Fast Absolute Irreducibility Testing via Newton Polytopes*

Shuhong Gao† and Alan G.B. Lauder‡

Abstract

We present a polytope method to test irreducibility of random sparse polynomials over an arbitrary field, not necessarily computable. For example, for polynomials with 1000 random terms in 10 variables each with degree at most 10, more than 85% of them can be recognized to be irreducible in half a second. This test does not recognize all irreducible polynomials though, so it should be used as a pretest before a more general slower algorithm is applied.

1 Introduction

In previous papers [2, 3], we considered a connection between multivariate polynomials and convex polytopes. It was shown that indecomposable integral polytopes lead to absolutely irreducible polynomials. We also gave a heuristic algorithm for testing indecomposability of integral polytopes, and thus a fast method for testing absolute irreducibility of multivariate polynomials. The purpose of the current paper is to show by computer experiments the effectiveness of our method, particularly for sparse polynomials.

Let $K[X_1, \ldots, X_n]$ be a polynomial ring in n variables over an arbitrary field K. A polynomial of degree bound by d in each variable in this ring will have a maximum of $(d+1)^n$ terms, and we shall call a polynomial with $\Omega((d+1)^n)$ non-zero terms *dense*. Intuitively speaking, "sparse" polynomials are those which have far fewer non-zero terms than this maximum. We define a more precise notion of sparseness for polynomials which is motivated in part by the range of applicability of our algorithm. We shall say that a polynomial in this ring with degree bound by d in each variable is *sparse* if the number of non-zero terms is $\mathcal{O}(nd)$.

*The first author was supported in part by the National Science Foundation (NSF) under Grant DMS0302549, the National Security Agency (NSA) under Grant MDA904-02-1-0067, the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research (ONR) under Grant N00014-00-1-0565. The second author gratefully acknowledges the support of the Marr Educational Trust and Wolfson College, Oxford.

†Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-0975, USA. E-mail: sgao@math.clemson.edu.

‡Mathematical Institute, Oxford University, Oxford OX1 3LB, U.K. E-mail: lauder@maths.ox.ac.uk.

The experiments detailed in Section 5 and Appendix A indicate that our method is extremely effective for sparse polynomials. In particular, our experiments reveal that the algorithm has a very high success rate for such polynomials, and has a running time which is at worst cubic in the input size. However, there are polynomials for which our algorithm will fail, such as dense polynomials. Indeed our method is not intended to be infallible. It is known that most polynomials are absolutely irreducible, but the existing general methods for showing this are very slow. Our contribution is in providing a fast method to decide irreducibility of a special class of polynomials, namely a large fraction of sparse polynomials as we define them. In practice, our algorithm may serve as a pretest before a slower general algorithm is applied.

Remarkably, our method does not involve computations in the field K, and thus works for any field, not necessarily computable. Two other observations on our algorithm, which are related to this and distinguish it from previous methods, seem worth making. Firstly, the actually values of the coefficients of the polynomial do not matter, all that is important is which terms have non-zero coefficients. Thus our method actually allows one to show absolute irreducibility of families of polynomials rather than single polynomials. This may be of interest in areas such as deformation theory. For example, given complex polynomials f and g, one may wish to show that all polynomials in a family $f + tg$ where t is a non-zero complex number, are absolutely irreducible. The hypersurfaces defined by such polynomials are considered deformations of that defined by f itself. Using our method one will very often be able to quickly do this.

The second observation is that once a polynomial over a particular field has been shown to be absolutely irreducible using our method, one knows that it will remain absolutely irreducible "over almost any field". This feature seems of particular relevance in number theory. For suppose we have a polynomial f with integer coefficients which has been shown to be absolutely irreducible using our method. Then given any prime p one immediately knows, subject to mild and easily checked conditions relating p and the coefficients of f, that the polynomial $f \bmod (p)$ remains absolutely irreducible over the field with p elements. Thus our method in fact proves a particularly strong form of absolute irreducibility.

Note that absolute irreducibility of polynomials is fundamental in algebra and geometry, and in computer algebra systems. Many good algorithms have been designed for testing irreducibility, see for examples [1, 4, 6, 7, 9, 10, 11, 12] and the literature given there. It should be remarked that our method is totally different from any of the methods in the literature.

Our paper is organised as follows. In the next section, we review the connection between polynomials and polytopes, and recall a projection lemma that is essential for our algorithm. In Section 3, we present our algorithm and give an analysis of its running time and space requirement. In Section 4, we describe our models of random polynomials and polytopes, and Section 5 and Appendix A contain details of computer experiments along with some observations on the range of parameters over which our method is effective.

2 Polynomials and Newton polytopes

Given a multivariate polynomial in n variables over a field K one may associate with it a convex polytope in $\mathbb{R}^n$ called its *Newton polytope*. This is done in the following way. Let $f \in K[X_1, X_2, \ldots, X_n]$ and define the *support set* $S_f \subseteq \mathbb{R}^n$ to be the set of all vectors $(z_1, z_2, \ldots, z_n)$ which occur as the exponent vector of some non-zero term $aX_1^{z_1}X_2^{z_2}\ldots X_n^{z_n}$ in f, where $a \in K$. The *Newton polytope* of f, denoted N_f, is defined to be the convex hull of S_f. Observe that all of the vertices of the polytope N_f have integer coordinates; we call polytopes with this property *integral*.

We shall now define a notion of addition of polytopes which reflects that of multiplication of polynomials in a certain sense. Given any two integral polytopes Q and R we define their *Minkowski sum* by

$$Q + R := \{q + r \mid q \in Q, r \in R\}.$$

It is not difficult to show that the Minkowksi sum of two integral polytopes Q and R is once again an integral polytope, P say. We call Q and R summands of P and call the identity $P = Q + R$ an *integral decomposition* of P, or simply a *decomposition*. If either Q or R consists of a single point then we call the decomposition trivial. Note that all polytopes have trivial decompositions obtained by adding a suitable translate of the polytope itself to any integral point. If a polytope possesses no non-trivial decompositions we shall it *integrally indecomposable*, or simply *indecomposable*.

We have the following proposition which is central to the algorithm we discuss.

Proposition 1 (Ostrowski 1921[13]) *Let $f, g, h \in K[X_1, X_2, \ldots, X_n]$ and N_f, N_g, N_h be their respective Newton polytopes in $\mathbb{R}^n$. If $f = gh$ then $N_f = N_g + N_h$.*

We refer the reader to [2] for a simple proof of this result. As an immediate corollary one obtains

Corollary 2 *Let $f \in K[X_1, X_2, \ldots, X_n]$. If N_f is indecomposable and f is not divisible by any nonconstant monomial, then f is absolutely irreducible over K.*

Here *absolute irreducibility* means that the polynomial f is irreducible over the algebraic closure of K.

Thus one method of detecting absolutely irreducibility of polynomials is to check whether their Newton polytopes are indecomposable. One can determine whether polygons are indecomposable quite easily, but working directly with polytopes in higher dimensions appears rather more difficult. We circumvent this problem by taking random projections of high dimensional polytopes down to two dimensions. The next result from [3] is then of great use.

Lemma 3 (Projection Lemma) *Let S be a set of integral points in $\mathbb{R}^n$ and A a $2 \times n$ matrix with integer entries. Define $A(S) = \{As \mid s \in S\}$ (a point in $\mathbb{R}^n$ is viewed as a column vector). Suppose that the polygon conv$(A(S))$ is indecomposable, and moreover, each vertex of the polygon has exactly one pre-image under A in the set S. Then the polytope conv(S) is indecomposable.*

Thus to detect indecomposability of the convex hull of a set of integral points in high dimension, one does not necessarily need to compute their convex hull and work with polytopes in the high dimension. It is often sufficient, as we shall see from the implementation details, to take random projections down to the plane and work solely with polygon computations.

3 The Algorithm

We now present an outline of the algorithm we shall consider, more details may be found in [3].

Algorithm 4 Absolute Irreducibility Test
Input: $f \in K[X_1, X_2, \ldots, X_n]$ with no nonconstant monomial factors.
Output: "Absolutely Irreducible" or no output.

Step 0: Let S_f denote the set of exponent vectors of non-zero terms in f. Choose positive integers b and e. Let $M(b)$ denote the set of all $2 \times n$ matrices with integer entries bound in absolute value by b. Repeat Steps 1–3 up to e times.

Step 1 (Random projection): Select a matrix A uniformly at random from $M(b)$ and compute the set of points in $\mathbb{R}^2$

$$A(S_f) := \{As \mid s \in S_f\}.$$

Step 2 (Convex hull): Compute the convex hull of $A(S_f)$ and check that each vertex of $\mathrm{conv}(A(S_f))$ has only one preimage in S_f under the projection A. If this condition is not met, return to Step 1.

Step 3 (Polygon indecomposability test): Test whether the polygon $\mathrm{conv}(A(S_f))$ is integrally indecomposable. If it is then output "Absolutely Irreducible" and halt the algorithm. If not, then return to Step 1.

Theorem 5 *Algorithm 4 halts within $\mathcal{O}(e((nbd)^3 + t(t+n))\log^2(nbd))$ binary operations for a polynomial in n variables, with t non-zero terms and degree at most d in each variable. The space requirement is $\mathcal{O}((nbd)^2 \log(nbd))$ bits. Moreover, if the output is "Absolutely Irreducible" then the input polynomial is absolutely irreducible.*

Proof: The final statement on the output follows from Corollary 2 and Lemma 3 in Section 3.

Before presenting a short analysis of the complexity of the algorithm we must give some details of the subroutines required.

Step 2 may be done using any standard planar convex hull algorithm. In our implementation we use a naive algorithm which has running time $\mathcal{O}(t\ell)$ for t points whose convex hull has ℓ edges. The worst-case time is thus $\mathcal{O}(t^2)$ operations on integers, although $\mathcal{O}(t\log(t))$ algorithms exist [8, pages 361-375].

Step 3 is performed using Algorithm 15 from [3]. For a polygon with ℓ integral points on the boundary and a integral points in the interior this has running time $\mathcal{O}(a\ell)$ integer operations and space requirement $\mathcal{O}(a)$. Note that determining whether a polygon is integrally decomposable is NP-complete (see Proposition 14 in [3]) and the Algorithm used to perform Step 4 is in fact only "pseudo-polynomial time" [5]; however, this is good enough for our purpose.

We now prove the running time and space requirement: The running time of the projection in Step 1 is $\mathcal{O}(tn)$ arithmetic operations with integers bound by nbd. Note that all t of the 2 dimensional points thus obtained have coordinates bound in absolute value by nbd. Hence a naive implementation of the convex hull algorithm in Step 2 requires $\mathcal{O}(t^2)$ arithmetic operations with integers bound by nbd. Similarly, in Step 3 one must test indecomposability of a polygon which lies in a square of area $\mathcal{O}((nbd)^2)$ and, consequently, has no more than $\mathcal{O}(nbd)$ edges. Thus this has running time $\mathcal{O}((nbd)^3)$ arithmetic operations, and space requirement $\mathcal{O}((nbd)^2)$ integers.

The total running time of one execution of Steps 1-3 of the algorithm is in the worst-case $\mathcal{O}((nbd)^3 + t^2 + tn)$ integer operations with integers bound by nbd. Thus the overall running time is $O(e((nbd)^3 + t(t+n))\log^2(nbd))$ bit operations. The space requirement is dominated by the polygon decomposability algorithm used in Step 4, which is $\mathcal{O}((nbd)^2 \log(nbd))$ bits. This completes the proof of Theorem 5

Recall that we called an n-variable polynomial with degree bound by d in each variable "sparse" if it had $\mathcal{O}(nd)$ non-zero terms. Algorithm 4 has a worst-case running time of $\mathcal{O}((nd)^3)$ integer operations for such polynomials, if we assume that the integers b and e are constant.

We shall demonstrate the effectiveness of our algorithm in practice by selecting random sets of points and showing that in many cases we can quickly show that their convex hull is indecomposable. We define precisely what is meant by "random" in the next section, after which we give implementation details.

4 Random polytopes and polynomials

For a positive integer d, define $\mathbb{Z}(d) = \{z \in \mathbb{Z} \mid 0 \le z \le d\}$, and for any natural number n let

$$\mathbb{Z}(d)^n = \{z = (z_1, \ldots, z_n) \in \mathbb{R}^n \mid z_i \in \mathbb{Z}(d)\}.$$

By selecting t points independently and uniformly at random from $\mathbb{Z}(d)^n$ and taking the convex hull we obtain a *random polytope of type* (n, d, t).

To get a random polynomial, we first pick t random points from $\mathbb{Z}(d)^n$ as above. This set S of points will be the support set of f. The polynomial f is then formed as

$$f = \mathrm{Poly}(S) := \sum_{z\in S} a_z X^z,$$

where $a_z \in K$ is arbitrarily nonzero and $X^z = X_1^{z_1} \ldots X_n^{z_n}$ for $z = (z_1, \ldots, z_n)$. So f has at most t nonzero terms and has degree in each variable at most d. Any nonconstant monomial factor of f is then removed. We call a polynomial chosen in such a fashion a *random polynomial* f *of type* (n, d, t) over K.

For different values of n, d and t, we shall measure experimentally the effectiveness of our algorithm for random polynomials of type (n, d, t). We wish to demonstrate that our algorithm is useful over a very wide range. Note that one could use an alternative notion of a random polynomial based upon restricting the total degree rather than the degree in each term, and also by choosing distinct random monomials rather than just random monomials. Implementations reveal, see Table 4, that our algorithm is also effective under this distribution.

5 Implementation details

The algorithm was implemented in C by the second author and programs run on a 550 MHz PC with 128 MB of RAM. Tables 1 - 7 in Appendix A give details of the results obtained; the glossary below explains the terms used in these tables.

Glossary of terms used:

Variables - the number of variables.
Degree - a bound on the degree in each variable.
Total Degree - a bound on the total degree.
Terms - the number of randomly selected monomials.
D.Terms - the number of randomly selected distinct monomials.
Success - the number of cases in which the Newton polytope of the randomly selected polynomial is integrally indecomposable (and thus the polynomial has at worst trivial factors which we assume have been removed).
Failure - the number of cases in which the algorithm gives no output (and thus the absolute irreducibility of the polynomial is left undetermined).
Time - the average time per randomly selected polynomial over all choices, given in 10^{-3} of a second (ms) or seconds (s), depending upon which is more appropriate.
Project. - the average number of projections required when the algorithm is successful.
Matrix Bound - the value of "b" in the algorithm, which bounds the absolute values of the random matrix entries.
Projection Bound - the value of "e" in the algorithm, which gives the maximum number of projections per polytope.

Thus a random polynomial of type (n, d, t) has "Variables = n", "Degree = d", and "Terms = t". The parameters "Total Degree" and "D. Terms" are used in Table 4 where we consider the slightly different notion of a random polynomial mentioned at the end of Section 4.

In the following sections we make brief comments on the effectiveness of our algorithm referring to the tables in Appendix A.

5.1 Bivariate polynomials

In general the algorithm becomes more effective the more variables there are in the polynomial under consideration. When looking at bivariate polynomials one can of course omit the random projection stage as the Newton polytope of the polynomial already lies in two dimensions. Randomly chosen polygons themselves are frequently decomposable, and this makes the algorithm far less useful for bivariate polynomials. However, the algorithm will always decide decomposability of polygons, whereas in higher dimensions it is possible that the decomposability will be left undecided after many projections.

In Table 1, the row with degree 5 means that, among $10,000$ polygons each formed by the convex hull of 10 random integral points in the square $[0,5] \times [0,5]$, $2,301$ of them are indecomposable while the rest are decomposable. (Or equivalently, among $10,000$ random polynomials of type $(2,5,10)$, $2,301$ have been shown to be absolutely irreducible whereas the irreducibility of the remainder has been left undecided.) If one increases the degree to 100, so picking points from the square $[0,100] \times [0,100]$, then about 90% of the polygons are indecomposable.

5.2 Trivariate polynomials

Observe from Table 3 that the algorithm has a near 100% success rate for polynomials chosen at random with number of terms suitably bounded with respect to the degree. For example, for random polynomials with no more than 10 terms and degree in each variable not greater than 5 the algorithm will almost without exception show them to be absolutely irreducible within 1 ms. Also one may increase the chance of success in certain cases by loosening the bound on the absolute values of the randomly chosen matrices, although this comes at the expense of a longer running time (see the starred row).

Table 2 gives less detailed information on the performance of the algorithm for polynomials of high degree. One practical problem on working with polynomials of very high degree is the space requirement of the algorithm. We found that this gave a limitation of around 2000 on the degree in each term for trivariate polynomials on a machine with 128MB RAM. However, up to this limit, absolute irreducibility of polynomials with a few hundred terms could be shown within minutes.

Table 4 gives information on polynomials of *total* degree bounded by 3. Here the monomials are distinct random monomials and thus column 1 contains the exact number of monomials in each polynomial. We have chosen to present this detailed information in a slightly different format since polynomials of low total degree are perhaps of greater practical interest, and also, at very low degree the difference between choosing distinct random monomials and just random monomials becomes significant.

5.3 Polynomials with more variables

For polynomials with more variables, Algorithm 4 becomes more effective. This is supported by Tables 5–7. Table 5 shows results for randomly chosen polynomials with 500 terms in

$n = 10$ variables and with degree d ranging from 3 to 20. For $d > 13$, we see that all of the $1,000$ random polynomials are found to be absolutely irreducible. Also the column "Project." indicates that the number of projections decreases as the degree increases.

In Table 6, we fix the number of variables to be $n = 10$ and the degree in each variable to be at most $d = 10$, and vary the number of terms. It is natural that the more terms the less effective our algorithm becomes. For a reasonable range of terms, our algorithm is very efficient. For example, for polynomials with 1000 random terms, more than 85% of them can be recoginzed to be irreducible in less than a second.

In Table 7, we fix the degree bound on each variable and the number of terms, and vary the number n of variables. We see that our algorithm becomes more effective as n increases. Also, the column "Project." indicates that indecomposable polytopes in higher dimensions need fewer projections to be proven so.

5.4 The effective range of the algorithm

A careful examination of the data indicates that our method is extremely effective if the number t of terms is about $\mathcal{O}(nd)$. More explicitly for $n > 3$, the effective range seems to be $t \leq 3nd$. This means that our method is indeed effective for sparse polynomials which have $\mathcal{O}(nd)$ terms, and as observed at the end of Section 3, the running time in this case is cubic in the input size. One may give a heuristic argument which to some extent explains this observation: For polynomials with $\mathcal{O}(nd)$ terms, in Step 3 of the Algorithm 4 one considers a polygon which is the convex hull of $\mathcal{O}(nd)$ points all lying in a square of side $\mathcal{O}(nd)$. It seems such polygons are frequently indecomposable in practice, and thus our algorithm is often successful for sparse polynomials. Also, note that if one increases the matrix bound with the other parameters fixed, the algorithm becomes more effective at the expense of a longer running time (see the starred row in Table 3). This corresponds to increasing the size of the square which the polygon in Step 3 lies within, which perhaps explains the improvement in effectiveness.

6 Conclusion

We presented a heuristic method for testing absolute irreducibility of multivariate polynomials over an arbitrary field. Our computer experiments indicate that it is most effective for sparse polynomials, that is to say for polynomials with $\mathcal{O}(nd)$ non-zero terms, where n is the number of variables and d is the degree in each variable. Since our algorithm is extremely fast for most polynomials, but not infallible, it can be used as a pretest before a slower general method is applied.

Our method is essentially one for testing indecomposability of integral polytopes. The data we obtained shows that almost all polytopes in high dimensions are indecomposable. It would be interest to verify this phenomenon in theory. Also, once one has established the indecomposability of a particular polytope, it follows that all polynomials whose Newton polytope is of that form are absolutely irreducible, regardless of the field over which they are defined. Thus our method allows one to show that families of polynomials are absolutely irreducible, and to find polynomials which are provably absolutely irreducible over "any field", a somewhat curious property.

Acknowledgements: The second author thanks Dr Gavin Brown of the Mathematical Institute, Oxford, for his illuminating observations on the applications of the algorithm, and the computer support staff at the Mathematical Institute for answering his programming questions.

References

[1] D. Duval, Absolute factorization of polynomials: a geometric approach, *SIAM J. Comput.* **20** (1991), 1–21.

[2] S. Gao, Absolute irreducibility of polynomials via Newton polytopes, *Journal of Algebra* **237** (2001), 501–520.

[3] S. Gao and A.G.B. Lauder, Decomposition of polytopes and polynomials, *Discrete and Computational Geometry* **26** (2001), 89–104.

[4] S. Gao, Factoring multivariate polynomials via partial differential equations, *Mathematics of Computation* **72** (2003), 801–822.

[5] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.

[6] J. von zur Gathen and E. Kaltofen, Factorization of multivariate polynomials over finite fields, *Math. Comp.* **45** (1985), no. 171, 251–261.

[7] J. von zur Gathen and I. Shparlinski, Computing components and projections of curves over finite fields, *SIAM. J. Comput.* **28** (1998), no. 3, 822–840.

[8] J.E. Goodman and J. O'Rourke (Eds), Handbook of Discrete and Computational Geometry, Elsevier Science, Amsterdam, 1997.

[9] D. Yu Grigoryev and A. L. Chistov, Fast factorization of polynomials into irreducible ones and the solution of systems of algebraic equations, *Dokl. Akad. Nauk SSSR* **275** (1984), no. 6, 1302–1306. English translation: *Soviet Math. Dokl.* **29** (1984), no. 2, 380–383.

[10] E. Kaltofen, Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization, *SIAM J. Comput.* **14** (1985), no. 2, 469–489.

[11] A. K. Lenstra, Factoring multivariate polynomials over finite fields, *J. Comput. System Sci.* **30** (1985), no. 2, 235–248.

[12] A. K. Lenstra, Factoring multivariate polynomials over algebraic number fields, *SIAM J. Comput.* **16** (1987), no. 3, 591–598.

[13] A. M. Ostrowski, On multiplication and factorization of polynomials, II. Irreducibility discussion, *Aequationes Math.* **14** (1976), 1–32.

A Appendix: Experimental data

Degree	Success	Failure	Time(ms)
5	2301	7699	0.1
10	4027	5973	0.2
15	5191	4809	0.3
20	6067	3933	0.4
30	7005	2995	0.8
40	7613	2369	1.4
50	8020	1980	2.1
100	8966	1034	8.2

Table 1: Variables 2, Terms 10.

Degree	Success	Failure	Project.	Time(s)
200	98	2	10	8.7
400	100	0	5	15.8
600	100	0	3	22.7
800	100	0	3	39.8
1000	100	0	3	54.2

Table 2: Variables 3, Terms 200, Matrix Bound 1, Projection Bound 100.

Degree	Terms	Success	Failure	Project.	Time
3	5	9876	124	2	0.2ms
3	10	9836	164	6	0.8ms
3	15	9276	724	13	2.7ms
3	20	7562	2438	18	6.9ms
3	25	5545	4455	23	12ms
3*	25	6282	3728	18	29ms
5	10	9991	9	4	0.8ms
5	20	9711	289	12	5ms
5	30	8222	1778	21	16ms
5	40	5766	4234	28	31ms
5	50	3688	6312	32	45ms
10	20	9991	9	5	6ms
10	40	9552	448	16	33ms
10	60	7907	2093	26	84ms
10	80	5821	4179	32	0.14s
10	100	3979	6021	36	0.19s
20	50	9938	62	10	82ms
20	100	8676	1324	24	0.36s
20	150	633	367	31	0.68s
20	200	434	566	35	0.94s
40	100	987	13	12	0.62s
40	200	848	152	26	2.5s
40	300	629	371	30	4.3s
40	400	451	549	37	5.8s
80	200	97	3	13	5.5s
80	400	78	22	28	19.9s
80	600	65	35	30	28.5s
80	800	54	46	30	35.9s

Table 3: Variables 3, Matrix Bound 2 (*Matrix Bound 4), Projection Bound 100.

D.Terms	Success	Failure	Project.	Time(ms)
3	9830	170	2	0.1
4	9071	929	3	0.4
5	9554	446	3	0.3
6	9429	571	4	0.5
8	8872	1128	7	1.2
10	7285	2715	9	2.8
12	4694	5306	12	5.3
14	2156	7844	13	7.9
16	460	9540	15	9.8
18	58	9942	49	10.4

Table 4: Variables 3, Total Degree 3, Matrix Bound 2, Projection Bound 100.

Degree	Success	Failure	Project.	Time(ms)
3	232	768	49	269
4	483	517	42	245
5	699	301	41	227
6	823	177	36	212
7	886	114	32	206
8	965	35	26	172
9	979	21	24	174
10	992	8	21	171
11	993	7	20	182
12	998	2	16	160
13	995	5	16	181
14	1000	0	13	165
15	1000	0	13	182
16	1000	0	11	174
17	1000	0	10	182
18	1000	0	10	181
19	1000	0	10	204
20	1000	0	9	204

Table 5: Variables 10, Terms 500 , Matrix Bound 1, Projection Bound 100

Terms	Success	Failure	Project.	Time
100	1000	0	5	16ms
200	1000	0	9	42ms
300	1000	0	13	75ms
400	998	2	17	115ms
500	986	14	21	168ms
600	979	21	24	220ms
700	959	41	28	294ms
800	927	73	32	380ms
900	904	96	33	433ms
1000	878	122	33	0.50s
2000	598	402	43	1.28s
3000	424	576	47	2.10s
4000	312	688	49	2.92s
5000	233	767	51	3.75s
7000	132	868	49	5.39s
10000	71	929	55	7.86s

Table 6: Variables 10, Degree 10, Matrix Bound 1, Projection Bound 100

Variables	Success	Failure	Project.	Time(ms)
3	140	860	52	106
4	717	283	47	134
5	706	294	39	127
6	953	47	27	85
7	997	3	18	63
8	1000	0	14	52
9	1000	0	10	44
10	1000	0	9	42
15	1000	0	5	37
20	1000	0	4	42
30	1000	0	3	48
40	1000	0	3	60
50	1000	0	3	66
75	1000	0	2	79
100	1000	0	2	110

Table 7: Degree 10, Terms 200, Matrix Bound 1, Projection Bound 100.