue that the average computing time of Algorithm M is perhaps approximately codominant with $n^4 + n^3 L(d)$, which may be compared with the average time $n^4 + n^2 L(d)^2$ conjectured for Algorithm D in [3]. In the practical cases observed, both algorithms appear to behave like $n^3 L(d)$, but with Algorithm M having a somewhat smaller constant factor.

TABLE 2
Random Polynomials, b=33

n	B	D	U	M
5	2	0.76	0.07	0.17
10	$3\frac{2}{3}$	3.77	0.21	0.66
15	4	8.06	0.32	1.44
20	$4\frac{1}{3}$	18.4	0.68	2.71
25	7	30.2	0.74	7.86

TABLE 3
Random Polynomials, n=15

b	B	D	U	M
33	3	7.9	0.32	0.95
66	5	12.1	0.55	1.89
99	4	10.4	0.43	1.61
132	5	14.2	1.16	2.25

TABLE 4
Random Resultants

n	B	D	U	M
8	7	3.79	0.31	1.01
18	11	21.0	1.76	6.73
32	14	83.4	4.73	33.4

TABLE 5
Random Products

n	B	D	U	M
5	9	1.42	0.14	0.50
10	15	6.83	0.96	2.23
15	32	17.9	4.32	12.5
20	25	38.0	6.77	17.6
25	37	64.0	21.5	40.4

TABLE 6
Chebyshev Polynomials

n	B	D	U	M
5	8	0.78	0.16	0.36
10	14	3.40	1.52	2.02
15	18	8.98	5.36	5.07
20	26	18.6	20.6	17.0
25	30	35.2	47.4	31.6
30	36		118.9	56.0

5. Conclusions

One may be tempted to conclude from the above empirical comparisons that the unmodified Uspensky algorithm is usually faster than the other known algorithms and that therefore these other algorithms do not merit continued use or investigation. Such a simple state of affairs, however, does not exist.

For example, the Collins-Loos algorithm computes isolating intervals not only for A(x) but also for all of its derivatives and, as observed in [3], there are important applications where isolating intervals for several or all derivatives are needed. Another point to observe is that root isolation is often used as a prelude to root refinement or approximation; hence a slower root isolation method which produces smaller isolating intervals may not be as bad as it would appear from the type of data presented above.

The modified Uspensky method is faster than the unmodified Uspensky method in some cases but slower in others. This appears to be another case where the fastest algorithm for "difficult" problems is somewhat slower for easier problems. However, "difficulty" here depends on root distribution as well as degree and coefficient size and since one will generally have little, if any, advance information about root distribution, the selection of the most appropriate algorithm will be difficult. For the present, at least, a computer algebra system should ideally supply several algorithms, allowing the user to exercise judgement in selection of the most appropriate algorithm for the problem at hand. In various contexts the preferred algorithm may be the Collins-Loos algorithm, the Uspensky algorithm or the modified Uspensky algorithm. The Sturm algorithm is generally much slower than the others but, as the data in [3] show, it is slightly faster than the others for Chebyshev polynomials. Certainly there are many questions deserving further study.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[2] G. E. Collins and E. Horowitz, The Minimum Root Separation of a Polynomial, Math. Comp., Vol. 28, No. 126 (April 1974), 589-597.

[3] G. E. Collins and R. Loos, Polynomial Real Root Isolation by Differentiation, Proc. 1976 ACM Symposium on Symbolic and Algebraic Calculation, 1976.

[4] L. E. Heindel, Algorithms for Exact Polynomial Root Calculation, Ph.D. Thesis, University of Wisconsin, 1970.

[5] L. E. Heindel, Integer Aritimetic Algorithms for Polynomial Real Zero Determination, Jour. ACM, Vol. 18, No. 4 (Oct. 1971), 533-548.

[6] J. V. Uspensky, Theory of Equations, McGraw-Hill, 1948.

[7] M. Vinent, Sur la Résolution des Équations Numeriques, Jour. de Mathematiques Pures et Appliquees, Vol. 1 (1836), 341-372.