Comparison of Several Decision Algorithms for the Existential Theory of the Reals

Hoon Hong

September 11, 1991

Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria
e-mail: hhong@risc.uni-linz.ac.at

Abstract

In this paper we compare the complexities of the following three decision algorithms on existential sentences over the reals: Collins (1975), Grigor'ev and Vorobjov (1988), and Renegar (1989). Let n be the number of variables, m the number of polynomials, d the total degree, and L the coefficient bit length. The table below shows their(already known) theoretical complexities, along with their estimated running time for small inputs ($n = m = d = L = 2$) on currently available machines:

Algorithm	Theoretical	$n = m = d = L = 2$
Collins	$L^3(md)^{2^{O(n)}}$	$\ll$ 1 second
Grigor′ev/Vorobjov	$L(md)^{n^2}$	$\gg$ 1 million years
Renegar	$L(\log L)(\log\log L)(md)^{O(n)}$	$\gg$ 1 million years

Thus it suggests that Collins' algorithm is the fastest among them for inputs which can be decided in a reasonable amount of time.

1 Introduction

Since Tarski [33] gave the first decision algorithm for the first order theory of the reals, many other algorithms with better theoretical complexities have been proposed [32, 9, 17, 5, 10, 4, 13, 15, 14, 7, 16, 29, 30, 31]. In this paper we compare the complexities of the following three algorithms on existential sentences: Collins' [10], Grigor'ev and Vorobjov's [15, 14], and Renegar's [29, 30, 31].

Let n be the number of variables in the input sentence, m the number of polynomials, d the degree[1], and L the bit length bound for the coefficients. The table below shows their (already known) theoretical complexities:

Algorithm	Theoretical
Collins	$L^3(md)^{2^{O(n)}}$
Grigor′ev/Vorobjov	$L(md)^{n^2}$
Renegar	$L(\log L)(\log\log L)(md)^{O(n)}$

From this table one might conclude that Renegar's is the fastest, Grigor'ev the next, Collins the slowest. However one should be careful when comparing algorithms based on their theoretical complexities, since theoretical complexity results are usually obtained by ignoring various constant factors appearing during analyses. Therefore the theoretical complexities usually tell us how the algorithms behave for *sufficiently large* inputs. This naturally leads us to the following questions:

(Q1) Which is the fastest for small inputs? Let X be the fastest one.

(Q2) If X is not Renegar's, then at what input size does Renegar's algorithm become faster than X?

(Q3) At that "trade–off" point, how much computing time is required by either of the algorithms?

In order to answer these questions we estimated their running times[2] for input sentences characterized by $n = m = d = L = 2$. Here is a summary.

Algorithm	$n = m = d = L = 2$
Collins	$\ll 1$ second
Grigor′ev/Vorobjov	$\gg 1$ million years
Renegar	$\gg 1$ million years

In the following sections we will present detailed analyses on which these estimates are based, but here we first give a quick overview on why the algorithms of Grigor'ev/Vorobjov and Renegar take such huge computing time. The algorithm of Grigor'ev and Vorobjov requires solving at least 10^{18} systems of polynomial equations of degree about 30 over a certain algebraically closed field obtained by adjoining two infinitesimals to $\mathbb{R}$. Renegar's algorithm requires, beside several other expensive computations, at least 10^{11} applications of the algorithm of Ben-Or, Kozen, and Reif [4] on univariate polynomials of degree at least $20,000$. From these estimates one can infer the following (partial) answers to the questions posed above:

[1] Throughout this paper, unless stated otherwise, by the degree of a multivariate polynomial, we mean its total degree.

[2] From now, all the timings are for a DEC–station 5000, running UNIX operating system, with 32 Mega byte main memory.

(A1) Collins' algorithm is the fastest for small inputs.

(A2) We do not know the trade–off point.

(A3) The computing time at the trade–off point is expected to be far greater than a million years on currently available machines, suggesting that Collins' algorithm is the fastest on inputs which can be decided in a reasonable amount of time.

Collins' algorithm, without any modification, can decide arbitrary sentences (possibly with quantifier alternation). In [14] Grigor'ev, by generalizing the algorithm of Grigor'ev and Vorobjov, gave an algorithm for deciding arbitrary sentences. In [30] Renegar did the same by generalizing the algorithm of [29]. Inspection of these algorithms show that the dependence of the complexity of Collins' algorithm on the quantifier used in the input is negligible, while Grigor'ev's and Renegar's algorithms slow down when the number of quantifier alternations increases. Thus it is expected that Collins' algorithm is faster than either of the algorithms of Grigor'ev [14] and Renegar [30] for any inputs (possibly with quantifier alternation) which can be decided in a reasonable amount of time.

The structure of this paper is as follows: In Sections 2, 3, and 4 we give brief overviews of the algorithms of Collins, Grigor'ev/Vorobjov, and Renegar. In Section 5 we discuss the test sentences used in comparing the algorithms. In Sections 6, 7, and 8 we give detailed analyses of the complexities of the three algorithms on the test inputs. In Section 9 we make certain conclusions based on the results obtained from the previous three sections. Further we briefly discuss their implication on the complexities of the algorithms for arbitrary inputs (possibly with quantifier alternation). We also give a brief comparison of Collins' algorithm and Tarski's algorithm.

2 Collins' Algorithm

We give only a brief overview of Collins' algorithm [10], since it is quite well known and understood. Those who are not familiar with this algorithm are referred to [10, 2].

The algorithm actually solves a more general class of problems, namely quantifier elimination in the first order theory of the reals. However here we will discuss the algorithm only as a decision procedure for the existential theory.

Collins' algorithm begins by constructing a *Cylindrical Algebraic Decomposition* (CAD) of the polynomials occurring in the input sentence, where a CAD of a set of polynomials in n variables is a certain finite partition of the n-dimensional real space such that in each element (cell) of the partition the polynomials have constant signs. Thus the signs of the polynomials in a cell can be determined by evaluating the polynomials on a "sample" point belonging to the cell.

The algorithm constructs a CAD of the input polynomials in three stages:

Projection Stage: The input n–variate polynomials are "projected" into a set of $(n-1)$–variate polynomials in a way that a CAD of n–space can be easily built on top of a CAD of $(n-1)$–space. This process is successively carried out until univariate projection polynomials are obtained.

Base Stage: A CAD of 1–space is constructed through isolating the real roots of the univariate projection polynomials.

Extension Stage: A CAD of 2–space is constructed by building "stacks" of cells over the cells of the CAD of 1–space. This process is successively carried out until a CAD of n–space is constructed.

Once a CAD of the polynomials in the input sentence is constructed, the truth of the sentence can be easily decided as follows: first determine the truth of the input matrix on each sample point. If there is at least one sample point on which the matrix is true, then the input sentence is true. Otherwise it is false. We will call this process **Decision Stage**.

In [10] it is shown that the worst time complexity of this algorithm is dominated by $L^3(md)^{2^{O(n)}}$.

Collins' algorithm has gone through various improvements [1, 28, 18, 19, 27, 20, 11, 26, 6, 25]. All these improvements, though do not change the theoretical complexity, gave significant speed-ups in practice, enabling mechanical solution of various nontrivial problems [3, 20, 21].

3 Algorithm of Grigor'ev and Vorobjov

Now we give an overview of the algorithm of Grigor'ev and Vorobjov [15]. Their algorithm, given a system of polynomial inequalities, decides whether there exists a real solution. In the case of positive answer, the algorithm also constructs a representative set for the family of components of connectivity of the set of all real solutions of the system, which is $\mathcal{T}$ in Step (8) of the algorithm described below. But in this paper we will investigate the algorithm only with respect to decision problem.

We begin by defining several notations which will be used in the algorithm description:

- Let K be an arbitrary ordered field. Then $\bar{K}$ denotes the algebraic closure of K, and $\tilde{K}$ the real algebraic closure of K.
- Let ϵ_1 be a positive infinitesimal over $\tilde{\mathbb{Q}}$. Then F_1 denotes the real algebraic closure of $\tilde{\mathbb{Q}}(\epsilon_1)$.

- Let ϵ be a positive infinitesimal over F_1. Then F denotes the real algebraic closure of $F_1(\epsilon)$.
- Let $f_1, \ldots, f_m$ be elements of $K[x_1, \ldots, x_n]$. Then $\{f_1 = 0, \ldots, f_m = 0\}$ denotes the set of all solutions of the system $f_1 = \cdots = f_m = 0$ over the algebraic closure of K.

We did our best to preserve the notations of [15] in order to help the reader who might want to refer to their paper. In a few places, however, we re-indexed several symbols in order to provide a uniform indexing among the three algorithms investigated in this paper.

We also omitted various details of several steps when they are not essential in comparing the complexity of this algorithm to those of Collins' or Renegar's. Those who are interested in the details are referred to the original paper [15].

$t \longleftarrow$ **Decide**(P)

Input: P is a quantifier–free formula of the following kind:

$$f_1 > 0 \ \wedge \cdots \wedge \ f_k > 0 \ \wedge \ f_{k+1} \geq 0 \ \wedge \cdots \wedge \ f_m \geq 0$$

where $f_i \in \mathbb{Z}[x_1, \ldots, x_n]$.

Output: t is the truth of the sentence $(\exists x_1 \in \mathbb{R}) \cdots (\exists x_n \in \mathbb{R}) P(x_1, \ldots, x_n)$.

(1) Let $f_{m+1} = x_0 f_1 \cdots f_k - 1$.
Let $g_1 = (f_1 + \epsilon_1) \cdots (f_{m+1} + \epsilon_1) - \epsilon_1^{m+1} \in \mathbb{Z}[\epsilon_1][x_0, \ldots, x_n]$.

(2) Let $\bar{L}$ be a bit length bound for the coefficients of f_i, and $\bar{d}$ a degree bound of f_i for every $1 \leq i \leq m+1$.
Let $R = 3^{(\bar{L} + \log(m+1))p(\bar{d}^{n+1})}$, where $p \in \mathbb{Z}[x]$ is a certain polynomial defined in Page 62 of [15].
Let $g = g_1^2 + (x_0^2 + \cdots + x_{n+1}^2 - (R+1))^2 \in \mathbb{Z}[\epsilon_1][x_0, \ldots, x_{n+1}]$.

(3) Let $N = (8md)^{n+2}$.
Let $\Gamma = \{1, \ldots, N\}^{n+1}$.
Let $\mathcal{I}' = \{\}$.

(4) For each $\gamma = (\gamma_1, \ldots, \gamma_{n+1}) \in \Gamma$ do

(4.1) Let $\bar{V}^{(\epsilon)} \subset \bar{F}^{n+2}$ be the variety of the system

$$g - \epsilon = \left(\frac{\partial g}{\partial x_1}\right)^2 - \frac{\gamma_1}{N(n+2)}\Delta = \cdots = \left(\frac{\partial g}{\partial x_{n+1}}\right)^2 - \frac{\gamma_{n+1}}{N(n+2)}\Delta = 0$$

where $\Delta = \sum_{j=0}^{n+1} \left(\frac{\partial g}{\partial x_j}\right)^2$.
Let $\bar{V}^{(\epsilon)} = \bigcup_j \bar{V}_j^{(\epsilon)}$, where each $\bar{V}_j^{(\epsilon)}$ is a component irreducible over the field $\mathbb{Q}(\epsilon_1, \epsilon)$.

(4.2) For each null dimensional $\bar{V}_j^{(\epsilon)}$ do

(4.2.1) Let $\mathcal{R}_1 \subset \bar{F}_1^n$ be a set containing the standard part (relative to ϵ) of every point (for which the standard part is definable) from the component $\bar{V}_j^{(\epsilon)}$.

(4.2.2) Let $\mathcal{R}'_1 = \mathcal{R}_1 \cap \{g = 0\} \cap F_1^{n+2}$.

(4.2.3) Set $\mathcal{I}'$ to $\mathcal{I}' \cup \mathcal{R}'_1$.

(5) Let $\mathcal{I} = \pi(\mathcal{I}') \subset F_1^{n+1}$ where $\pi(x_0, \ldots, x_{n+1}) = (x_0, \ldots, x_n)$.

(6) Let $\mathcal{R} \subset \bar{\mathbb{Q}}^{n+1}$ be a set containing the standard part (relative to ϵ_1) of every point (for which the standard part is definable) from the set $\mathcal{I}$.

(7) Let $\mathcal{T}' = \mathcal{R} \cap \{f_1 \geq 0, \ldots, f_{m+1} \geq 0\} \cap \tilde{\mathbb{Q}}^{n+1}$.

(8) Let $\mathcal{T} = \pi_1(\mathcal{T}')$ where $\pi_1(x_0, \ldots, x_n) = (x_1, \ldots, x_n)$.

(9) Finally let t be true if $\mathcal{T}$ is non-empty, false otherwise. ■

In [15] it shown that the worst case time complexity of this algorithm is dominated by $L(md)^{n^2}$.

In [14] Grigor'ev, by generalizing this algorithm, gave a decision algorithm for the first order theory of real closed fields.

4 Renegar's Algorithm

Now we give an overview of Renegar's algorithm [29]. We did our best to keep Renegar's original notations in order to help the reader who might want to refer to his paper. In a few places, however, we corrected simple typographical errors in his paper.

$t \longleftarrow$ **MainAlgorithm** (P)

Input: P is a quantifier–free formula with variables $x_1, \ldots, x_n$.
Output: t is the truth of the sentence $(\exists x_1 \in \mathbb{R}) \cdots (\exists x_n \in \mathbb{R}) P(x_1, \ldots, x_n)$.

(1) Let $g = \{g_1, \ldots, g_m\}$, $g_i \in \mathbb{Z}[x_1, \ldots, x_n]$, be the polynomials occurring in the formula P.
Let $h = \{h_1, \ldots, h_{6m+2}\}$, where

$$
\begin{array}{lcll}
h_i & = & g_i & i=1,\ldots,m \\
h_{m+i} & = & x_0 g_i - 1 & i=1,\ldots,m \\
h_{2m+i} & = & x_0 g_i + 1 & i=1,\ldots,m \\
h_{3m+1} & = & x_0 - 1 & \\
h_{3m+1+i} & = & -g_i & i=1,\ldots,m \\
h_{4m+1+i} & = & -x_0 g_i + 1 & i=1,\ldots,m \\
h_{5m+1+i} & = & -x_0 g_i - 1 & i=1,\ldots,m \\
h_{6m+2} & = & -x_0 + 1 &
\end{array}
$$

(2) Let d be the maximum of the degrees of the polynomials g_i.
Let d' be the least even integer which is greater than or equal to $d+1$.
Let $\bar{h} = \{\bar{h}_1, \ldots, \bar{h}_{6m+2}\}$, where

$$\bar{h}_i(\delta; x_0, \ldots, x_n) = (1-\delta)h_i + \delta(1 + \sum_{j=0}^{n} i^j x_j^{d'}).$$

(3) For each $A \subseteq \{1, \ldots, 6m+2\}$ such that $|A| \leq n+1$ do:
(3.1) Let $\hat{h}(x_0, \ldots, x_n) = \sum_{j=0}^{n} (6m+3)^j x_j^{d'}$.
Let M_A be the matrix with the last row $\nabla_{x_0,\ldots,x_n} \hat{h}$ and with earlier rows $\nabla_{x_0,\ldots,x_n} \bar{h}_i$, $i \in A$, ordered by increasing indices i.
(3.2) Let $h_A(\delta; x_0, \ldots, x_n) = \det(M_A M_A^T) + \sum_{i \in A} \bar{h}_i^2$.
Let d_A be the degree of h_A with respect to $x_0, \ldots, x_n$.
(3.3) Let $\tilde{h}_A(\epsilon, \delta; x_0, \ldots, x_n) = (1-\epsilon)h_A - \epsilon \sum_{j=0}^{n} x_j^{d_A}$.
Let $\tilde{h}_A^{(i)} = \frac{\partial \tilde{h}_A}{\partial x_i}$ for each i.
(3.4) Let $R_A(\epsilon, \delta; u_0, \ldots, u_{n+1})$ be the modified u–resultant of the polynomials $\tilde{h}_A^{(0)}, \ldots, \tilde{h}_A^{(n)}$. (Use the sub-algorithm shown below.)
(3.5) Let R_A be expanded as $\sum_{i,j,k} \epsilon^i \delta^j u_0^k R_A^{<i,j,k>}$.
(4) Let $\mathcal{R}$ be the set of all $R_A^{<i,j,k>}$ computed in Step (3).
(5) Let $\mathcal{B}_{n+1,D} = \{(i^{n-1}, i^{n-2}, \ldots, 1, 0) | 0 \leq i \leq nD^2\}$.
For $R \in \mathcal{R}$, let D_R be the degree of R.
Let $\mathcal{Q} = \{\frac{d^j}{dt^j} \nabla_{u_1,\ldots,u_{n+1}} R(\beta + te_{n+1}) | R \in \mathcal{R}, \beta \in \mathcal{B}_{n+1,D_R}, j \in [0, D_R]\}$.
(6) Let $G = \{G_1, \ldots, G_m\}$, $G_i \in \mathbb{Z}[x_1, \ldots, x_{n+1}]$, be such that each G_i is the degree d–homogenization of g_i; *i.e.* the monomials of G_i are obtained from those of g_i by multiplying by the appropriate powers of x_{n+1} so as to become of degree d.
Let $\mathcal{F}^+ = \{(G_1(q), \ldots, G_m(q), q_{n+1}) | q \in \mathcal{Q}\}$.
Let $\mathcal{F}^- = \{(G_1(-q), \ldots, G_m(-q), -q_{n+1}) | q \in \mathcal{Q}\}$.
Let $\mathcal{F} = F^+ \cup F^-$.
(7) For each $f \in \mathcal{F}$ let $\bar{S}_f$ be the set of all consistent sign vectors for the polynomials in f. (Use the algorithm of Ben-Or, Kozen, and Reif [4].)
(8) Let $\bar{S} = \bigcup_{f \in \mathcal{F}} \bar{S}_f$.
Let $S = \{(\sigma_1, \ldots, \sigma_m) | (\sigma_1, \ldots, \sigma_m, 1) \in \bar{S}\}$.

Finally let $t = \bigvee_{\sigma \in S} P_\sigma$, where the formula P_σ is obtained from the formula P by replacing g_i with σ_i. ■

$R \longleftarrow$ **SubAlgorithm** $(f_0, \ldots, f_n)$[3]

Input: $f_0, \ldots, f_n \in W[x_0, \ldots, x_n]$, where W is a commutative ring with unity.

Output: $R \in W[u_0, \ldots, u_{n+1}]$ is the modified u-resultant of $f_0, \ldots, f_n$. (See Renegar's paper [29] for the definition.)

(1) Let $\tilde{d}$ be the maximum of the degrees of f_i.
Let $\mathbb{B} = \{x_0^{d_0} \cdots x_{n+1}^{d_{n+1}} | d_0 + \cdots + d_{n+1} = \hat{d}\}$ where $\hat{d} = (n+1)(\tilde{d} - 1) + 1$.
Let $\mathbb{H}$ be the vector space generated by the basis $\mathbb{B}$ over the ring W.

(2) For each $x_0^{d_0} \cdots x_{n+1}^{d_{n+1}} \in \mathbb{B}$, let i denote the least index $i \leq n$ and $\tilde{d} \leq d_i$ if such an i exists, otherwise let $i = n + 1$.
Let $F_i \in W[x_0, \ldots, x_{n+1}]$ be the degree $\tilde{d}$–homogenization of f_i for each i.
Let $T : \mathbb{H} \longrightarrow \mathbb{H}$ be the linear transformation defined by the following mapping on the basis $\mathbb{B}$:

$$T(x_0^{d_0} \cdots x_{n+1}^{d_{n+1}}) = \begin{cases} x_0^{d_0} \cdots x_i^{d_i - \tilde{d}} \cdots x_{n+1}^{d_{n+1}} F_i & \text{if } i \leq n \\ x_0^{d_0} \cdots x_n^{d_n} x_{n+1}^{d_{n+1}-1} u \cdot x & \text{if } i = n+1 \end{cases}$$

where $u = (u_0, \ldots, u_{n+1})$ and $x = (x_0, \ldots, x_{n+1})$.
Let M be the matrix representing T with respect to the basis $\mathbb{B}$.

(3) Finally set $R = D! \det(M)$ where M is $D \times D$ matrix. ■

In [29] it is shown that the worst case time complexity of this algorithm is dominated by $L(\log L)(\log \log L)(md)^{O(n)}$, which is the best compared to the theoretical complexities of all other algorithms proposed in the literature so far.

In [30, 31] Renegar, by generalizing the above algorithm, gave a quantifier elimination algorithm for the first order theory of the reals.

5 Input Sentences Used for Comparison

In an attempt to answer the questions (A1)—(A3) posed in the introduction, we will compare the complexities of the three algorithms on small inputs. In particular we will use the sentences with the following characteristics:

$$n = m = d = L = 2$$

[3] Our indexing starts from 0 while Renegar's starts from 1. We made this change in order to enhance the cross–readability between the main algorithm and this sub-algorithm.

where n is the number of variables, m the number of polynomials, d their (total) degree bound, and L the bit length bound on their coefficients.[4] We will make sure that our analysis does not depend on any other characteristics of the input sentences other than the ones given just above.

However we will also trace the three algorithms on the following sentence in order to obtain a concrete impression of the various estimates appearing during the analysis:

$$(\exists x_1)(\exists x_2)\ \ [\, x_1^2 + x_2^2 < 1 \quad \wedge \quad x_1 x_2 > 1 \,].$$

Simple drawings of the two curves $x_1^2 + x_2^2 = 1$ and $x_1 x_2 = 1$ show immediately that this sentence is false.

The traces shown in the following three sections have been obtained from interactive sessions on computer algebra systems MAPLE [8] and SAC2 [12], running on a DEC–Station 5000 with 32 Mega bytes of main memory. In order to enhance readability of the trace, certain minor editing has been made on the original trace. In the traces we also used the following transcription for various mathematical symbol modifiers:

h_i	`hi`
h^i	`hi`
$\bar{h}$	`hb`
$\tilde{h}$	`ht`
$\hat{h}$	`hh`
h'	`hp`
h^*	`hs`

6 Analysis of Collins' Algorithm

In this section we analyze the complexity of Collins' algorithm following its stages one after the other. In doing so we use the original algorithm of Collins without taking any advantage of subsequent improvements. As mentioned in the previous section, we will also consider only the case $n = m = d = L = 2$.

Projection Stage

We are given two input polynomials g_1 and g_2 of degree two in two variables. For the example input, they are

$$\begin{aligned} g_1 &= x_2^2 + x_1^2 - 1 \\ g_2 &= x_1 x_2 - 1 \end{aligned}$$

[4] More precisely, the absolute value of any coefficient is less than 2^{L-1}. Here the exponent is $L-1$ because one bit is used for indicating the sign.

Since there are only two variables, we need to project the input polynomials only once, obtaining in general the following univariate polynomials:

$$\begin{gathered} ldcf(g_1) \\ ldcf(g_2) \\ discr(g_1) \\ discr(g_2) \\ res(g_1, g_2) \end{gathered}$$

where $ldcf$ stands for leading coefficient, $discr$ for discriminant, and res for resultant. For certain inputs, some of the polynomials above might not appear in the projection set.

The polynomials $ldcf(g_1)$ and $ldcf(g_2)$ can be trivially obtained, and their degrees are at most 1. In order to compute $discr(g_1)$ and $discr(g_2)$, one only needs to compute the determinants of at most 3×3 matrices, and their degrees are at most 2. In order to compute $res(g_1, g_2)$, one only needs to compute the determinant of at most 4×4 matrix, and its degree is at most 4.

For the example input, we have the following projection polynomials:

$$\begin{aligned} discr(g_1) &= -4x_1^2 + 4 \\ ldcf(g_2) &= x_1 \\ res(g_1, g_2) &= x_1^4 - x_1^2 + 1 \end{aligned}$$

Base Stage

Next we isolate the real roots of the univariate projection polynomials. Each polynomial $ldcf(g_i)$ has at most one rational root. Each polynomial $disc(g_i)$ has at most 2 real roots of algebraic degree[5] at most 2. The polynomial $res(g_1, g_2)$ has at most 4 real roots of algebraic degree at most 4.

Thus altogether there are at most 10 real roots, in turn at most 21 cells in the CAD of 1–space. Among them, about 13 cells have rational sample points, about 4 cells have sample points of algebraic degree 2, and about 4 cells have sample points of algebraic degree 4.

For the example input, we have three real roots: -1, 0, 1. Thus the CAD of 1–space has 7 cells with the following sample points:

Cell	Sample point
(1)	-2
(2)	-1
(3)	$-1/2$
(4)	0
(5)	$1/2$
(6)	1
(7)	2

[5] By the algebraic degree of a number, we mean the degree of an irreducible polynomial with rational coefficients which has the number as a root.

Extension Stage

Now on each cell of the CAD of 1-space, we build a stack of cells, by substituting its sample point into the polynomials g_i, and by isolating the real roots.

Since each polynomial g_i is of degree at most 2 after substitution, it has at most 2 real roots, and thus each stack has at most 9 cells. Therefore all together the CAD of 2–space has at most $21 \cdot 9 = 189$ cells. Among them, about $13 \cdot 5 = 65$ cells have rational sample points, about $13 \cdot 4 + 4 \cdot 5 = 72$ cells have sample points of algebraic degree 2, about $4 \cdot 4 + 4 \cdot 5 = 36$ cells have sample points of algebraic degree 4, and about $4 \cdot 4 = 16$ cells have sample points of algebraic degree 8.

In order to find out the actual number of the cells for the example input, let us first build a stack on the cell (1). After substituting $x_1 = -2$ into g_1 and g_2, we obtain the polynomials:

$$\begin{aligned} h_1 &= x_2^2 + 3 \\ h_2 &= -2x_2 - 1 \end{aligned}$$

The polynomial h_1 does not have any real root, and h_2 has only one real root $-1/2$. Thus the stack over the cell (1) has 3 cells with the following sample points:

$$\begin{array}{ll} (1,1) & (-2,-1) \\ (1,2) & (-2,-1/2) \\ (1,3) & (-2,0) \end{array}$$

Continuing in the same way with all other cells in the CAD of 1–space, we obtain a CAD of 2–space with 35 cells as shown in Table 1.

Decision Stage

Now we determine the signs of the input polynomials g_1 and g_2 on each sample point of the CAD of 2-space, and by using them determine also the truth of the input matrix[6] on each sample point. If there is at least one sample point on which the matrix is true, then the input sentence is true. Otherwise the input sentence is false.

Thus in order to decide the example input, we begin by determining the signs of the polynomials $g_1 = x_1^2 + x_2^2 - 1$ and $g_2 = x_1 x_2 - 1$ on the point $(-2,-1)$, which is the sample point of the cell (1,1). We find that $g_1 > 0$ and $g_2 > 0$ on it. Thus the matrix

$$g_1 < 0 \quad \wedge \quad g_2 > 0$$

is false on the cell (1,1). Continuing in the same way, we find that the matrix is false on all other cells. Therefore the input sentence is decided to be false.

[6] By the input matrix, we mean the quantifier–free part of the input sentence.

Cell	Sample point
$(1,1)$	$(-2,-1)$
$(1,2)$	$(-2,-1/2)$
$(1,3)$	$(-2,0)$
$(2,1)$	$(-1,-2)$
$(2,2)$	$(-1,-1)$
$(2,3)$	$(-1,-1/2)$
$(2,4)$	$(-1,0)$
$(2,5)$	$(-1,1)$
$(3,1)$	$(-1/2,-3)$
$(3,2)$	$(-1/2,-2)$
$(3,3)$	$(-1/2,-1)$
$(3,4)$	$(-1/2,-\sqrt{3/4})$
$(3,5)$	$(-1/2,0)$
$(3,6)$	$(-1/2,\sqrt{3/4}$
$(3,7)$	$(-1/2,1)$
$(4,1)$	$(0,-2)$
$(4,2)$	$(0,-1)$
$(4,3)$	$(0,0)$
$(4,4)$	$(0,1)$
$(4,5)$	$(0,2)$
$(5,1)$	$(1/2,-1)$
$(5,2)$	$(1/2,-\sqrt{3/4})$
$(5,3)$	$(1/2,0)$
$(5,4)$	$(1/2,\sqrt{3/4})$
$(5,5)$	$(1/2,1)$
$(5,6)$	$(1/2,2$
$(5,7)$	$(1/2,3)$
$(6,1)$	$(1,-1)$
$(6,2)$	$(1,0)$
$(6,3)$	$(1,1/2)$
$(6,4)$	$(1,1)$
$(6,5)$	$(1,2)$
$(7,1)$	$(2,0)$
$(7,2)$	$(2,1/2)$
$(7,3)$	$(2,1)$

Table 1: A CAD of 2-space for $x_1^2+x_2^2-1$ and x_1x_2-1

This whole computation can be easily carried out by hand. It was also carried out on a computer (DEC-Station 5000), taking about 0.1 second. We also carried out experiments on every input sentence of the following form:

$$(\exists x_1)(\exists x_2) \qquad \begin{array}{l}[a_1x_1^2 + a_2x_2^2 + a_3x_1x_2 + a_4x_1 + a_5x_2 + a_6 > 0 \wedge \\ a_7x_1^2 + a_8x_2^2 + a_9x_1x_2 + a_{10}x_1 + a_{11}x_2 + a_{12} > 0]\end{array}$$

where each a_i ranges on $\{-1, 0, 1\}$. Thus altogether we have carried out $3^{12} = 531441$ experiments. We did not test the inputs with the other relational operators such as $=$ or $<$, since the computing time of Collins' algorithm does not depend on the relational operators being used. The experiments show that every input sentence of the above form is decided within a second. (More precisely between 10 through 200 milli-seconds.) Thus we can conclude that Collins' algorithm can decide any sentence with $n = m = d = L = 2$ within a second (on the same computer).

7 Analysis of the algorithm of Grigor'ev and Vorobjov

Now we analyze the complexity of the algorithm of Grigor'ev and Vorobjov, mainly for the case $n = m = d = L = 2$.

Step (1)

In general we begin with the polynomials $f_1, \ldots, f_k$ and $f_{k+1}, \ldots, f_m$ in the variables $x_1, \ldots, x_n$ of degree d and of coefficient bit length bound L. For the example input, we have only strict inequalities, and thus we have $k = n = m = d = L = 2$. The input polynomials are:

```
f1 = x1^2 + x2^2 - 1
f2 = x1 x2 - 1.
```

From these we form the polynomial $f_{m+1} = x_0 f_1 \cdots f_k - 1$. Clearly the degree of g_1 is bounded by $kd + 1$. For the example input we have

```
f3 = x0 f1 f2 - 1
   = x0 x1^3 x2 - x0 x1^2 + x0 x2^3 x1 - x0 x2^2 - x0 x1 x2 + x0 - 1
```

Next we form the polynomial $g_1 = (f_1 + \epsilon_1) \cdots (f_{m+1} + \epsilon_1) - \epsilon_1^{m+1}$. Obviously the degree of g_1 is bounded by $md + kd + 1$. For the example input we have

```
g1 = (f1 + e1) (f2 + e1) (f3 + e1) - e1^3
   = - 1 + 4 x0 x2^3 x1 + x1 x2 - x1^3 x2 - x2^3 x1 - 2 x0 x1^2 - 2 x0 x2^2 +
     3 e1 - 2 x1^2 e1 - 2 x2^2 e1 - 3 e1^2 + x1^4 x0 + x1^2 e1^2 + x2^4 x0+ x2
     ^2 e1^2 - 2 e1 x0 + e1^2 x0 + x1^2 + x2^2 + x0 + 4 x0 x1^3 x2 - 2  x0 x1
     x2 - 2 e1 x1 x2 + x1^6 x2^2 x0 - 2 x1^5 x2 x0 + 2 x1^4 x2^4 x0  - 4 x1^3
     x2^3 x0 - 2 x1^4 x2^2 x0 + 3 x1^2 x0 x2^2 - x1^4 e1 x0 + x2^6 x1^2 x0 - 2
     x2^5 x1 x0 - 2 x2^4 x1^2 x0 - x2^4 e1 x0 + e1^2 x1 x2 - e1^2 x0 x1^2 - e1
     ^2 x0 x2^2 + x1^3 x2 e1 + 3 x1^2 e1 x0 + x2^3 x1 e1 + 3 x2^2 e1 x0 + x1^5
     e1 x0 x2 + 2 x1^3 e1 x0 x2^3 - 3 x1^2 e1 x0 x2^2 - 4 x1^3 e1 x0 x2 + x2^5
     e1 x0 x1 - 4 x2^3 e1 x0 x1 + e1 x1^4 x2^2 x0 + e1 x1^2 x2^4 x0 + e1^2 x0
     x1^3 x2 + e1^2 x0 x2^3 x1 - e1^2 x0 x1 x2 + 3 e1 x0 x1 x2
```

Step (2)

Now we need to compute the integer R, and for this we need to determine the polynomial p. The following lemma indirectly defines what the polynomial p is.

Lemma 1 (Lemma 10 of [15]) *There exists a polynomial $p \in \mathbb{Z}[x]$ with the following property:*

Consider any system $f_1 \geq 0, \ldots, f_m \geq 0$ where for every i, $f_i \in \mathbb{Z}[x_1, \ldots, x_n]$, $\deg(f_i) < d$, the bit length of the coefficients of f_i is bounded by L, and such that at least one real point does not satisfy the system. Let $W = \bigcup_j W_j$ be the semi-algebraic set consisting of all real solutions of the system, where each W_j is a connected component. Let $R = 3^{(L+\log m)p(d^n)}$. Let $\mathcal{D}(R)$ denote the closed ball centered at the origin with radius R. Then for each W_j we have $\mathcal{D}(R) \cap W_j \neq \emptyset$ and $\mathcal{D}(R) \setminus W_j \neq \emptyset$.

In [15] this lemma is proved, but without constructing one such polynomial p. Therefore in our analysis, we will be satisfied with the following extreme lower bound: $p(x) \geq 0$ for all positive x. Thus we have $R \geq 1$. In fact we will use $R = 1$, since we already "know" that the example input system does not have any solution and thus any positive integer would be acceptable to be used as R. But one should still remember that the Grigor'ev and Vorobjov algorithm would assign (much) bigger integer to R.

Next we form the polynomial $g = g_1^2 + (x_0^2 + \cdots + x_{n+1}^2 - (R+1))^2$. The degree of g is bounded by $2(md + kd + 1)$. For the example input, we have

```
g = g1^2 + (x0^2 + x1^2 + x2^2 + x3^2 - 2)^2
  = 5 - 18 x0 x2^3 x1 + 6 x2^6 x0 e1^2 - 2 x1 x2 + 4 x1^3 x2 + 4 x2^3 x1 + 6
    x0 x1^2 + 6 x0 x2^2 - 6 e1 + 10 x1^2 e1 + 10 x2^2 e1 + 15 e1^2 - 6 x1^4 x
    0 - 20 x1^2 e1^2 - 6 x2^4 x0 - 20 x2^2 e1^2 + 10 e1 x0 - 20 e1^2 x0 - 6 x
    1^2 - 6 x2^2 - 2 x0 - 18 e1^3 - 18 x0 x1^3 x2 + 6 x0 x1 x2 + 10 e1 x1 x2
    - 18 x1^6 x2^2 x0 + 18 x1^5 x2 x0 - 36 x1^4 x2^4 x0 + 38 x1^3 x2^3 x0 + 2
    4 x1^4 x2^2 x0 - 18 x1^2 x0 x2^2 + 22 x1^4 e1 x0 - 18 x2^6 x1^2 x0 + 18 x
    2^5 x1 x0 + 24 x2^4 x1^2 x0 + 22 x2^4 e1 x0 - 20 e1^2 x1 x2 + 44 e1^2 x0
    x1^2 + 44 e1^2 x0 x2^2 - 16 x1^3 x2 e1 - 26 x1^2 e1 x0 - 16 x2^3 x1 e1 -
    26 x2^2 e1 x0 - 54 x1^5 e1 x0 x2 - 114 x1^3 e1 x0 x2^3 + 66 x1^2 e1 x0 x2
    ^2 + 66 x1^3 e1 x0 x2 - 54 x2^5 e1 x0 x1 + 66 x2^3 e1 x0 x1 - 3 x0^2 - 4
    x3^2 - 72 e1 x1^4 x2^2 x0 - 72 e1 x1^2 x2^4 x0 - 92 e1^2 x0 x1^3 x2 - 92
    e1^2 x0 x2^3 x1 + 44 e1^2 x0 x1 x2 - 26 e1 x0 x1 x2 + 40 x0^2 x2^6 x1^2 -
    52 x0^2 x2^3 x1^3 - 24 x0^2 x2^5 x1 + 64 x0^2 x2^3 x1^5 + 24 x1^3 x2 e1^2
    + 6 x1^5 x2 e1 + 12 x1^3 x2^3 e1 - 6 x1^7 x2 x0 - 6 x1^5 x2 e1^2 + 6 x2^5
    x1 e1 + 24 x2^3 x1 e1^2 - 24 x2^3 x1^5 x0 - 12 x2^3 x1^3 e1^2 + 18 x0^2 x
    1^2 x2^2 - 30 x0 x1^4 e1^2 - 36 x0^2 x2^2 x1^4 + 18 x1^2 e1^2 x2^2 - 6 x1
    ^6 e1 x0 - 12 x2^2 e1^3 x1^2 + 5 x1^2 x2^2 - 2 x1^4 x2^2 - 2 x1^2 x2^4 +
    x3^4 + x0^4 + 9 e1^4 + 2 x1^4 + 2 x2^4 + 58 x0 x2^5 x1 e1^2 - 48 x0^2 x2^
    3 x1 e1 + 52 x0^2 x2^3 x1 e1^2 + 122 x0 x2^3 x1^3 e1^2 + x1^6 x2^2 + 2 x1
    ^4 x2^4 + x2^6 x1^2 + 6 x0^2 x1^4 - 4 x0^2 x1^6 + 6 x0^2 x2^4 + 18 x1^2 e
    1^3 + 6 x1^4 e1^2 - 4 x1^4 e1^3 + 6 x2^4 e1^2 + 18 x2^2 e1^3 - 6 e1^4 x1^
    2 + x1^8 x0^2 + x1^4 e1^4 - 2 x1^5 x2 - 4 x1^3 x2^3 - 2 x2^5 x1 - 2 x0^2
    x1^2 - 4 x0^2 x2^6 - 2 x0^2 x2^2 + 18 e1^3 x0 - 4 x1^4 e1 - 4 x2^4 e1^3 -
    4 x2^4 e1 - 6 e1^4 x2^2 - 6 e1^4 x0 + 2 x1^6 x0 + x2^8 x0^2 + 2 x2^6 x0 +
    x2^4 e1^4 + 6 e1^2 x0^2 - 4 e1^3 x0^2 - 4 e1 x0^2 + e1^4 x0^2 - 90 x0 x2^
```

```
2 x1^2 e1^2 + 6 x1^6 x0 e1^2 + 16 x0^2 x2^7 x1 + 16 x0^2 x2^3 x1 + 79 x0^
2 x2^4 x1^4 - 36 x0^2 x2^4 x1^2 + 48 x0^2 x2^5 x1^7 - 76 x0^2 x2^4 x1^6 +
48 x0^2 x2^7 x1^5 - 12 x1^2 x2^2 e1 + 6 x1^7 x2^3 x0 + 12 x1^5 x2^5 x0 -
24 x1^3 x2^5 x0 + 6 x1^4 x2^2 e1 - 2 x1^9 x2^3 x0 + 6 x1^8 x2^2 x0 - 6 x1
^7 x2^5 x0 - 6 x2^7 x1 x0 - 6 x2^5 x1 e1^2 + 6 x2^4 x1^2 e1 + 18 x2^4 x1^
6 x0 - 6 x2^7 x1^5 x0 + 14 x0^2 x1^2 e1 - 18 x0^2 x1^2 e1^2 - 24 x0^2 x1^
5 x2 + 16 x0^2 x1^3 x2 - 24 x0^2 x1^8 x2^2 + 16 x0^2 x1^7 x2 - 30 x0 x2^4
e1^2 + 14 x0^2 x2^2 e1 - 18 x0^2 x2^2 e1^2 - 76 x0^2 x2^6 x1^4 - 32 x1^2
e1^3 x0 - 6 x2^6 e1 x0 - 32 x2^2 e1^3 x0 - 40 x2^3 e1^2 x1^5 x0 + 12 x2^6
e1^2 x1^4 x0 - 48 e1 x0^2 x1^3 x2 - 10 x1^8 e1 x2^2 x0 + 14 x1^7 e1 x2 x0
- 30 x1^6 e1 x2^4 x0 + 56 x2^3 e1 x1^5 x0 - 30 x2^6 e1 x1^4 x0 + 18 e1^3
x1 x2 - 18 x1^4 x0^2 e1 + 19 x1^4 x0^2 e1^2 + 6 x1^10 x0^2 x2^2 - 4 x1^9
x0^2 x2 + 30 x1^8 x0^2 x2^4 + 3 x1^2 e1^4 x2^2 + 8 x1^2 e1^4 x0 - 14 x1^3
e1^3 x2 - 18 x2^4 x0^2 e1 + 19 x2^4 x0^2 e1^2 + 64 x2^5 x0^2 x1^3 + 48 x2
^6 x0^2 x1^6 - 72 x2^5 x0^2 x1^5 + 30 x2^8 x0^2 x1^4 + 8 x2^2 e1^4 x0 - 1
4 x2^3 e1^3 x1 + 18 x2^6 x1^4 x0 - 4 x0^2 x1 x2 + 40 x1^6 x2^2 x0^2 + 16
x0^2 x1^9 x2^3 - 40 x0^2 x1^7 x2^3 - 32 e1^2 x1^6 x2^2 x0 + 58 e1^2 x1^5
x2 x0 - 64 e1^2 x1^4 x2^4 x0 + 76 x1^4 x0 x2^2 e1^2 + 76 x1^2 e1^2 x2^4 x
0 + 4 x1^8 e1^2 x2^2 x0 - 10 x1^7 e1^2 x2 x0 + 12 x1^6 e1^2 x2^4 x0 + 42
e1 x1^6 x2^2 x0 + 84 e1 x1^4 x2^4 x0 + 14 e1 x0^2 x1 x2 - 80 e1 x0^2 x1^6
x2^2 + 60 e1 x0^2 x1^5 x2 - 158 e1 x0^2 x1^4 x2^4 + 52 e1^2 x0^2 x1^3 x2
- 18 e1^2 x0^2 x1 x2 - 32 e1^3 x0 x1 x2 + 52 e1^2 x0^2 x1^6 x2^2 - 52 e1^
2 x0^2 x1^5 x2 + 103 e1^2 x0^2 x1^4 x2^4 + x1^12 x2^4 x0^2 - 4 x1^11 x2^3
x0^2 + 4 x1^10 x2^6 x0^2 - 16 x1^9 x2^5 x0^2 + 6 x1^8 x2^8 x0^2 + 16 x0^2
x2^9 x1^3 - 24 x0^2 x2^8 x1^2 - 40 x0^2 x2^7 x1^3 + 6 x1^3 x2^7 x0 - 6 x1
^4 x2^2 e1^2 - 2 x1^6 x2^2 e1 - 2 x2^9 x1^3 x0 + 6 x2^8 x1^2 x0 - 6 x2^4
x1^2 e1^2 - 4 x2^4 x1^4 e1 + 10 x0^2 x1^6 e1 + 10 x0^2 x2^6 e1 + 16 x1^4
e1^3 x0 - 10 e1 x1^7 x2^3 x0 - 20 e1 x1^5 x2^5 x0 + 60 x0^2 x2^5 x1 e1 +
114 x0^2 x2^4 x1^6 e1 - 128 x0^2 x2^3 x1^5 e1 - 32 x0^2 x2^7 x1 e1 - 112
x0^2 x2^3 x1^3 e1^2 - 52 x0^2 x2^5 x1 e1^2 + 130 x0^2 x2^3 x1^3 e1 + 56 x
1^3 x2^5 e1 x0 + 14 x2^7 x1 e1 x0 + 90 x0^2 x1^2 x2^4 e1 + 57 x0^2 x1^2 e
1^2 x2^2 + 90 x0^2 x2^2 x1^4 e1 - 54 x0^2 x2^2 x1^2 e1 + 48 x1^2 e1^3 x0
x2^2 - 10 x2^8 e1 x1^2 x0 + 16 x2^4 e1^3 x0 - 6 e1^4 x1 x2 - 2 x1^8 x0^2
e1 - 8 x1^6 x0^2 e1^2 - 2 x1^6 e1^3 x0 + 2 x1^3 e1^4 x2 - 2 x1^4 e1^4 x0
+ 2 x1^5 e1^3 x2 - 2 x2^6 x1^2 e1 + 4 x1^3 e1^3 x2^3 + 6 x2^10 x0^2 x1^2
- 4 x2^9 x0^2 x1 - 2 x2^8 x0^2 e1 - 8 x2^6 x0^2 e1^2 - 2 x2^6 e1^3 x0 + 2
x2^3 e1^4 x1 - 2 x2^4 e1^4 x0 + 10 e1^3 x0^2 x1^2 + 10 e1^3 x0^2 x2^2 - 8
e1^3 x0^2 x1^4 - 8 e1^3 x0^2 x2^4 - 2 e1^4 x0^2 x1^2 - 2 e1^4 x0^2 x2^2 -
32 e1^2 x2^6 x1^2 x0 - 76 x1^4 x0^2 e1^2 x2^2 - 40 x1^3 e1^2 x2^5 x0 - 28
x1^2 e1^3 x2^4 x0 - 6 x1^2 e1^4 x0 x2^2 + 42 e1 x2^6 x1^2 x0 + 4 x2^8 e1^
2 x1^2 x0 + 114 x0^2 x2^6 x1^4 e1 - 128 x0^2 x2^5 x1^3 e1 + 36 x0^2 x2^8
x1^2 e1 - 80 x0^2 x2^6 x1^2 e1 + 108 x0^2 x2^5 x1^5 e1 + 60 x0^2 x2^7 x1^
3 e1 + 52 x0^2 x2^6 x1^2 e1^2 - 76 x0^2 x2^4 x1^2 e1^2 - 10 x2^7 x1^3 e1
x0 - 32 x0^2 x1^7 e1 x2 + 52 e1^3 x0 x1^3 x2 + 52 e1^3 x0 x2^3 x1 - 22 x1
^5 e1^3 x0 x2 - 46 x1^3 e1^3 x0 x2^3 - 10 x2^7 e1^2 x0 x1 - 22 x2^5 e1^3
x0 x1 - 28 e1^3 x1^4 x2^2 x0 - 10 e1^4 x0 x1^3 x2 - 10 e1^4 x0 x2^3 x1 +
8 e1^4 x0 x1 x2 + 6 x1^9 x0^2 e1 x2 + 60 x1^7 x0^2 e1 x2^3 + 36 x1^8 x0^2
e1 x2^2 + 20 x1^7 x0^2 e1^2 x2 + 80 x1^5 x0^2 e1^2 x2^3 + 2 x1^7 e1^3 x0
x2 + 8 x1^5 e1^3 x0 x2^3 + 8 x1^3 e1^3 x2^5 x0 + 6 x1^6 e1^3 x2^2 x0 + 12
x1^4 e1^3 x2^4 x0 + 2 x1^5 e1^4 x0 x2 + 4 x1^3 e1^4 x0 x2^3 + 6 x2^9 x0^2
e1 x1 - 4 x1^10 x2^4 x0^2 - 24 x1^7 x2^7 x0^2 - 12 x1^8 x2^6 x0^2 + 4 x1^
6 x2^10 x0^2 - 16 x1^5 x2^9 x0^2 - 12 x1^6 x2^8 x0^2 + 2 x2^5 e1^3 x1 + x
1^8 e1^2 x0^2 + 2 x1^6 e1^3 x0^2 + x2^12 x1^4 x0^2 - 4 x2^11 x1^3 x0^2 -
4 x2^10 x1^4 x0^2 + x2^8 e1^2 x0^2 + 2 x2^6 e1^3 x0^2 + 2 e1^3 x1^4 x2^2
+ e1^4 x0^2 x1^4 + e1^4 x0^2 x2^4 + x1^6 x2^2 e1^2 - 6 x1^10 x2^2 x0^2 e1
```

```
- 48 x1^6 x2^6 x0^2 e1 + 4 x1^7 x2^3 x0 e1^2 - 16 x1^8 x2^2 x0^2 e1^2 - 5
0 x1^6 x2^4 x0^2 e1^2 + 2 x1^9 x2^3 x0 e1 - 30 x1^8 x2^4 x0^2 e1 - 30 x1^
4 x2^8 x0^2 e1 + 8 x1^5 x2^5 x0 e1^2 - 50 x1^4 x2^6 x0^2 e1^2 + 6 x1^7 x2
^5 x0 e1 + 80 x2^5 x0^2 e1^2 x1^3 + 20 x2^7 x0^2 e1^2 x1 + 2 x2^7 e1^3 x0
x1 + 6 x2^6 e1^3 x1^2 x0 + 2 x2^5 e1^4 x0 x1 - 24 e1^3 x0^2 x1^3 x2 - 24
e1^3 x0^2 x2^3 x1 + 10 e1^3 x0^2 x1 x2 + 18 e1^3 x0^2 x1^5 x2 + 38 e1^3 x
0^2 x1^3 x2^3 - 24 e1^3 x0^2 x1^2 x2^2 + 18 e1^3 x0^2 x2^5 x1 + 24 e1^3 x
0^2 x1^4 x2^2 + 24 e1^3 x0^2 x1^2 x2^4 + 4 e1^4 x0^2 x1^3 x2 + 4 e1^4 x0^
2 x2^3 x1 - 2 e1^4 x0^2 x1 x2 + 2 x1^11 x2^3 x0^2 e1 + 8 x1^9 x2^5 x0^2 e
1 - 16 x1^9 x2^3 x0^2 e1 + 12 x1^7 x2^7 x0^2 e1 - 48 x1^7 x2^5 x0^2 e1 +
2 x1^10 x2^4 x0^2 e1 + 6 x1^8 x2^6 x0^2 e1 + 4 x1^9 x2^3 x0^2 e1^2 + 12 x
1^7 x2^5 x0^2 e1^2 - 24 x1^7 x2^3 x0^2 e1^2 + 6 x1^5 x2^7 x0 e1 + 8 x1^5
x2^9 x0^2 e1 - 48 x1^5 x2^7 x0^2 e1 + 6 x1^6 x2^8 x0^2 e1 + 12 x1^5 x2^7
x0^2 e1^2 - 44 x1^5 x2^5 x0^2 e1^2 - 6 x2^10 x1^2 x0^2 e1 + 4 x2^7 x1^3 x
0 e1^2 - 16 x2^8 x1^2 x0^2 e1^2 + 3 e1^4 x0^2 x1^2 x2^2 + 2 e1^3 x1^2 x2^
4 + 2 x1^4 x2^4 e1^2 + x2^6 x1^2 e1^2 - 2 x1^9 e1^2 x0^2 x2 - 4 x1^7 e1^3
x0^2 x2 - 16 x1^5 e1^3 x0^2 x2^3 + 2 x2^9 x1^3 x0 e1 + 2 x2^11 x1^3 x0^2
e1 - 16 x2^9 x1^3 x0^2 e1 + 2 x2^10 x1^4 x0^2 e1 + 4 x2^9 x1^3 x0^2 e1^2
- 24 x2^7 x1^3 x0^2 e1^2 - 2 x2^9 e1^2 x0^2 x1 - 16 x2^5 e1^3 x0^2 x1^3 -
4 x2^7 e1^3 x0^2 x1 + 2 e1^4 x1^4 x2^2 x0 + 2 e1^4 x1^2 x2^4 x0 - 12 e1^3
x0^2 x1^6 x2^2 - 24 e1^3 x0^2 x1^4 x2^4 - 2 e1^4 x0^2 x1^5 x2 - 4 e1^4 x0
^2 x1^3 x2^3 - 12 e1^3 x0^2 x2^6 x1^2 - 2 e1^4 x0^2 x2^5 x1 + 2 x0^2 x3^2
+ 2 x1^2 x3^2 + 2 x2^2 x3^2 + x1^10 e1^2 x0^2 x2^2 + 5 x1^8 e1^2 x0^2 x2^
4 + 8 x1^6 e1^2 x0^2 x2^6 + 2 x1^8 e1^3 x0^2 x2^2 + 6 x1^6 e1^3 x0^2 x2^4
+ 5 x1^4 e1^2 x0^2 x2^8 + 6 x1^4 e1^3 x0^2 x2^6 + x2^10 e1^2 x0^2 x1^2 +
2 x2^8 e1^3 x0^2 x1^2 + 2 e1^3 x1^7 x2^3 x0^2 + 4 e1^3 x1^5 x2^5 x0^2 + 2
e1^3 x1^3 x2^7 x0^2 + e1^4 x0^2 x1^6 x2^2 + 2 e1^4 x0^2 x1^4 x2^4 - 2 e1^
4 x0^2 x1^4 x2^2 + e1^4 x0^2 x2^6 x1^2 - 2 e1^4 x0^2 x2^4 x1^2
```

This polynomial is huge, having 414 terms and degree 18 in the variables x_0, x_1, x_2, x_3.

Step (3)

We first compute the integer $N = (8md)^{n+2}$. For the case $n = m = d = 2$, we have

$$N = (8 \cdot 2 \cdot 2)^{2+2} = 1048576 \approx 10^6.$$

From this we form the set $\Gamma = \{1, \ldots, N\}^{n+1}$. Clearly the size of this set is N^{n+1}. For the case $n = m = d = 2$, we have

$$|\Gamma| = 1048576^3 = 1152921504606846976 \approx 10^{18}.$$

Note that this analysis depends only on the fact that $n = m = d = 2$. Thus for any input with $n = m = d = 2$, this algorithm requires that $|\Gamma| \approx 10^{18}$.

The set $\mathcal{I}'$ is initialized as an empty set here, and is augmented iteratively at Step (4.2.3). At the beginning of Step (5), this set becomes a representative set for the equation $g = 0$.

Step (4)

This step iterates the sub–steps (4.1) and (4.2) for each $\gamma = (\gamma_1, \ldots, \gamma_{n+1}) \in \Gamma$. Therefore the number of iterations is just the size of the set Γ. Recalling the analysis of the last step, we conclude that for the case $n = m = d = 2$, the number of iterations in Step (4) is

$$1152921504606846976 \approx 10^{18}.$$

Now each iteration requires various expensive computations. For example, In Step (4.1) the following system must be solved over $\bar{F}^{n+2}$:

$$g - \epsilon = \left(\frac{\partial g}{\partial x_1}\right)^2 - \frac{\gamma_1}{N(n+2)}\Delta = \cdots = \left(\frac{\partial g}{\partial x_{n+1}}\right)^2 - \frac{\gamma_{n+1}}{N(n+2)}\Delta = 0$$

where $\Delta = \sum_{j=0}^{n+1} \left(\frac{\partial g}{\partial x_j}\right)^2$. Recalling that the degree of g is bounded by $2(md + kd + 1)$, we know that the degrees of the polynomials in the above system are bounded by $2(2(md + kd + 1) - 1) = 4(md + kd) + 2$. For the example input, the degree bound is

$$4(2 \cdot 2 + 2 \cdot 2) + 2 = 34.$$

In order to get concrete impression on the size of the polynomials in the above system, we actually computed one of the polynomials, namely

$$\left(\frac{\partial g}{\partial x_1}\right)^2 - \frac{\gamma_1}{N(n+2)}\Delta$$

for the iteration where $\gamma = \{1, 1, 1\}$. The trace shows (not included here because it is huge) that the polynomial is indeed of degree 34 with 5784 terms. It took about 20 seconds on MAPLE just to construct the polynomial in distributed representation. Thus one can easily expect that solving the above system would take much computation time.

Any algorithm for solving a system of equations must at least read in every term appearing in the system. Experiments with several sample input polynomials suggest that the average number of terms in a system is at least 10000. Clearly it takes at least one machine instruction to read in a term. For any currently available workstations, it takes at least 10^{-8} seconds per instruction. (SUN-4 Sparc station and DEC-Station 5000 take about 10^{-7} seconds per instruction.) Therefore just reading in a system of equations takes at least $10000 \cdot 10^{-8} = 10^{-4}$ seconds, and thus solving a system should take at least 10^{-4} seconds also, which is *very far under–estimate*. But then there are 10^{18} systems to solve, thus we estimate that Step (4) will take at least

$$10^{14} \text{ seconds } \approx 3 \text{ million years.}$$

Step (5) through Step (9)

Since Step (4) takes enormous computation time, the analysis of the subsequent steps does not seem to be necessary, and thus we will not investigate their complexities.

8 Analysis of Renegar's Algorithm

Now we analyze the complexity of Renegar's algorithm following its steps one after the other. Though we are mostly interested in the behavior of the algorithm on the inputs with $n = m = d = L = 2$, we will also try to analyze it for arbitrary inputs, whenever possible.

Step (1)

We begin with two input polynomials g_1 and g_2 of two variables x_1 and x_2 of degrees two. For the example input, they are

```
g1  = x1^2 + x2^2 - 1
g2  = x1 x2 - 1.
```

From these we form $(6m + 2 = 14)$ polynomials h_i in $(n + 1 = 3)$ variables of degree $(d + 1 = 3)$. For the example input, they are

```
h1  = g1
    = x1^2 + x2^2 - 1
h2  = g2
    = x1 x2 - 1
h3  = x0 g1 - 1
    = x0 x1^2 + x0 x2^2 - x0 - 1
h4  = x0 g2 - 1
    = x0 x1 x2 - x0 - 1
h5  = x0 g1 + 1
    = x0 x1^2 + x0 x2^2 - x0 + 1
h6  = x0 g2 + 1
    = x0 x1 x2 - x0 + 1
h7  = x0 - 1
h8  = - g1
    = - x1^2 - x2^2 + 1
h9  = - g2
    = - x1 x2 + 1
h10 = - x0 g1 + 1
    = - x0 x1^2 - x0 x2^2 + x0 + 1
h11 = - x0 g2 + 1
    = - x0 x1 x2 + x0 + 1
h12 = - x0 g1 - 1
    = - x0 x1^2 - x0 x2^2 + x0 - 1
h13 = - x0 g2 - 1
    = - x0 x1 x2 + x0 - 1
h14 = - x0 + 1.
```

Step (2)

Since $d = 2$, the integer $d' = 4$. Now we form the polynomials

$$\bar{h}_i(\delta; x_0, \ldots, x_n) = (1-\delta)h_i + \delta(1 + \sum_{j=0}^{n} i^j x_j^{d'})$$

for each $i = 1, \ldots, 14$. For the example input, they are

```
hb1  = (1 - d) h1 + d (1 + 1^0 x0^dp + 1^1 x1^dp + 1^2 x2^dp)
     = x1^2 + x2^2 - 1 - d x1^2 - d x2^2 + 2 d + d x0^4 + d x1^4 + d x2^4
hb2  = (1 - d) h2 + d (1 + 2^0 x0^dp + 2^1 x1^dp + 2^2 x2^dp);
     = x1 x2 - 1 - d x1 x2 + 2 d + d x0^4 + 2 d x1^4 + 4 d x2^4
hb3  = (1 - d) h3 + d (1 + 3^0 x0^dp + 3^1 x1^dp + 3^2 x2^dp);
     = x0 x1^2 + x0 x2^2 - x0 - 1 - d x0 x1^2 - d x0 x2^2 + d x0 + 2 d + d x0^4
      + 3 d x1^4 + 9 d x2^4
hb4  = (1 - d) h4 + d (1 + 4^0 x0^dp + 4^1 x1^dp + 4^2 x2^dp)
     = x0 x1 x2 - x0 - 1 - d x0 x1 x2 + d x0 + 2 d + d x0^4 + 4 d x1^4
       + 16 d x2^4
hb5  = (1 - d) h5 + d (1 + 5^0 x0^dp + 5^1 x1^dp + 5^2 x2^dp)
     = x0 x1^2 + x0 x2^2 - x0 + 1 - d x0 x1^2 - d x0 x2^2 + d x0 + d x0^4
       + 5 d x1^4 + 25 d x2^4
hb6  = (1 - d) h6 + d (1 + 6^0 x0^dp + 6^1 x1^dp + 6^2 x2^dp)
     = x0 x1 x2 - x0 + 1 - d x0 x1 x2 + d x0 + d x0^4 + 6 d x1^4 + 36 d x2^4
hb7  = (1 - d) h7 + d (1 + 7^0 x0^dp + 7^1 x1^dp + 7^2 x2^dp)
     = x0 - 1 - d x0 + 2 d + d x0^4 + 7 d x1^4 + 49 d x2^4
hb8  = (1 - d) h8 + d (1 + 8^0 x0^dp + 8^1 x1^dp + 8^2 x2^dp)
     =  - x1^2 - x2^2 + 1 + d x1^2 + d x2^2 + d x0^4 + 8 d x1^4 + 64 d x2^4
hb9  = (1 - d) h9 + d (1 + 9^0 x0^dp + 9^1 x1^dp + 9^2 x2^dp)
     =  - x1 x2 + 1 + d x1 x2 + d x0^4 + 9 d x1^4 + 81 d x2^4
hb10 = (1 - d) h10 + d (1 + 10^0 x0^dp + 10^1 x1^dp + 10^2 x2^dp)
     = - x0 x1^2 - x0 x2^2 + x0 + 1 + d x0 x1^2 + d x0 x2^2 - d x0 + d x0^4
       + 10 d x1^4 + 100 d x2^4
hb11 = (1 - d) h11 + d (1 + 11^0 x0^dp + 11^1 x1^dp + 11^2 x2^dp)
     = - x0 x1 x2 + x0 + 1 + d x0 x1 x2 - d x0 + d x0^4 + 11 d x1^4 + 121 d x2^4
hb12 = (1 - d) h12 + d (1 + 12^0 x0^dp + 12^1 x1^dp + 12^2 x2^dp)
     = - x0 x1^2 - x0 x2^2 + x0 - 1 + d x0 x1^2 + d x0 x2^2 - d x0 + 2 d
       + d x0^4 + 12 d x1^4 + 144 d x2^4
hb13 = (1 - d) h13 + d (1 + 13^0 x0^dp + 13^1 x1^dp + 13^2 x2^dp)
     = - x0 x1 x2 + x0 - 1 + d x0 x1 x2 - d x0 + 2 d + d x0^4 + 13 d x1^4
       + 169 d x2^4
hb14 = (1 - d) h14 + d (1 + 14^0 x0^dp + 14^1 x1^dp + 14^2 x2^dp)
     = - x0 + 1 + d x0 + d x0^4 + 14 d x1^4 + 196 d x2^4
```

Step (3)

This step iterates the sub-steps (3.1)—(3.5) for each $A \subseteq \{1, \ldots, 6m+2\}$ such that $|A| \leq n+1$. Thus generally the number of iterations is

$$\binom{6m+2}{0} + \binom{6m+2}{1} + \cdots + \binom{6m+2}{n+1}$$

in case $6m + 2 \geq n + 1$. For $n = m = 2$, the number of iterations is

$$\binom{14}{0} + \binom{14}{1} + \binom{14}{2} + \binom{14}{3} = 470.$$

It will be too time-consuming to trace all the 470 iterations (at least for this interactive session), thus whenever necessary we will consider only the case

$$A = \{3, 4\}.$$

Step (3.1)

We first form the polynomial $\hat{h}(x_0, \ldots, x_n) = \sum_{j=0}^{n}(6m+3)^j x_j^{d'}$. For the example input, it is

```
hh = (6 2 + 3)^0 x0^dp + (6 2 + 3)^1 x1^dp + (6 2 + 3)^2 x2^dp
   = x0^4 + 15 x1^4 + 225 x2^4.
```

Next we form the matrix M_A with the last row $\nabla_{x_0,\ldots,x_n}\hat{h}$ and with earlier rows $\nabla_{x_0,\ldots,x_n}\bar{h}_i$, $i \in A$, ordered by increasing indices i. For the example input and $A = \{3, 4\}$, we have

```
MA1 = grad(hb3,[x0,x1,x2])
MA2 = grad(hb4,[x0,x1,x2])
MA3 = grad(hh,[x0,x1,x2])
MA  = [MA1,MA2,MA3]

where

MA[1,1] = x1^2 + x2^2 - 1 - d x1^2 - d x2^2 + d + 4 d x0^3
MA[1,2] = 2 x0 x1 - 2 x0 x1 d + 12 d x1^3
MA[1,3] = 2 x0 x2 - 2 x0 x2 d + 36 d x2^3

MA[2,1] = x1 x2 - 1 - d x1 x2 + d + 4 d x0^3
MA[2,2] = x0 x2 - x0 x2 d + 16 d x1^3
MA[2,3] = x0 x1 - x0 x1 d + 64 d x2^3

MA[3,1] = 4 x0^3
MA[3,2] = 60 x1^3
MA[3,3] = 900 x2^3
```

Step (3.2)

Now we compute the polynomial $h_A = \det(M_A M_A^T) + \sum_{i \in A} \bar{h}_i^2$. An easy check on the size of the matrix M_A and the degree of its entries shows that d_A is (very tightly) bounded above by $2(d+1)(|A|+1)$. For $n = m = d = 2$ and $A = \{3, 4\}$, we have

$$d_{\{3,4\}} = 2(2+1)(2+1) = 18.$$

For the example input, here is the polynomial $h_{\{3,4\}}$:

```
hA = det(multiply(MA,transpose(MA))) + hb3^2 + hb4^2
   = 23040 x1^10 d x0^5 + 6 d x0 x1^2 - 69120 d^2 x1^10 x0^5 + 4 x0^2 x2^2 d
     + 18 x0 x2^6 d + 384 x0^10 x2^4 d^2 - 14400 x0^2 x1^8 d - 256 x0^10 x2^4
     d - 55296 x0^9 x2^6 d + 14 x0 x1^4 d^2 + 2 + 6 d x0 x2^2 - 14400 d x1^12
     x0^2 - 50 x0 x2^4 d + 18662400 d^3 x2^10 x0^5 - 9216 x0^9 x1^6 d^3 - 144
     00 d^4 x2^8 x0^6 + 50 x0 x2^4 d^2 - 2 x0^5 x2^2 d^2 + 18662400 x0^5 x2^8
```

```
d^2 + 2 x0^5 x2^2 d + 2 x0^5 x1^2 d - 8 d - 3240000 x0^2 x2^8 d - 2 x0^2
x1^2 d^2 + 3840 x0^6 x1^8 d + 57600 d^3 x2^8 x0^6 + 3 x0^2 x1^2 x2^2 - 1
8662400 d^2 x2^10 x0^5 - 4 d^2 x0 x1^2 + 21600 x0^2 x1^8 d^2 - 18662400
x0^5 x2^8 d^3 + 111600 x1^8 x0^2 x2^4 + 810000 d^4 x2^12 x0^2 + 3072 x0^
9 x1^6 d^4 + 4 x0 - 2 x0 x1^2 - 2 x0 x2^2 - 4 d x0^4 - 14 d x1^4 - 50 d
x2^4 - 2 x0^5 x1^2 d^2 - 2 x0^2 x2^4 d - 4 d^2 x0 x2^2 - 6 d^2 x0 x1^6 +
d^2 x0^2 x2^4 + 64 x0^10 x2^4 d^4 + 6 x0 x1^6 d - 2 x0 x1 x2 + 2 x0^2 +
4 x0^2 x1^2 d - 18 d^2 x0 x2^6 + 182 d^2 x1^4 x2^4 + 7200 x2^2 x0^2 x1^8
- 4 d x0^2 - 182476800 x1^9 x2^7 d^2 - 2 x0^2 x1^4 d - 4 d x0^5 - 12 d x
0 + d^2 x0^2 x1^4 + 2 d^2 x0^2 - 23887872 x0^8 x2^8 d^3 + 8 d^2 x0 - 2 x
0^2 x1 x2 + 4 d^2 x0^5 - 3072 x0^9 x1^6 d - 540000 x1^4 x0^2 x2^4 - 14 x
0 x1^4 d - 2 x0^2 x2^2 d^2 + 6 d x0 x1 x2 - 256 x0^10 x2^4 d^3 - 4055040
0 d^4 x1^8 x2^6 + 9216 x0^9 x1^6 d^2 - 14400 x0^6 x1^2 d^4 x2^4 - 622080
0 d^4 x2^10 x0^5 - 3240000 d x2^12 x0^2 + 4860000 x0^2 x2^8 d^2 + 8 d^2
x0^4 + 1620000 x1^2 x0^2 x2^8 - 7200 d^4 x1^10 x0^2 + 2 d^2 x0^8 + x0^2
x1^4 + 28 d^2 x1^4 + 100 d^2 x2^4 + 8 d^2 - 2 x0^2 x1^2 - 2 x0^2 x2^2 +
4 x0^2 x1 x2 d - 2 x0^2 x1 d^2 x2 + 5760 d^2 x0^6 x1^6 + 86400 d^2 x0^6
x2^6 + 3240000 x0^2 x1^2 x2^6 + 3686400 d^2 x1^6 x2^6 + 14400 x0^2 x2^2
x1^6 - 14400 x0^2 x1^7 x2 - 3240000 x0^2 x1 x2^7 + 216000 x1^5 x0^2 x2^3
+ 14 x0^4 x1^4 d^2 + 216000 x2^5 x0^2 x1^3 + 50 x0^4 x2^4 d^2 + 3600 x0^
2 x1^8 + 810000 x0^2 x2^8 + 25 d^2 x1^8 + 337 d^2 x2^8 - 1267200 x1^11 d
x2^3 x0 - 57600 x0^2 x2^2 d x1^6 + 19440000 x0^2 x1^2 d^2 x2^6 - 1296000
0 x0^2 x1^2 d x2^6 + 86400 x0^2 x2^2 d^2 x1^6 + 3456000 d x1^3 x0 x2^7 -
10368000 d^2 x1^3 x0 x2^7 - 230400 d x2^3 x0 x1^7 + 691200 d^2 x2^3 x0 x
1^7 - 960 x0^6 x1^8 - 14400 x0^6 x2^8 + 2160000 x1^4 x0^2 x2^4 d - 32400
00 d^2 x1^4 x0^2 x2^4 + x0^2 x2^4 + 20736000 x0 x1^4 d^2 x2^6 + 460800 x
0 x2^4 d x1^6 - 1382400 x0 x2^4 d^2 x1^6 - 6912000 x0 x1^4 d x2^6 + 5760
0 x0^2 x1^7 x2 d + 12960000 x0^2 x1 x2^7 d - 86400 x0^2 x1^7 d^2 x2 - 19
440000 x0^2 x1 d^2 x2^7 - 864000 x1^5 x0^2 d x2^3 + 1296000 x0^2 x1^5 d^
2 x2^3 - 864000 x2^5 x0^2 d x1^3 + 1296000 x0^2 x2^5 d^2 x1^3 - 3840 d^3
x0^6 x1^6 - 57600 d^3 x0^6 x2^6 - 7372800 d^3 x1^6 x2^6 + 57600 x0^2 x1^
7 d^3 x2 - 864000 x0^2 x1^5 d^3 x2^3 - 20736000 x0 x1^4 d^3 x2^6 - 86400
0 x0^2 x2^5 d^3 x1^3 + 1382400 x0 x2^4 d^3 x1^6 + 12960000 x0^2 x2^7 d^3
x1 + 6 x1^4 x2^2 d x0 + 18 x1^2 x2^4 d x0 + 8 x1^5 d x0 x2 + 32 x2^5 d x
0 x1 - 8 d^2 x1^5 x0 x2 - 32 d^2 x2^5 x0 x1 - 6 d^2 x1^4 x2^2 x0 - 18 d^
2 x1^2 x2^4 x0 + 2 x0^5 x1 x2 d - 691200 d^3 x1^7 x2^3 x0 + 10368000 d^3
x2^7 x1^3 x0 - 2 x0^5 x1 x2 d^2 - 57600 d^3 x1^6 x0^2 x2^2 + 2160000 d^3
x1^4 x0^2 x2^4 - 12960000 d^3 x2^6 x0^2 x1^2 - 14400 x1^8 d^3 x0^2 - 324
0000 x2^8 d^3 x0^2 - 165888 x0^9 x2^6 d^3 + 165888 x0^9 x2^6 d^2 + 55296
x0^9 x2^6 d^4 + 297676800 d^4 x1^8 x2^8 + 28800 x1^10 x0^2 d - 69120 d^3
x2^2 x0^5 x1^8 - 7680 d^3 x2^2 x0^6 x1^6 + 23040 d^4 x2^2 x0^5 x1^8 + 19
20 d^4 x2^2 x0^6 x1^6 - 622080 d^2 x2^6 x0^5 x1^4 - 178421760 d^4 x2^8 x
0^3 x1^6 - 122880 x0^6 x2^3 d x1^3 - 768 x0^10 x2^2 d^2 x1^2 + 19440000
x0^2 x2^9 d^2 x1 + 115200 x0^6 x2^5 d x1 - 9720000 x0^2 x2^10 d^2 x1^2 +
253440 x0^5 x2^2 d^2 x1^6 + 6480000 x0^2 x2^10 d^3 x1^2 + 921600 x0^2 x2
^3 d^3 x1^7 - 115200 x0^6 x2^6 d x1^2 - 12960000 x0^2 x2^9 d^3 x1 - 2534
40 x0^5 x2^2 d^3 x1^6 + 512 x0^10 x2^2 d^3 x1^2 - 446400 x0^2 x2^4 d^3 x
1^8 + 669600 x0^2 x2^4 d^2 x1^8 + 61440 x0^6 x2^4 d x1^4 - 1382400 x0^2
x2^3 d^2 x1^7 - 247808 x0^8 x2^2 d^3 x1^6 - 264960 x0^5 x1^7 d^2 x2 - 30
72 x0^9 x1^4 d^4 x2^2 - 139968000 x0 x1^5 d^3 x2^7 + 4573440 x0^5 x1^5 d
^2 x2^3 + 46656000 x0 x1^6 d^3 x2^8 - 112189440 x0^4 x1^4 d^3 x2^6 - 155
52000 x0 x1^6 d^4 x2^8 + 622080 x0^5 x1^6 d^2 x2^4 + 46656000 x0 x1^5 d^
4 x2^7 + 9216 x0^9 x1^4 d^3 x2^2 + 69120 x0^5 x1^8 d^2 x2^2 + 6912000 x0
x1^4 d^4 x2^6 + 56094720 x0^4 x1^4 d^4 x2^6 + 113000448 x0^7 x1^4 d^4 x2
^6 - 1432320 x0^5 x2^5 d x1^3 - 165888 x0^9 x2^4 d^2 x1^2 + 24480000 x0^
```

```
5 x2^7 d x1 - 3379200 x0^4 x2^4 d^2 x1^6 + 9331200 x0 x2^5 d^3 x1^7 - 62
20800 x0^5 x2^8 d x1^2 + 6758400 x0^4 x2^4 d^3 x1^6 + 165888 x0^9 x2^4 d
^3 x1^2 - 3110400 x0 x2^6 d^3 x1^8 + 3110400 x0 x2^6 d^2 x1^8 + 207360 x
0^5 x2^6 d x1^4 - 9331200 x0 x2^5 d^2 x1^7 + 5947392 x0^7 x2^4 d^3 x1^6
+ 4296960 x0^5 x2^5 d^2 x1^3 - 73440000 x0^5 x2^7 d^2 x1 - 3110400 x0 x2
^5 d^4 x1^7 + 18662400 x0^5 x2^8 d^2 x1^2 - 3379200 x0^4 x2^4 d^4 x1^6 -
55296 x0^9 x2^4 d^4 x1^2 + 1036800 x0 x2^6 d^4 x1^8 - 460800 x0 x2^4 d^4
x1^6 - 5947392 x0^7 x2^4 d^4 x1^6 + 28800 d^3 x1^10 x2^2 x0^2 + 56094720
d^2 x1^4 x2^6 x0^4 - 7200 d^4 x1^10 x2^2 x0^2 - 1296000 d^2 x1^6 x2^6 x0
^2 - 1620000 d^4 x1^2 x2^10 x0^2 - 6480000 x1^2 x0^2 x2^8 d - 43200 x1^1
0 x0^2 d^2 + 960 x0^6 x1^6 - 5760 x1^8 d^2 x0^6 - 40550400 x1^8 d^2 x2^6
+ 6480000 x2^10 x0^2 d - 9720000 x2^10 x0^2 d^2 - 86400 x2^8 d^2 x0^6 -
40550400 x2^8 d^2 x1^6 - 3840 x0^6 d x1^6 - 256 x0^10 x1^4 d^3 - 69120 x
0^5 x1^8 d^3 + 23040 x0^5 x1^8 d^4 + 64 x0^10 x1^4 d^4 + 3600 x0^2 x1^8
d^4 + 36864 x0^8 x1^8 d^4 - 595353600 x1^8 d^3 x2^8 + 69120 d^3 x1^10 x0
^5 + 3840 d^3 x1^8 x0^6 - 23040 d^4 x1^10 x0^5 - 960 d^4 x1^8 x0^6 - 576
00 x0^6 d x2^6 + 28800 d^3 x1^10 x0^2 + 81100800 d^3 x1^8 x2^6 + 14400 x
0^6 x2^6 + 6480000 d^3 x2^10 x0^2 + 81100800 d^3 x2^8 x1^6 + 21600 d^2 x
1^12 x0^2 - 14400 d^3 x1^12 x0^2 - 223027200 d^3 x1^10 x2^6 + 4860000 d^
2 x2^12 x0^2 + 9720000 x1^2 x0^2 x2^8 d^2 + 172800 x1^2 d^2 x0^6 x2^6 -
28800 x2^2 x0^2 x1^8 d + 43200 x2^2 x0^2 x1^8 d^2 + 11520 x2^2 d^2 x0^6
x1^6 - 11520 x0^6 x1^5 d^2 x2 - 128 x0^10 x1^2 d^4 x2^2 + 13824000 x0^2
x1^3 d^3 x2^7 + 184320 x0^6 x1^3 d^2 x2^3 - 3672000 x0^2 x1^4 d^3 x2^8 -
72230400 x0^5 x1^2 d^3 x2^6 + 918000 x0^2 x1^4 d^4 x2^8 + 14400 x0^2 x1^
9 d^4 x2 - 92160 x0^6 x1^4 d^2 x2^4 - 3456000 x0^2 x1^3 d^4 x2^7 - 57600
x0^2 x1^9 d^3 x2 + 3240000 x0^2 x1^2 d^4 x2^6 + 24076800 x0^5 x1^2 d^4 x
2^6 + 44729344 x0^8 x1^2 d^4 x2^6 - 43200 x1^10 d^2 x2^2 x0^2 + 864000 x
1^6 d x2^6 x0^2 - 115200 d^3 x1^2 x0^6 x2^6 - 18662400 d^3 x1^2 x0^5 x2^
8 + 6220800 d^4 x1^2 x0^5 x2^8 + 28800 d^4 x1^2 x0^6 x2^6 - 178421760 d^
4 x1^8 x0^3 x2^6 - 6480000 d^3 x1^2 x0^2 x2^8 - 28800 d^3 x2^2 x0^2 x1^8
+ 5508000 d^2 x1^4 x0^2 x2^8 - 446400 d x1^8 x0^2 x2^4 - 7200 x1^10 x0^2
- 1620000 x2^10 x0^2 + 3600 x1^12 x0^2 + 810000 x2^12 x0^2 + 64 x0^10 x1
^4 + 64 x0^10 x2^4 - 3240000 d^3 x2^12 x0^2 - 223027200 d^3 x2^10 x1^6 +
384 x0^10 x1^4 d^2 + 69120 x0^5 x1^8 d^2 - 73728 x0^8 x1^8 d^3 - 256 x0^
10 x1^4 d - 3672000 d x2^8 x1^4 x0^2 + 7680 x0^6 x1^5 d x2 - 20736000 x0
^2 x1^3 d^2 x2^7 + 72230400 x0^5 x1^2 d^2 x2^6 - 7680 x0^6 x1^6 d x2^2 +
86400 x0^2 x1^9 d^2 x2 - 89458688 x0^8 x1^2 d^3 x2^6 - 176394240 d^3 x2^
9 x0^4 x1^3 + 88197120 d^4 x2^9 x0^4 x1^3 + 18247680 d^3 x2^5 x0^4 x1^7
- 9123840 d^4 x2^5 x0^4 x1^7 + 622080 d^3 x2^6 x0^5 x1^4 - 207360 d^4 x2
^6 x0^5 x1^4 - 12960000 x0^2 x2^9 d x1 + 6480000 x0^2 x2^10 d x1^2 + 512
x0^10 x2^2 d x1^2 + 921600 x0^2 x2^3 d x1^7 - 3041280 x0^5 x2^3 d^2 x1^7
+ 36115200 x0^5 x2^9 d^2 x1 + 3041280 x0^5 x2^3 d^3 x1^7 - 36115200 x0^5
x2^9 d^3 x1 + 337920 x0^9 x2^3 d^2 x1^3 - 337920 x0^9 x2^3 d^3 x1^3 - 32
1024 x0^9 x2^5 d^2 x1 + 321024 x0^9 x2^5 d^3 x1 - 172800 x0^6 x2^5 d^2 x
1 + 61440 x0^6 x2^4 d^3 x1^4 - 122880 x0^6 x2^3 d^3 x1^3 + 115200 x0^6 x
2^5 d^3 x1 + 139968000 x0 x1^5 d^2 x2^7 - 46656000 x0 x1^6 d^2 x2^8 - 92
16 x0^9 x1^4 d^2 x2^2 - 346705920 x0^4 x1^5 d^3 x2^7 + 173352960 x0^4 x1
^5 d^4 x2^7 + 270336 x0^8 x1^7 d^3 x2 - 135168 x0^8 x1^7 d^4 x2 - 540672
0 x0^8 x1^5 d^3 x2^3 + 2703360 x0^8 x1^5 d^4 x2^3 - 622080 x0^5 x1^6 d^3
x2^4 + 264960 x0^5 x1^7 d^3 x2 - 4573440 x0^5 x1^5 d^3 x2^3 + 207360 x0^
5 x1^6 d^4 x2^4 - 88320 x0^5 x1^7 d^4 x2 + 1524480 x0^5 x1^5 d^4 x2^3 -
1036800 x0 x2^6 d x1^8 + 55296 x0^9 x2^4 d x1^2 + 3110400 x0 x2^5 d x1^7
- 9123840 x0^4 x2^5 d^2 x1^7 + 2297856 x0^8 x2^5 d^2 x1^3 - 4595712 x0^8
x2^5 d^3 x1^3 - 46227456 x0^8 x2^7 d^2 x1 + 92454912 x0^8 x2^7 d^3 x1 -
4296960 x0^5 x2^5 d^3 x1^3 + 73440000 x0^5 x2^7 d^3 x1 + 2297856 x0^8 x2
```

```
^5 d^4 x1^3 - 46227456 x0^8 x2^7 d^4 x1 + 1432320 x0^5 x2^5 d^4 x1^3 - 2
4480000 x0^5 x2^7 d^4 x1 + 864000 d^3 x1^6 x2^6 x0^2 - 216000 d^4 x1^6 x
2^6 x0^2 - 46656000 x1^5 d x0 x2^7 + 1497600 x1^9 d x2^3 x0 - 4492800 x1
^9 d^2 x2^3 x0 - 22464000 x2^9 d x1^3 x0 + 67392000 x2^9 d^2 x1^3 x0 - 1
26720 x0^5 x1^9 d^3 x2 + 37509120 x0^5 x1^3 d^3 x2^7 + 42240 x0^5 x1^9 d
^4 x2 - 12503040 x0^5 x1^3 d^4 x2^7 + 16896 x0^9 x1^5 d^3 x2 - 5632 x0^9
x1^5 d^4 x2 + 112640 x0^9 x1^3 d^4 x2^3 + 7680 x0^6 x1^5 d^3 x2 - 15360
x0^6 x1^4 d^4 x2^4 - 1920 x0^6 x1^5 d^4 x2 + 30720 x0^6 x1^3 d^4 x2^3 +
28800 x1^10 d x2^2 x0^2 + 10137600 d^3 x1^9 x0^4 x2^3 - 5068800 d^4 x1^9
x0^4 x2^3 + 4492800 d^3 x1^9 x2^3 x0 - 67392000 d^3 x2^9 x1^3 x0 + 53222
400 d^2 x1^7 x0 x2^7 - 53222400 d^3 x1^7 x0 x2^7 + 3801600 d^2 x1^11 x2^
3 x0 - 3801600 d^3 x1^11 x2^3 x0 - 57024000 d^2 x2^11 x1^3 x0 + 57024000
d^3 x2^11 x1^3 x0 + 13824000 x0^2 x1^3 d x2^7 - 57600 x0^2 x1^9 d x2 + 1
26720 x0^5 x1^9 d^2 x2 - 37509120 x0^5 x1^3 d^2 x2^7 - 16896 x0^9 x1^5 d
^2 x2 + 88320 x0^5 x1^7 d x2 - 1524480 x0^5 x1^5 d x2^3 - 207360 x0^5 x1
^6 d x2^4 - 230400 x0^2 x2^3 d^4 x1^7 + 3240000 x0^2 x2^9 d^4 x1 + 84480
x0^5 x2^2 d^4 x1^6 - 40550400 d^4 x1^6 x2^8 + 3600 d^4 x1^12 x0^2 + 1115
13600 x1^10 d^2 x2^6 + 918000 x2^8 x1^4 x0^2 + 111513600 x2^10 d^2 x1^6
+ 810000 x0^2 x2^8 d^4 + 6220800 x0^5 x2^8 d^4 + 11943936 x0^8 x2^8 d^4
- 216000 x1^6 x2^6 x0^2 + 297676800 x1^8 x2^8 d^2 + 111513600 d^4 x1^10
x2^6 + 960 d^4 x0^6 x1^6 - 1920 x0^6 x1^5 x2 + 30720 x0^6 x1^3 x2^3 - 23
040 x0^5 x1^8 d x2^2 - 113000448 x0^7 x1^4 d^3 x2^6 + 111600 x0^2 x2^4 d
^4 x1^8 + 14400 x0^2 x2^2 d^4 x1^6 + 123904 x0^8 x2^2 d^4 x1^6 + 3632640
d x0^5 x1^4 x2^4 - 32440320 d^3 x0^3 x1^6 x2^6 + 178421760 x1^8 d^3 x0^3
x2^6 + 178421760 x2^8 d^3 x0^3 x1^6 + 7200 d^4 x2^2 x0^2 x1^8 - 24076800
x0^5 x1^2 d x2^6 - 23040 x0^5 x1^8 d - 15360 x0^6 x1^4 x2^4 - 66355200 d
^3 x1^7 x2^7 + 33177600 d^4 x1^7 x2^7 - 28800 x0^6 x2^5 x1 - 6220800 x0^
5 x2^8 d + 28800 x0^6 x2^6 x1^2 + 111513600 d^4 x2^10 x1^6 - 1620000 d^4
x2^10 x0^2 + 14400 d^4 x0^6 x2^6 + 3686400 d^4 x1^6 x2^6 + 1920 x0^6 x1^
6 x2^2 + 36864 x0^8 x1^8 d^2 + 11943936 x0^8 x2^8 d^2 + 6220800 x2^10 d
x0^5 + 57600 x0^6 x2^8 d - 7200 x1^10 x2^2 x0^2 - 1620000 x1^2 x2^10 x0^
2 - 960 x0^6 x1^4 d^4 x2^2 + 1751040 d^2 x1^7 x0^4 x2^3 + 10752000 d^2 x
1^8 x0^4 x2^4 - 84480 x0^5 x2^2 d x1^6 - 28477440 d^2 x2^7 x0^4 x1^3 + 5
7600 x0^6 x1^2 d^3 x2^4 + 32440320 d^4 x2^6 x0^3 x1^6 + 1620000 d^4 x1^2
x0^2 x2^8 - 10897920 d^2 x0^5 x1^4 x2^4 - 6035456 d^2 x0^8 x1^4 x2^4 + 7
1368704 d^4 x0^6 x1^6 x2^6 + 44729344 x0^8 x1^2 d^2 x2^6 + 123904 x0^8 x
2^2 d^2 x1^6 - 200970240 d^2 x2^8 x0^4 x1^4 - 17740800 x1^7 d x0 x2^7 +
19008000 x2^11 d x1^3 x0 + 15552000 x0 x1^6 d x2^8 + 3072 x0^9 x1^4 d x2
^2 + 173352960 x0^4 x1^5 d^2 x2^7 - 135168 x0^8 x1^7 d^2 x2 - 1013760 x0
^5 x2^3 d^4 x1^7 + 12038400 x0^5 x2^9 d^4 x1 + 88197120 x2^9 d^2 x0^4 x1
^3 + 2703360 x0^8 x1^5 d^2 x2^3 - 107008 x0^9 x2^5 d^4 x1 - 28800 x0^6 x
2^5 d^4 x1 + 56954880 d^3 x0^4 x1^3 x2^7 - 3502080 d^3 x0^4 x2^3 x1^7 +
10897920 d^3 x0^5 x1^4 x2^4 - 5068800 x1^9 d^2 x0^4 x2^3 + 17740800 d^4
x1^7 x0 x2^7 + 1267200 d^4 x1^11 x2^3 x0 - 3456000 x0^2 x1^3 x2^7 - 128
x0^10 x1^2 x2^2 + 14400 x0^2 x1^9 x2 + 33177600 d^2 x1^7 x2^7 + 3240000
x0^2 x2^9 x1 - 230400 x0^2 x2^3 x1^7 - 182476800 d^4 x1^9 x2^7 - 1824768
00 d^4 x2^9 x1^7 + 58392576 d^3 x0^7 x1^3 x2^7 - 42240 x0^5 x1^9 d x2 +
12503040 x0^5 x1^3 d x2^7 + 5632 x0^9 x1^5 d x2 - 145981440 d^3 x1^7 x0^
3 x2^7 + 145981440 d^4 x1^7 x0^3 x2^7 + 1013760 x0^5 x2^3 d x1^7 - 12038
400 x0^5 x2^9 d x1 - 112640 x0^9 x2^3 d x1^3 - 1497600 d^4 x1^9 x2^3 x0
- 19008000 d^4 x2^11 x1^3 x0 + 22464000 d^4 x2^9 x1^3 x0 - 3456000 d^4 x
1^3 x0 x2^7 + 230400 d^4 x2^3 x0 x1^7 - 540000 d^4 x1^4 x0^2 x2^4 - 2847
7440 d^4 x0^4 x1^3 x2^7 + 1751040 d^4 x0^4 x2^3 x1^7 - 3632640 d^4 x0^5
x1^4 x2^4 - 58392576 d^4 x0^7 x1^3 x2^7 - 3244032 d^3 x0^7 x2^3 x1^7 + 3
244032 d^4 x0^7 x2^3 x1^7 + 12070912 d^3 x0^8 x1^4 x2^4 - 6035456 d^4 x0
```

```
^8 x1^4 x2^4 - 21504000 d^3 x1^8 x0^4 x2^4 + 10752000 d^4 x1^8 x0^4 x2^4
+ 107008 x0^9 x2^5 d x1 + 401940480 d^3 x2^8 x0^4 x1^4 - 200970240 d^4 x
2^8 x0^4 x1^4 + 3840 x0^6 x1^4 x2^2 d^3 + 4308480 x0^5 x1^5 d^2 x2^5 - 1
436160 x0^5 x2^5 d x1^5 + 106444800 x0 x1^6 d^3 x2^6 + 364953600 x1^9 d^
3 x2^7 - 35481600 x0 x1^6 d^4 x2^6 - 106444800 x0 x1^6 d^2 x2^6 - 960 x0
^6 x1^4 x2^2 - 14400 x0^6 x1^2 x2^4 + 432000 x1^5 x2^5 x0^2 + 364953600
x2^9 d^3 x1^7 - 1728000 d x1^5 x2^5 x0^2 + 35481600 x0 x1^6 d x2^6 + 267
632640 d^3 x2^6 x0^4 x1^6 - 133816320 d^4 x2^6 x0^4 x1^6 + 3110400 d^3 x
1^10 x2^4 x0 - 1036800 d^4 x1^10 x2^4 x0 - 46656000 d^3 x2^10 x1^4 x0 +
15552000 d^4 x2^10 x1^4 x0 + 160704000 x0 x1^4 d^3 x2^8 - 53568000 x0 x1
^4 d^4 x2^8 + 10713600 x0 x2^4 d^2 x1^8 - 10713600 x0 x2^4 d^3 x1^8 - 31
10400 x1^10 d^2 x2^4 x0 + 3571200 x0 x2^4 d^4 x1^8 + 53568000 x1^4 x2^8
d x0 - 160704000 x1^4 x2^8 d^2 x0 - 3571200 x1^8 x2^4 d x0 - 4308480 d^3
x2^5 x0^5 x1^5 + 1436160 d^4 x2^5 x0^5 x1^5 - 133816320 x0^4 x2^6 d^2 x1
^6 + 46656000 x2^10 d^2 x1^4 x0 + 3840 x0^6 x1^4 x2^2 d + 57600 x0^6 x1^
2 x2^4 d - 5760 x0^6 x1^4 x2^2 d^2 + 2592000 x1^5 x0^2 x2^5 d^2 - 172800
0 d^3 x1^5 x0^2 x2^5 - 86400 x0^6 x1^2 x2^4 d^2 - 15552000 x2^10 x1^4 d
x0 - 182476800 x2^9 x1^7 d^2 - 14400 x0^2 x1^7 d^4 x2 - 3240000 x0^2 x1
d^4 x2^7 + 1036800 x1^10 x2^4 d x0 + 432000 d^4 x1^5 x0^2 x2^5 + 216000
d^4 x0^2 x1^5 x2^3 + 216000 d^4 x0^2 x2^5 x1^3 - 4 d^2 x0 x1 x2 - 6 x0^2
x1^2 d x2^2 + 3 d^2 x0^2 x1^2 x2^2
```

Obviously this polynomial is huge, having 556 terms. The degree of the polynomial is, as expected, 18 with respect to the variables (x_0, x_1, x_2).

From now on, in order to save space we will not include the trace any more; instead we will report various measures such as the number of terms or the degree.

Step (3.3)

Now we form the polynomial $\tilde{h}_A = (1-\epsilon)h_A - \epsilon\sum_{j=0}^{n} x_j^{d_A}$, then from it the polynomials $\tilde{h}_A^{(i)} = \frac{\partial \tilde{h}_A}{\partial x_i}$ for each i. Since the degree of h_A is d_A, the degrees of $\tilde{h}_A^{(i)}$ are $d_A - 1$. Thus for $n = m = d = 2$ and $A = \{3,4\}$, the degrees are 17.

For the example input, we obtain from the trace (not included here) the following information:

Poly	*Terms*	*Degree*
$\tilde{h}_{\{3,4\}}^{(0)}$	1039	17
$\tilde{h}_{\{3,4\}}^{(1)}$	957	17
$\tilde{h}_{\{3,4\}}^{(2)}$	957	17.

Step (3.4)

In this step, we need to compute the modified u–resultant of the polynomials $\tilde{h}_A^{(0)}, \ldots, \tilde{h}_A^{(n)}$. Before making any actual attempt, let us first get a rough estimate of its size.

According to the sub-algorithm, the modified u–resultant R is a constant multiple of the determinant of a certain matrix M, which is a representation of a linear transformation between the vector spaces generated by the monomials $x_0^{d_0} \cdots x_{n+1}^{d_{n+1}}$ with the degree $\hat{d} = (n+1)(\tilde{d}-1)+1$ where $\tilde{d}$ is the maximum of the degrees of the polynomials $f_0, \ldots, f_n$, with respect to $x_0, \ldots, x_n$.

Since the input polynomials f_i to the sub-algorithm are the polynomials $\tilde{h}_A^{(i)}$ of the main algorithm, we have that $\tilde{d} = d_A - 1$.

■ The size of the matrix M

Now we estimate the size D of the matrix M. For this, we simply need to count the number of the monomials with the degree $\hat{d}$. The following proposition gives us the answer:

Proposition 1 *The size of the set* $\{(d_0, \ldots, d_{n+1}) \mid d_0 + \cdots + d_{n+1} = \hat{d},\ d_i \geq 0\}$ *is* $\binom{\hat{d}+n+1}{n+1}$.

For $n = m = d = 2$ and $A = \{3,4\}$, since $d_{\{3,4\}} = 18$, we have $\tilde{d} = 18 - 1 = 17$, and so $\hat{d} = (2+1)(17-1)+1 = 49$. Thus the number of the monomials is $\binom{49+2+1}{2+1} = 22100$. The matrix M is, therefore, of the size

$$22100 \times 22100.$$

■ The degree of the polynomial $\det(M)$

Next we estimate the degree of the polynomial $\det(M)$ with respect to the variables $u_0, \ldots, u_{n+1}$. Note first that u_j's appear only in the rows corresponding to the monomials of $\mathbb{B}$ such that $d_0 < \tilde{d}, \ldots, d_n < \tilde{d}$. Clearly there are $\tilde{d}^{n+1}$ such rows. Furthermore each entry of those rows is either u_j for some j or 0. Therefore every monomial $u_0^{l_0} \cdots u_{n+1}^{l_{n+1}}$ of $\det(M)$ is of degree $\tilde{d}^{n+1}$. In other words, the polynomial $\det(M)$ is a homogeneous polynomial in $u_0, \ldots, u_{n+1}$ of degree $\tilde{d}^{n+1}$.

For $n = m = d = 2$ and $A = \{3,4\}$, $\tilde{d} = 17$. Thus $\det(M)$ is of degree

$$17^3 = 4913$$

in the variables $u_0, \ldots, u_{n+1}$.

■ The number of monomials in the polynomial $\det(M)$

It is extremely hard (impossible) to exactly determine the number of monomials in $\det(M)$. So we will be satisfied with a rough estimate, which is based on the reasonable conjecture: $\det(M)$ is dense with respect to the variables $u_0, \ldots, u_{n+1}$. The inspection of the structure of the matrix seems to support this conjecture. Under this conjecture, the number of monomials in $\det(M)$ can be estimated, by using Proposition 1, to be

$$\binom{\tilde{d}^{n+1} + n + 1}{n+1.}$$

For $n = m = d = 2$ and $A = \{3, 4\}$, it is

$$\binom{17^3+3}{3} = 19,788,792,660 \approx 10^{10}.$$

Now even though we assume that the representation of each coefficient takes only 1 byte (which is far under–estimate since they are again polynomials of the variables ϵ and δ), the representation of the polynomial $\det(M)$ would take about 10 Giga bytes. Since our computer (DEC-Station 5000) has only 32 Mega bytes main memory, it is hopeless to store the polynomial in the computer's memory, let alone computing it.

■ **The size of base (integer) coefficients of R**

As shown in Step (3) of the sub-algorithm, Renegar defines the modified u–resultant R to be a constant ($D!$) multiple of the determinant of M, in order to avoid division during determinant computation. However this causes the absolute value of the base (integer) coefficient of every monomial of R to be at least $D!$. For $n = m = d = 2$ and $A = \{3, 4\}$,

$$D! = 22100! = \Gamma(22101) \approx 10^{80000}.$$

Thus even if we would use one consecutive memory block to represent any long integer, each base coefficient would take at least

$$\log_2 10^{80000} \text{ bits} \approx 260000 \text{ bytes} \approx 33 \text{ Kbytes}.$$

Remembering that the number of monomials in $\det(M)$ can be as big as 10^{10}, it is totally impossible to store the polynomial R in any currently available computers' memory. Since it is not possible to trace the algorithm on a computer any more, the following discussion will depend only on theoretical estimation.

Step (3.5)

We will estimate the degrees of the polynomials $R_A^{<i,j,k>}$ produced in this step. Recall first that R_A is a *homogeneous polynomial* in $u_0, \ldots, u_{n+1}$. So the degree of $R_A^{<i,j,k>}$ is k less than that of R_A. Since the degree of R_A is $(d_A - 1)^{n+1}$, therefore, the the degree of $R_A^{<i,j,k>}$ is $(d_A - 1)^{n+1} - k$.

Step (4)

In this step, we gather into the set $\mathcal{R}$ all the polynomials $R_A^{<i,j,k>}$ produced in Step (3). We are interested in estimating the degrees and the numbers of the polynomials in $\mathcal{R}$. In order to simplify our analysis we will assume that for each A there is *one* polynomial with the degree $(d_A - 1)^{n+1}$. This is *far under–estimate*, since in general R_A will be a polynomial of high degree in ϵ, δ, u_0.

From now on let D_ℓ denote $(2(d+1)(\ell+1)-1)^{n+1}$. Since there are $\binom{6m+2}{\ell}$ sets A such that $|A| = \ell$, we conclude that the set $\mathcal{R}$ has at least $\binom{6m+2}{\ell}$ polynomials of degree D_ℓ for each $\ell = 0, \ldots, n+1$.

Thus for $n = m = d = 2$ we obtain the following estimates:

Degree	*Number*
125	$\gg 1$
1331	$\gg 14$
4913	$\gg 91$
12167	$\gg 364$

Step (5)

We will estimate the degrees and the numbers of the polynomial list in the set $\mathcal{Q}$. An easy check shows that the degrees of the polynomials $\nabla_{u_1,\ldots,u_{n+1}} R(\beta + te_{n+1})$ are (very tightly) bounded above by the degree of R with respect to the variable u_{n+1}, which in turn is (very tightly) bounded above by the degree D_R of R. Thus for each $R \in \mathcal{R}$ the set $\mathcal{Q}$ has nD_R^2 polynomial lists of degree $D_R - j$ for each $j = 0, \ldots, D_R$.

Now according to the analysis of Step (4), the set $\mathcal{R}$ has at least $\binom{6m+2}{\ell}$ polynomials R of degree D_ℓ for each $\ell = 0, \ldots, n+1$.

Thus we see that the set $\mathcal{Q}$ has at least $\binom{6m+2}{\ell} nD_\ell^2$ polynomial lists of degree $D_\ell - j$ for each $\ell = 0, \ldots, n+1$ and $j = 0, \ldots, D_\ell$. But in order to simplify our analysis we ignore all the polynomial lists in $\mathcal{Q}$ with $\ell < n+1$. (Here we are again making a *far under–estimate* on the size of the set $\mathcal{Q}$.) In conclusion, the set $\mathcal{Q}$ has at least $\binom{6m+2}{n+1} nD_{n+1}^2$ polynomials lists of degree μ for each $\mu = 0, 1, \ldots, D_{n+1}$.

For $n = m = d = 2$, since $D_{2+1} = 12167$, the set $\mathcal{Q}$ has at least

$$\binom{14}{3} \cdot 2 \cdot 12167^2 \approx 10^{11}$$

polynomials lists of degree μ, for each $\mu = 1, 2, \ldots, 12167$. Thus we have

$$|\mathcal{Q}| \gg 10^{15}.$$

Step (6)

We will estimate the degrees and the numbers of the univariate polynomials lists in the set $\mathcal{F}$. Since the degree of G_i is d, the degree of $G_i(q)$ is $d \cdot \deg(q)$. Thus referring the analysis of Step (5), we obtain the following estimate: the set $\mathcal{F}$ has at least $2\binom{6m+2}{n+1} nD_{n+1}^2$ univariate polynomial lists of degree 2μ for each $\mu = 0, 1, \ldots, D_{n+1}$.

Thus for $n = m = d = 2$, the set $\mathcal{F}$ contains at least

$$2 \cdot 10^{11}$$

univariate polynomial lists of degree μ, for each $\mu = 2, 4, \ldots, 24334$. Again we estimate the size of the set $\mathcal{F}$ as

$$|\mathcal{F}| \gg 2 \cdot 10^{15}.$$

Step (7)

Now we need to compute the consistent sign vectors of each univariate polynomial list in the set $\mathcal{F}$. Renegar proposes to use the algorithm of Ben-Or, Kozen, and Reif [4]. So we need to make at least $2\binom{6m+2}{n+1} n D_{n+1}^2$ calls to the BKR algorithm with univariate polynomials lists of degree μ for each $\mu = 0, \ldots, D_{n+1}$.

For $n = m = d = 2$, this means that we need to make at least

$$2 \cdot 10^{11}$$

calls to the BKR algorithm with polynomials lists of degree μ, for each $\mu = 2, 4, \ldots, 24334$.

The author conjectures that even just one call of BKR on a polynomial list of degree 24334 will take longer than a day using any modern computer.[7] Even though we assume that each call takes in average only 0.1 second, since we need to do this at least $2 \cdot 10^{15}$ times, Step (7) alone will take at least

$$2 \cdot 10^{14} \text{ seconds } \approx \text{ 6 million years.}$$

Step (8)

This final step is trivial. For the example input, this step should not take more than a second to decide it to be false.

9 Conclusion and Discussion

In this section we make certain conclusions based on the results obtained from the previous three sections. Further we briefly discuss their implication on the complexities of the three algorithms for arbitrary inputs (possibly with quantifier alternation). We also give a brief comparison of Collins' algorithm and Tarski's algorithm.

■ Conclusion of the last three sections

We set out to answer the following questions posed in the introduction of this paper.

[7] It is shown in [4] that the computing time of one application of BKR algorithm is dominated by $(md)^{O(1)}$ where m is the number of polynomials in a univariate polynomial list, and d is its degree bound.

(Q1) Which algorithm among the three is the fastest for small inputs? Let X be the fastest one.

(Q2) If X is not Renegar's, then at what input size does Renegar's algorithm become faster than X?

(Q3) At that trade–off point, how much computing time is required by either of the algorithms?

In an attempt to answer these questions, in the last three sections we estimated the running time of the three algorithms on the inputs with $n = m = d = L = 2$ (on DEC–station 5000, running UNIX operating system, with 32 Mega byte main memory). Here is their summary:

Algorithm	$n = m = d = L = 2$
Collins	$\ll$ 1 second
Grigor′ev/Vorobjov	$\gg$ 1 million years
Renegar	$\gg$ 1 million years

Choice of different hardware and programming environments make some difference on the speed of algorithms, but not by the factor of millions. Therefore we can safely answer to the question Q1 as follows:

(A1) Collins' algorithm is the fastest for small inputs with $n = m = d = L = 2$.

As for the question Q2, our analysis in the previous sections is not sufficient enough to give a meaningful answer.

In order to answer the question Q3, let us make a safe assumption that the three algorithms will take longer on bigger inputs. From this assumption and their estimated running times for the case $n = m = d = L = 2$, we logically deduce the following answer to Q3:

(A3) The computing time at the trade–off point is far greater than a million years on currently available machines.

In conclusion, Collins' algorithm is expected to be the fastest among the three algorithms on inputs which can be decided in a reasonable amount of time.

■ Comparison on inputs with quantifier alternation

Until now, we compared the three algorithms only on existential sentences (or equivalently sentences without quantifier alternation). Below we briefly compare them or their extensions on deciding arbitrary sentences possibly with quantifier alternations. (A paper with a detailed discussion is in preparation [22].)

Collins' algorithm, without any modification, can decide arbitrary sentences. In [14] Grigor'ev, by generalizing the algorithm reviewed in Section 3, gave an algorithm for deciding arbitrary sentences. In [30] Renegar did the same by generalizing the algorithm reviewed in Section 4.

Let ω be the number of quantifier blocks in the input sentence. Let n_i be the number of quantifiers in the i–th quantifier block so that $n_1 + \cdots + n_\omega = n$. The table below gives the (already known) theoretical complexities of the three general algorithms:

Algorithm	Theoretical
Collins	$L^3(md)^{2^{O(n)}}$
Grigor′ev	$L(md)^{O(n)^{4\omega-2}}$
Renegar	$L(\log L)(\log\log L)(md)^{2^{O(\omega)}\prod_i n_i}$

The table shows that, for sufficiently large n and small ω, Renegar's algorithm is the fastest, Grigor'ev's is the next fastest, and Collins' is the slowest. It also shows that, for sufficiently large n and large ω, the algorithms of Renegar and Collins are similar in speed, while Grigor'ev's is slower than either. In order to find out the complexities of these algorithms on small inputs, let us again consider the inputs with $n = m = d = L = 2$ but this time with the quantifiers such as $(\exists x_1)(\forall x_2)$ or $(\forall x_1)(\exists x_2)$.

The dependence of the running time of Collins' algorithm on the quantifier being used is negligible, and we can safely conclude that the input sentences can be decided within a second on a DEC–station.

Grigor'ev's algorithm, besides other expensive computations, makes at least one call of the algorithm of Section 3 with the same input polynomials (see page 70 of [14]), so it is clear that the algorithm will take more computing time than the algorithm of Section 3. Thus we safely conclude that Grigor'ev's algorithm will take at least one million years to decide the input sentences.

Renegar's algorithm begins by projecting the two bivariate polynomials into a set of univariate polynomials. An analysis of the projection sub–algorithm shows that the number of the univariate polynomials is expected to be at least one million[8] (see [22] for details). But then the algorithm should continue to compute sample points of the connected sign partition of $\mathbb{R}$ for the univariate polynomials, then lift them to obtain sample points in $\mathbb{R}^2$, and then determine signs of the input polynomials on them. Thus it is expected that the whole computation would take enormous amount of computing time.

Thus it suggests that Collins' algorithm is faster than either of the algorithms of Grigor'ev [14] and Renegar [30] on any inputs (possibly with quantifier alternations) which can be decided in a reasonable amount of time.

■ Comparison of the algorithms of Tarski and Collins

Since Collins' algorithm is faster than the algorithms with better theoretical complexities for small inputs, a question naturally arises: *Is Tarski's algorithm (with non-elementary time complexity[9]) in turn faster than Collins' for small*

[8] In contrast, Collins' projection produces at most five polynomials as shown in Section 6.

[9] In particular, the running time of Tarski's algorithm cannot be bounded by any finite tower of exponential functions in the number of variables.

inputs? In fact this question can be broadened by recalling that both algorithms solve quantifier elimination problems, which are more general than decision problems. A paper with the detailed discussion on this question is in preparation [23], but here we give a brief summary of what we have learned so far.

- For decision problems, Collins' algorithm is in general much faster than Tarski's, even for small inputs. For the example input in Section 5, Tarski's algorithm, after eliminating one quantifier ($\exists x_2$), produces a formula in x_1 having at least 20,000 atomic formulas, where several polynomials are of degrees about 500 with coefficients of about 80 decimal digits. But then, in order to decide the sentence, the algorithm should continue to eliminate the remaining quantifier ($\exists x_1$) from this huge formula. An inspection of the algorithm (especially of the T operator) suggests that it will take an enormous amount of computing time, producing a huge number of large intermediate polynomials.
- For quantifier elimination problems, Collins' algorithm is also in general much faster than Tarski's, even for small inputs. But for certain quantifier elimination problems, especially those with only one quantifier to eliminate, Tarski's algorithm can be improved so that it is often faster than Collins' [24].

■ Suggestions

The investigations in this paper suggest that theoretical analyses based on the big O notation are too coarse for comparing decision algorithms over the reals. It seems that one should devise finer methods for analyzing the algorithms, in order to obtain meaningful insight into their relative complexities. It also shows the need for more research on devising and improving practically efficient decision algorithms for small inputs.

The author would like to thank Bruno Buchberger who has made various helpful suggestions on the draft of this paper.

References

[1] D. S. Arnon. Algorithms for the geometry of semi-algebraic sets. Technical Report 436, Computer Sciences Dept, Univ. of Wisconsin-Madison, 1981. Ph.D. Thesis.

[2] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM J. Comp.*, 13:865–877, 1984.

[3] D. S. Arnon and M. Mignotte. On mechanical quantifier elimination for elementary algebra and geometry. *Journal of Symbolic Computation*, 5(1,2):237–260, 1988.

[4] M. Ben-Or, D. Kozen, and J. H. Reif. The complexity of elementary algebra and geometry. *J. Comput. System Sci.*, 32(2):251–264, 1986.

[5] W. Böge. Decision procedures and quantifier elimination for elementary real algebra and parametric polynomial nonlinear optimization. Manuscript in preparation, 1980.

[6] B. Buchberger and H. Hong. Speeding-up quantifier elimination by Groebner bases. Technical Report 91-06.0, Research Institute for Symbolic Computation, Johannes Kepler University A-4040 Linz, Austria, 1991.

[7] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the 20th annual ACM symposium on the theory of computing*, pages 460–467, 1988.

[8] B. W. Char, K. O. Geddes, G. H. Gonnet, and S. M. Watt. *Maple User's Guide*. WATCOM Publications Limited, 4th edition, 1985.

[9] P. J. Cohen. Decision procedures for real and p-adic fields. *Comm. Pure and Applied Math.*, 22:131–151, 1969.

[10] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Lecture Notes In Computer Science*, pages 134–183. Springer-Verlag, Berlin, 1975. Vol. 33.

[11] G. E. Collins and H. Hong. Partial CAD construction in quantifier elimination. Technical Report OSU-CISRC-10/89 TR45, Computer Science Dept, The Ohio State University, 1989. To appear in *Journal of Symbolic Computation*.

[12] G. E. Collins and R. Loos. *The SAC-2 Computer Algebra System*. Department of Computer and Information Sciences, Ohio State University.

[13] N. Fitchas, A. Galligo, and J. Morgenstern. Algorithmes repides en séquential et en parallele pour l'élimination de quantificateurs en géométrie élémentaire. Technical report, UER de Mathématiques Universite de Paris VII, 1987. To appear in: Séminaire Structures Algébriques Ordonnées.

[14] D. Yu. Grigor'ev. The complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1,2):65–108, 1988.

[15] D. Yu. Grigor'ev and N. N. Vorobjov (Jr). Solving systems of polynomial inequalities in subexponential time. *Journal of Symbolic Computation*, 5(1,2):37–64, 1988.

[16] J. Heintz, M-F. Roy, and P. Solernó. On the complexity of semialgebraic sets. In *Proc. IFIP*, pages 293–298, 1989.

[17] C. Holthusen. *Vereinfachungen für Tarski's Entscheidungsverfahren der elementaren reellen Algebra*. PhD thesis, University of Heidelberg, January 1974.

[18] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. Technical Report OSU-CISRC-12/89 TR55, Computer Science Dept, The Ohio State University, 1989.

[19] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *International Symposium of Symbolic and Algebraic Computation*, 1990.

[20] H. Hong. *Improvements in CAD–based Quantifier Elimination.* PhD thesis, The Ohio State University, 1990.

[21] H. Hong. Collision problems by partial CAD construction. Technical report, Research Institute for Symbolic Computation, Johannes Kepler University A-4040 Linz, Austria, January 1991.

[22] H. Hong. Comparison of several decision algorithms for the first order theory of the reals. Technical report, Research Institute for Symbolic Computation, Johannes Kepler University A-4040 Linz, Austria, 1991.

[23] H. Hong. Comparison of the algorithms of Tarski and Collins for quantifier elimination in the first order theory of the reals. Technical report, Research Institute for Symbolic Computation, Johannes Kepler University A-4040 Linz, Austria, 1991.

[24] H. Hong and A. Manache. Improvements of Tarski's quantifier elimination algorithm for the first order theory of the reals. Technical report, Research Institute for Symbolic Computation, Johannes Kepler University A-4040 Linz, Austria, 1991.

[25] J. R. Johnson. *Algorithms for Polynomial Real Root Isolation.* PhD thesis, The Ohio State University, 1991. Dissertation in preparation.

[26] L. Langemyr. The cylindrical algebraic decomposition algorithm and multiple algebraic extensions. In *Proc. 9th IMA Conference on the Mathematics of Surfaces*, September 1990.

[27] D. Lazard. An improved projection for cylindrical algebraic decomposition. Unpublished manuscript, 1990.

[28] S. McCallum. *An Improved Projection Operator for Cylindrical Algebraic Decomposition.* PhD thesis, University of Wisconsin-Madison, 1984.

[29] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals (part I). Technical Report 853, Cornell University, Ithaca, New York 14853-7501 USA, July 1989.

[30] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals (part II). Technical Report 854, Cornell University, Ithaca, New York 14853-7501 USA, July 1989.

[31] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals (part III). Technical Report 856, Cornell University, Ithaca, New York 14853-7501 USA, August 1989.

[32] A. Seidenberg. A new decision method for elementary algebra. *Ann. of Math*, 60:365–374, 1954.

[33] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* Univ. of California Press, Berkeley, second edition, 1951.