SOLVING SPARSE DIOPHANTINE EQUATIONS IN ONE VARIABLE

ADAM STRZEBOŃSKI

ABSTRACT. We discuss practical methods for computing integer roots of sparse univariate polynomials. The best currently known algorithm solving the problem was given in [1]. We investigate how a small change in the algorithm, namely using a different root isolation method, would influence the computation time. It turns out that the change makes the algorithm significantly faster in practice.

1. INTRODUCTION

Let
$$f = a_1 t^{e_1} + \ldots + a_k t^{e_k}$$
where $e_1 > \ldots > e_k$, and $a_i \in Z \setminus \{0\}$, for $1 \le i \le k$, and let us denote
$$minexp(f) := e_k$$

Definition 1.1. *The sparse derivative sequence of f, is the sequence $f_1, \ldots, f_k$ defined by*
$$f_1 := f/t^{minexp(f)}$$
and
$$f_{i+1} := f_i'/t^{minexp(f_i')}$$
for $1 \le i \le k-1$.

The algorithm described in [1] finds integer roots of f by isolating the real roots of polynomials of the sparse derivative sequence of f, taken in the reverse order, up to intervals $(u, u+1)$ or $[u, u]$, with $u \in Z$. The method is based on the fact that, by Rolle's theorem, a polynomial can have at most one root between two consecutive roots of its derivative. An alternative approach is to isolate the roots of f directly, using a "sparse variant" of Fourier's theorem.

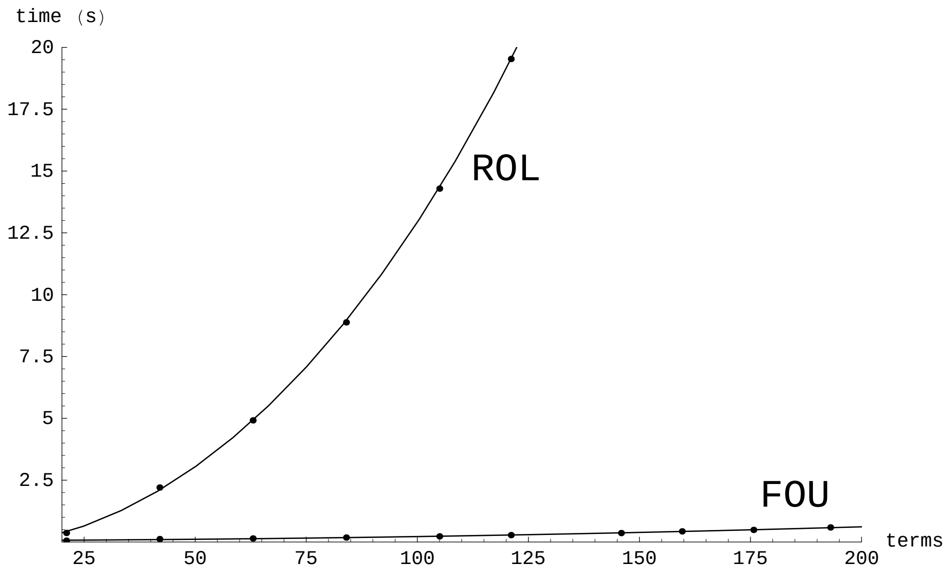
Figure 1 shows the dependence of the running time of the integer root finding algorithm on the number of terms of the polynomial, for randomly generated polynomials of degree 1000 with 20 roots. The curve marked ROL (for Rolle) corresponds to the root isolation method used in [1], the curve marked FOU (for Fourier) corresponds to root isolation based on Fourier's theorem. The method based on Fourier's theorem is clearly faster here, and the difference increases with the number of terms.

In the next section we investigate the reason of the observed timing difference. We state the variant of Fourier's theorem we use for root isolation and estimate complexities of the two root isolation methods.

Remark 1.2. *The integer root finding algorithm described in* [1]*, as well as the algorithm described in* [2] *restricted to finding integer roots, use an exponent gap theorem (Proposition 2 of* [1]*, Proposition 2.3 of* [2]*) which allows to reduce the size of the polynomial*

Date: November 21, 2005.

FIGURE 1. Timings for polynomials of degree 1000 with 20 roots.



for very sparse polynomials with relatively small coefficients. Root isolation methods discussed in this paper affect the final stage of the algorithm, when the exponent gap theorem no longer applies. The examples used here have been chosen from the sparsity and coefficient size range for which the exponent gap theorem does not apply.

2. COMPARISON OF INTEGER ROOT ISOLATION METHODS

Let $f_1,\ldots,f_k : I \longrightarrow R$ be a sequence of differentiable functions defined in an open interval $I \subseteq R$, and such that for all $1 \le i \le k-1$, and for all $x \in I$

$$sign(f_{i+1}(x)) = sign(f_i'(x)) \tag{1}$$

and

$$f_k(x) \neq 0$$

For $x \in I$, and $1 \le i \le j \le k$, let $sgc_{i,j}(x)$ denote the number of sign changes in the sequence $f_i(x),\ldots,f_j(x)$ with terms equal to zero removed, and let $sgc(x) := sgc_{1,k}(x)$. The following is a simple variant Fourier's root counting theorem.

Theorem 2.1. *For any $a \in I$*

$$\lim_{x\to a^+} sgc(x) = sgc(a)$$

and

$$\lim_{x\to a^-} sgc(x) = sgc(a) + r + 2s$$

where $r \in Z_+$, $s \in Z_+ \cup \{0\}$, and

$$f_1(a) = \ldots = f_r(a) = 0,\ f_{r+1}(a) \neq 0$$

Moreover, $s = 0$ unless there is $t > r+1$ such that $f_t(a) = 0$.

The proof is straightforward.

Corollary 2.2. *Let $f_1, \dots, f_k$ be the sparse derivative sequence of a polynomial f, let $0 \le a < b$, and let $1 \le p \le k$, be such that $sgc_{p,k}(a) = sgc_{p,k}(b)$. Then*

(1) *f_p has a constant nonzero sign on $(a,b]$.*
(2) *$sgc_{p,k}(c) = sgc_{p,k}(a)$ for any $a \le c \le b$.*
(3) *$sgc_{1,p}(a) - sgc_{1,p}(b) = r + 2s$, where r is the number of roots of f in $(a,b]$, counted with multiplicities, and s is a nonnegative integer. In particular, if $sgc_{1,p}(a) = sgc_{1,p}(b)$ then f has no roots in $(a,b]$, and if $sgc_{1,p}(a) - sgc_{1,p}(b) = 1$ then f has exactly one root in $(a,b]$ and $sign(f(a)) \neq sign(f(b))$.*

Corollary 2.3. *A polynomial with $k > 0$ nonzero terms has at most $k-1$ positive roots and at most $k-1$ negative roots, and hence at most $2k-1$ real roots.*

A detailed description of the root isolation algorithm using Fourier's theorem can be found in [3].

Let ROL be the algorithm finding integer roots using Rolle's theorem, as described in [1], and let FOU be a variant of the algorithm using Fourier's theorem. Let $f \in Z[t] \setminus \{0\}$, let k be the number of nonzero terms in f, and let M be a bound for absolute value of integer roots of f, and let $f_1 = f, \dots, f_k$ be the sparse derivative sequence of f.

ROL isolates all real roots of f_{k-m+1}, for $m = 1, \dots, k$, up to unit length intervals. This in the worst case requires $(2m-1)logM$ interval bisections, for $m = 1, \dots, k$. Hence, the total number of interval bisections performed by ROL is at most $k^2 logM$. On the other hand, FOU isolates at most $2k-2$ changes in value of *sgc* up to unit length intervals. Hence the total number of interval bisections performed by FOU is bounded by $2(k-1)logM$.

This of course is not an entirely fair comparison, since ROL needs to compute a single polynomial sign at each bisection, while FOU might need to compute k polynomial signs. However, Corollary 2.2 allows to reduce the number of sign computations needed. If $sgc(a) - sgc(b) = 1$ then any further bisections of $(a,b]$ need to compute only the sign of f, and if $sgc_{p,k}(a) = sgc_{p,k}(b)$, which in particular must be the case if none of $f_p, \dots, f_k$ has roots in $(a,b]$, then any further bisections of $(a,b]$ do not need to compute the sign of $f_p, \dots, f_k$.

In the following section we describe experiments measuring the dependence of the number of sign computations needed on the number of polynomial terms.

3. Experimental results

Both variants of the algorithm have been implemented in C as a part of *Mathematica* kernel. The computations have been done on a 1.8 GHz Pentium M laptop computer with 1740 MB of RAM available. The results given are averages for sets of 10 randomly generated polynomials. Columns marked Terms and Roots give, respectively, the average number of terms in the polynomial and the average number of integer roots. Time is given in seconds. Columns marked Sign Tests show the average number of polynomial sign tests performed during root isolation.

The complexity of finding integer roots highly depends on the number and relative location of the roots of polynomials in the sparse derivative sequence. To measure the dependence of the number s_{FOU} and s_{ROL} of polynomial sign tests, used by FOU and ROL, on the number k of polynomial terms we kept the degree of the polynomial, the number of integer roots and the method of generating factors with integer roots fixed. We varied k by multiplying by a random sparse polynomial with a variable number of terms. It never contributed integer roots other than zero, since a random sparse polynomial is very unlikely to

TABLE 1. Polynomials of degree 1000 with 20 linear factors

Poly	Terms	Roots	Time FOU	Time ROL	Sign Tests FOU	Sign Tests ROL
$P_1(1)$	21	21	0.053	0.367	2296	22916
$P_1(2)$	42	20	0.118	2.203	3000	79525
$P_1(3)$	63	20	0.144	4.918	3769	153546
$P_1(4)$	84	20	0.180	8.881	4384	244985
$P_1(5)$	105	20	0.228	14.29	4905	353900
$P_1(6)$	121.1	20	0.279	19.53	5367	451030
$P_1(7)$	145.9	20	0.362	29.60	5809	616541
$P_1(8)$	159.6	20	0.430	36.57	6227	720909
$P_1(9)$	175.7	20	0.489	45.16	6434	851127
$P_1(10)$	193	20	0.589	57.36	6888	1004228

TABLE 2. Polynomials of degree 10000 with 3 three-term factors with one root each

Poly	Terms	Roots	Time FOU	Time ROL	st# FOU	st# ROL
$P_2(1)$	20	4	0.030	0.055	113	260
$P_2(2)$	40	3	0.046	0.246	223	841
$P_2(4)$	79.7	3	0.156	1.731	428	3054
$P_2(8)$	159.1	3	0.366	7.272	792	10532
$P_2(16)$	315.3	3	1.046	33.05	1260	31067
$P_2(32)$	618.4	3	3.200	147.9	2168	108430

have integer roots, unless it has one term. We used the following two classes of randomly generated sparse polynomials.

(1) Random sparse polynomials of degree 1000 obtained by multiplying an n-term polynomial and 20 monic linear factors.

$$P_1(n) = (a_1 t^{980} + \sum_{i=2}^{n-1} a_i t^{e_i} + a_n) \prod_{j=1}^{20} (t - r_j)$$

where a_i and r_j are randomly generated 100-bit integers, r_j are all distinct, and $0 < e_i < 980$ are randomly generated distinct exponents.

(2) Random sparse polynomials of degree 10000 obtained by multiplying an n-term polynomial and three degree 1001 three-term polynomials with one integer root each.

$$\begin{aligned} f_j(t) &= a_j t^{1001} + b_j t^{e_j} \\ P_2(n) &= (a_1 t^{6997} + \sum_{i=2}^{n-1} a_i t^{e_i} + a_n) \prod_{j=1}^{3} (f_j(t) - f_j(r_j)) \end{aligned}$$

where a_j, b_j and r_j are randomly generated 10-bit integers, and $0 < e_i < 6997$ are randomly generated exponents.

TABLE 3. Polynomials with maximal number of integer roots

Poly	Terms	Roots	Time FOU	Time ROL	Sign Tests FOU	Sign Tests ROL
$P_3(1)$	2	3	3.312	3.312	199	201
$P_3(2)$	3	5	7.857	9.418	406	596
$P_3(3)$	4	7	13.57	19.40	613	1192
$P_3(4)$	5	9	21.00	35.52	819	1985
$P_3(5)$	6	11	27.35	53.70	1027	2989
$P_3(6)$	7	13	34.60	76.53	1236	4212
$P_3(7)$	8	15	44.69	110.9	1453	5639
$P_3(8)$	9	17	55.05	151.9	1673	7284
$P_3(9)$	10	19	62.33	186.4	1891	9161
$P_3(10)$	11	21	74.98	251.9	2113	11238

Using least-squares linear fit to find the dependence of $log(s)$ on $log(k)$ we found that, for polynomials $P_1(n)$, $s_{FOU} \sim 469k^{0.506}$ and $s_{ROL} \sim 139k^{1.687}$, and for polynomials $P_2(n)$, $s_{FOU} \sim 9.4k^{0.856}$ and $s_{ROL} \sim 1.35k^{1.757}$.

The next experiment we used polynomials of degree 5041 which have the maximal possible (Corollary 2.3) number of real roots for a polynomial with k terms.

$$P_3(n) = t\prod_{j=1}^{n}(t^{5040/n} - r_j^{5040/n})$$

As might be expected, since the number of roots grows with k, in this example the complexity grows much faster with k previous two. However, like in the previous two experiments, the growth exponent is significantly higher for ROL. We obtained $s_{FOU} \sim 88k^{1.35}$ and $s_{ROL} \sim 44.3k^{2.33}$.

4. CONCLUSIONS

Using root isolation based on Fourier's theorem significantly improves practical performance of the algorithm described in [1].

REFERENCES

[1] F. Cucker, P. Koiran, S. Smale, "A Polynomial Time Algorithm for Diophantine Equations in One Variable", J. Symbolic Comp., 27 (1999), 21-29.
[2] H. W. Lenstra, Jr., "Finding Small Degree Factors of Lacunary Polynomials", in K. Gyory, H. Iwaniec, J. Urbanowicz (eds.), Number Theory in Progress, Walter de Gruyter 1999, 267-276.
[3] A. Strzebonski, "An Improved Algorithm for Diophantine Equations in One Variable", ACA 2002 conference talk, http://members.wolfram.com/adams.

WOLFRAM RESEARCH INC., 100 TRADE CENTRE DRIVE, CHAMPAIGN, IL 61820, U.S.A.
E-mail address: `adams@wolfram.com`