Available online at www.sciencedirect.com
ScienceDirect
Journal of Symbolic Computation 43 (2008) 811–844
Journal of Symbolic Computation
www.elsevier.com/locate/jsc

Rational Univariate Reduction via toric resultants

Koji Ouchi[1], John Keyser

Department of Computer Science, 3112 Texas A&M University, College Station, TX 77843-3112, USA

Received 19 September 2006; accepted 8 February 2008
Available online 10 March 2008

Abstract

We describe algorithms for solving a given system of multivariate polynomial equations via the Rational Univariate Reduction (RUR). We compute the RUR from the toric resultant of the input system. Our algorithms derandomize several of the choices made in similar prior algorithms. We also propose a new derandomized algorithm for solving an overdetermined system. Finally, we analyze the computational complexity of the algorithm, and discuss its implementation and performance.

Keywords: Polynomial system solving; Rational univariate representation; Geometric resolution; Toric (or sparse) resultant

1. Introduction

Solving systems of multivariate polynomial equations is a common problem across many applications. In this paper, we describe a method for solving such a system based on the Rational Univariate Reduction (RUR) of the system. Given a system of polynomials in n variables, the RUR reduces the system into $n + 1$ univariate polynomials h and $h_1, \ldots, h_n$ so that the value of the ith coordinate of a common root of the input system is the value of the univariate polynomial h_i evaluated at some root of the univariate polynomial h. This representation of the common roots of the input system is called the Rational Univariate Representation or the geometric resolution of the system.

The RUR of a square system of multivariate polynomials a system where the number of polynomials and the number of the variables are the same is obtained by looking at the quotient

E-mail addresses: kouchi@cs.tamu.edu, kojio@wolfram.com (K. Ouchi), keyser@cs.tamu.edu (J. Keyser).
[1] Currently at Wolfram Research. Tel.: +1 979 777 5326; fax: +1 217 398 0747.

0747-7171/$ - see front matter
doi:10.1016/j.jsc.2008.02.002

ring modulo the ideal generated by polynomials belonging to the system. The structure of this quotient ring is usually understood by computing the Gröbner basis for this ideal. We instead derive the RUR from the toric u-resultant of the system. In terms of computational complexity, the method is more efficient than those methods using the Gröbner basis.

Our method succeeds even if the zero set of the input system is not zero-dimensional. We compute the RUR for some finite set that contains all the isolated common roots of the input system as well as at least one point from every irreducible component of the zero set of the system. Taking advantage of this feature, we develop a new algorithm for computing the RUR of an overdetermined system of multivariate polynomial equations.

One improvement on prior method is that it is "derandomized"; we have eliminated random choices made in prior approaches. For us, a random choice is made only when we know beforehand that the probability of failure is 0, when we have a method to determine when such a failure occurs, and when a (less efficient) deterministic alternative is also known.

The algorithms we describe here can be used to compute the exact RUR, meaning that all the coefficients of univariate polynomials forming the RUR are computed to full precision. In particular, when the coefficients of the input system belong to the field of rational numbers or some finite field, our algorithms can be implemented on a Turing machine (i.e. existing computers) to compute the exact RUR.

1.1. Main results

In contrast to the earlier work on computing the RUR using the toric u-resultant, the main new results we present here are:

- We present a derandomized version of the algorithm for solving square systems (Section 3.1).
- We present a new derandomized algorithm for solving overdetermined systems (Section 3.2.2).
- We present a new algorithm for finding real roots of zero-dimensional square systems (Section 3.3).
- We analyze the arithmetic complexity of our algorithm and give a tight bound (Section 5).
- We describe an implementation of the algorithm and show some experimental results (Section 6).

1.2. Organization

This paper is organized as follows:

- In Section 2, we describe the theoretical background and the algorithms presented in this paper. We list related work (Section 2.1) and preliminary definitions and results (Section 2.2).
- In Section 3, we state our algorithms precisely and prove their correctness. We first describe the algorithm for computing the toric roots of a square system (Section 3.1). Then, we describe the RUR for the affine zero set of a non-square system (Section 3.2). Finally, we discuss the algorithm for computing the real roots of a system of polynomials with rational coefficients (Section 3.3).
- In Section 4, we show some examples.
- In Section 5, we give the complexity analysis of our main algorithm.
- In Section 6, we describe an implementation of our algorithm and show some experimental results.
- In Section 7, we conclude and list some directions for future work.

2. Background

2.1. Previous work

The RUR for a system of polynomials has been known for a long time. The RUR was first seen in Kronecker (1895–1931), and variations on it have been known for a while (Canny, 1988). The RUR has become prominent in computer algebra mainly in the recent years.

If an input system is of dimension zero then the RUR can be computed via the "multiplication table method" (Rouillier, 1999; Gonzalez-Vega et al., 1999; Basu et al., 2003). An extension of this method finds all the isolated real roots as well as at least one point from every *real* positive-dimensional component (Aubry et al., 2002; Din and Shost, 2004). A standard implementation of the method requires reduction of the input polynomials into some normal form via the Gröbner basis. The Gröbner basis is discontinuous with respect to changes in the coefficients of the input polynomials (Mourrain, 1999).

A Gröbner-free algorithm to compute the RUR for a zero-dimensional system has been proposed (Giusti et al., 2001). Recent work even handles systems with multiple roots (Lecerf, 2002). The complexity analysis of this algorithm is considered in Jeronimo et al. (2004).

The *toric resultant* (or the *sparse resultant*) of a system of $n + 1$ polynomials with indeterminate coefficients in n variables is a polynomial with integer coefficients in these indeterminates (as variables) that vanishes iff the system has a common root on some toric variety over an algebraic closure of the field to which the coefficients of the polynomials belong (Gelfand et al., 1994; Cox et al., 1998; Sturmfels, 2002; Cox, 2003). The toric resultant is expressed as a divisor of the determinant of some square matrix, called the *toric resultant matrix* or the *Newton matrix* (Sturmfels, 1994; Emiris, 1996; Rojas, 2003). The *mixed-subdivision-based algorithm* (Canny and Emiris, 1993, 2000; Emiris, 2003) is historically the first practical algorithm that constructs the resultant matrix, but the size of the matrix constructed is often too large. Another version of this algorithm first constructs a small matrix and incrementally constructs larger matrices until one that works is found (Emiris and Canny, 1995). Several efforts have been made to construct smaller resultant matrices in order to speedup the toric resultant computation (D'Andrea, 2002; Emiris, 2002; Hong and Minimair, 2002; Minimair, 2002; Dickenstein and Emiris, 2003; Khetan, 2003, 2005).

The resultant-based method for solving a system of polynomial equations fails if the zero set of the input system has some positive-dimensional components. In order to solve such systems, a perturbation technique is used. The Generalized Characteristic Polynomial (GCP) by Canny (1990) can be used to express solutions to dense homogeneous square systems with degeneracies. The toric perturbation is defined as a particular coefficient of the toric GCP (Rojas, 1997, 1999a, 2000). The toric perturbation works even if an input square system has some multiple roots at the point at infinity. A potentially more efficient perturbation technique that finds expectedly fewer monomials has been proposed by D'Andrea and Emiris (2001; 2003).

The toric resultant-based method can be modified so that it finds some set containing all the affine roots of a square system (Rojas and Wang, 1996; Li and Wang, 1996; Rojas, 1999b,a, 2000).

The algorithm for computing the RUR of a given system of polynomials with rational coefficients described in this paper improves the versions in Rojas (1999a) and Keyser et al. (2005).

2.2. Definitions and prior results

Let $\mathbb{K}$ be a field. Write $\overline{\mathbb{K}}$ for the algebraic closure of $\mathbb{K}$ and $\mathbb{K}^*$ for $\mathbb{K} \setminus \{0\}$.

2.2.1. Rational univariate reduction

Consider a *square* system of n polynomials $f_1, \ldots, f_n$ in n variables with coefficients in $\mathbb{K}$. It is known (Rojas, 1999a; Rouillier, 1999; Basu et al., 2003) that there exists a finite set Z' containing all the isolated common roots of the input system (in $\overline{\mathbb{K}}^n$) such that each coordinate of points in Z' is represented as some univariate polynomial h_i with coefficients in $\mathbb{K}$ evaluated at a root (in $\overline{\mathbb{K}}$) of some other univariate polynomial h with coefficients in $\mathbb{K}$. That is

$$Z' = \left\{ (h_1(\theta), \ldots, h_n(\theta)) \in \overline{\mathbb{K}}^n \mid \theta \in \overline{\mathbb{K}} \text{ with } h(\theta) = 0 \right\}. \tag{1}$$

This reduction is called a *Rational Univariate Reduction (RUR)* and this representation of the zero set of the system is called a *Rational Univariate Representation (RUR)*.

In the rest of paper, we write M' for the cardinality of the finite set of Z'. Generally, the quantity M' is the number of the roots of the input system. More precisely, if the input system is zero-dimensional and all the roots are toric, i.e. there are finite roots in $(\overline{\mathbb{K}}^*)^n$, then M' matches the number of distinct roots of the input system.[2]

We derive the RUR from the toric perturbation (Rojas, 1999a), which is a generalization of the "toric" u-resultant. In this section, we will describe preliminary facts about toric resultants and toric perturbations.

2.2.2. Toric resultants

Let f be a polynomial in n variables $X_1, \ldots, X_n$ with coefficients in $\mathbb{K}$. Define the *support* of f to be the finite set A of exponents of all the monomials appearing in f with non-zero coefficients. Thus, A is some non-empty finite set of integer points in $\mathbb{R}^n$, and

$$f = \sum_{a \in A} c_a X^a, \quad c_a \in \mathbb{K}^*$$

where $X^a = X_1^{a_1} \cdots X_n^{a_n}$ for $a = (a_1, \ldots, a_n)$.

Fix $n+1$ non-empty sets $A_0, A_1, \ldots, A_n$ of integer points in $\mathbb{R}^n$. A system of $n+1$ polynomials $f_0, f_1, \ldots, f_n$ in n variables $X_1, \ldots, X_n$ with supports $A_0, A_1, \ldots, A_n$ is specified via coefficient vectors $\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ where

$$\boldsymbol{c}_i = \left(c_{ia} \in \mathbb{K}^* \mid a \in A_i \right) \text{ such that } f_i = \sum_{a \in A_i} c_{ia} X^a.$$

For $i = 0, 1, \ldots, n$, write MV_{-i} for the mixed-volume of the convex hulls of $A_0, A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n$ (Sturmfels, 1994; Cox et al., 1998; Sturmfels, 2002). Recall that all these mixed-volumes are non-negative, i.e., $\mathrm{MV}_{-i} \geq 0$ for $i = 0, 1, \ldots, n$ (Cox et al., 1998). Assume that at least one of these mixed-volumes $\mathrm{MV}_{-0}, \mathrm{MV}_{-1}, \ldots, \mathrm{MV}_{-n}$ is strictly positive, i.e., $\sum_{i=0}^{n} \mathrm{MV}_{-i} > 0$. Then, there exists a unique (up to sign) irreducible polynomial

$$\mathrm{TRes}_{A_0, A_1, \ldots, A_n}(\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_n) \in \mathbb{Z}[\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_n],$$

called the *toric resultant* or the *sparse resultant* of the system which has the following property:

$$\text{the system } (f_0, f_1, \ldots, f_n) \text{ has a common root in } (\overline{\mathbb{K}}^*)^n$$
$$\Longrightarrow \mathrm{TRes}_{A_0, A_1, \ldots, A_n}(\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_n) = 0.$$

The toric resultant is also written as $\mathrm{TRes}(f_0, f_1, \ldots, f_n)$.

[2] Vanishing of the toric resultant of a system of polynomials with coefficients in $\mathbb{K}$ actually tells us whether or not the system has common roots in the toric variety rather than $(\overline{\mathbb{K}}^*)^n$. The toric variety has a naturally embedded copy of $(\overline{\mathbb{K}}^*)^n$ and, for simplicity, we use $(\overline{\mathbb{K}}^*)^n$ instead of the toric variety.

Several algorithms for computing the toric resultant of a given system of $n+1$ polynomials in n variables with supports $A_0, A_1, \ldots, A_n$ have been proposed (Emiris and Canny, 1995; Canny and Emiris, 2000). These algorithms construct a square matrix N, called the *toric resultant matrix* or the *Newton matrix*, whose determinant is some non-trivial multiple of the toric resultant. The non-zero entries of every row of N are the coefficients $\boldsymbol{c}_i$ of some input polynomial f_i. It follows that $\det N$ is a homogeneous polynomial in each coefficient vector $\boldsymbol{c}_i$, and thus, the total degree of $\det N$ with respect to each coefficient vector $\boldsymbol{c}_i$, $\deg_{\boldsymbol{c}_i}(\det N)$, is well-defined. These quantities $\deg_{\boldsymbol{c}_i}(\det N)$ are bounded in terms of the mixed-volumes. More precisely, it is known that

$$\deg_{\boldsymbol{c}_0}(\det N) = \mathrm{MV}_{-0}, \tag{2}$$

$$\deg_{\boldsymbol{c}_i}(\det N) \geq \mathrm{MV}_{-i}, \quad i = 1, \ldots, n. \tag{3}$$

Note that equality (2) always holds, while, in (3), the equalities hold only when $\det N$ is the toric resultant without any extraneous factor (Pedersen and Sturmfels, 1993).

2.2.3. Toric perturbations

Consider a *square* system of n polynomials $f_1, \ldots, f_n \in \mathbb{K}[X_1, \ldots, X_n]$ with supports $A_1, \ldots, A_n$. Assume that the mixed-volume MV_{-0} of the convex hulls of $A_1, \ldots, A_n$ is strictly positive, i.e., $\mathrm{MV}_{-0} > 0$.

Let $A_0 = \{\boldsymbol{o}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ where $\boldsymbol{o}$ is the origin and $\boldsymbol{b}_i$ is the ith standard basis vector in $\mathbb{R}^n$. Also, let $f_0 = u_0 + u_1 X_1 + \cdots + u_n X_n$ where $\boldsymbol{u} = (u_0, u_1, \ldots, u_n)$ is a vector of parameters. Choose n polynomials $f_1^*, \ldots, f_n^* \in \mathbb{K}[X_1, \ldots, X_n]$ with supports contained in $A_1, \ldots, A_n$, that have only finitely many common roots in $(\overline{\mathbb{K}}^*)^n$. Define the *toric Generalized Characteristic Polynomial* $\mathrm{TGCP}(s, \boldsymbol{u})$ for the system $(f_1, \ldots, f_n)$ to be the toric resultant of the perturbed system $\left(f_0, f_1 - sf_1^*, \ldots, f_n - sf_n^*\right)$:

$$\mathrm{TGCP}(s, \boldsymbol{u}) = \mathrm{TRes}\left(f_0, f_1 - sf_1^*, \ldots, f_n - sf_n^*\right) \in \mathbb{K}[s][\boldsymbol{u}].$$

Also, define a *toric perturbation* $\mathrm{TPert}(\boldsymbol{u})$ for the system $(f_1, \ldots, f_n)$ to be the non-zero coefficient of the lowest degree term in $\mathrm{TGCP}(s, \boldsymbol{u})$ regarded as a polynomial in the variable s.

Theorem 1 (*Rojas, 1999a[Main Theorem 2.4]*). $\mathrm{TPert}(\boldsymbol{u})$ *is well-defined, i.e., for a given square system of polynomials $f_1, \ldots, f_n$ in n variables with coefficients in $\mathbb{K}$, making a suitable choice of polynomials $f_1^*, \ldots, f_n^*$,* $\mathrm{TGCP}(s, \boldsymbol{u})$ *(regarded as a polynomial in s) always has a non-zero coefficient.* $\mathrm{TPert}(\boldsymbol{u})$ *is a homogeneous polynomial in parameters $u_0, u_1, \ldots, u_n$ with coefficients in $\mathbb{K}$ which has the following properties:*

(1) *If $(\zeta_1, \ldots, \zeta_n) \in (\overline{\mathbb{K}}^*)^n$ is an isolated common root of the input system $(f_1, \ldots, f_n)$ then $u_0 + u_1\zeta_1 + \cdots + u_n\zeta_n$ is a linear factor of* $\mathrm{TPert}(\boldsymbol{u})$.
(2) $\mathrm{TPert}(\boldsymbol{u})$ *completely splits into linear factors over $\overline{\mathbb{K}}$. Letting Z be the zero set of the system (which might be infinite), for every irreducible component W of $Z \cap (\overline{\mathbb{K}}^*)^n$, there is at least one factor of* $\mathrm{TPert}(\boldsymbol{u})$ *corresponding to a point $(\zeta_1, \ldots, \zeta_n) \in W$.*

Immediately from (2) and (3) together with the definitions above

Corollary 2.

$$\deg_{u_0} \mathrm{TPert}(\boldsymbol{u}) = \mathrm{MV}_{-0}, \tag{4}$$

$$\deg_s \mathrm{TGCP}(s, \boldsymbol{u}) = \sum_{i=1}^{n} \mathrm{MV}_{-i} \leq \dim N - \mathrm{MV}_{-0}. \tag{5}$$

Remark 3. The assumption that $\mathrm{MV}_{-0} > 0$ can be removed by paying only a reasonable amount of extra computation. In the event that $\mathrm{MV}_{-0} = 0$, we can add $\boldsymbol{O}(n)$ points to the supports $A_1, \ldots, A_n$ so that MV_{-0} is strictly positive. Those points can be chosen deterministically (Rojas, 1999b) or at random. Because of efficiency, we will use a randomized method, and will not focus on these details in this paper.

Remark 4. If $u_0 + u_1\zeta_1 + \cdots + u_n\zeta_n$ is a linear factor of TPert $(\boldsymbol{u})$ then $(\zeta_1, \ldots, \zeta_n)$ is a common root of the input system in the toric variety associated with the Minkowski sum of the convex hulls of the supports of the input system. Note that there might exist a root of the input system in the toric variety but not in $(\overline{\mathbb{K}}^*)^n$.

Remark 5. There is a deterministic method to choose polynomials $f_1^*, \ldots, f_n^*$ in Theorem 1 (Rojas, 1999b). However, because of efficiency, we will use a randomized method.

3. Algorithm

In this section, we will describe an algorithm for computing the RUR. After discussing the derandomized process for computing the RUR for the toric zero set (the zero set in $(\overline{\mathbb{K}}^*)^n$) of a square system (Section 3.1), we will discuss extending this to the RUR for the affine zero set (the zero set in $\overline{\mathbb{K}}^n$) of a non-square system (Section 3.2). We will also discuss handling the cases when $\mathbb{K} = \mathbb{Q}$ or $\mathbb{K} = \mathbb{R}$ and we are interested in finding the real roots only (Section 3.3).

In the rest of this paper, we will assume that the characteristic of the field $\mathbb{K}$ is 0 or sufficiently large. The actual value for the characteristic of $\mathbb{K}$ (when it is not zero) will be given later in Section 3.1.

Algorithms described in this section are as follows (Fig. 1):

Algorithm **RUR_toric_square** computes the RUR for the toric zero set (the zero set in $(\overline{\mathbb{K}}^*)^n$) of a square system of polynomials $f_1, \ldots, f_n$ in n variables with rational coefficients.

Algorithm **RUR_square** computes the RUR for the affine zero set (the zero set in $\overline{\mathbb{K}}^n$) of a square system of polynomials $f_1, \ldots, f_n$ in n variables with rational coefficients. It internally calls algorithm **RUR_toric_square**.

Algorithm **RUR_overconstrained** computes the RUR for the affine zero set of a system of polynomials $f_1, \ldots, f_m$ in n variables with rational coefficients when $m > n$. It internally calls Algorithm **RUR_square**.

Algorithm **RUR** computes the RUR for a system of polynomials $f_1, \ldots, f_m$ in n variables with rational coefficients. It internally calls Algorithm **RUR_overconstrained** or Algorithm **RUR_square**.

3.1. Toric RUR for square systems

Consider a *square* system of n polynomials $f_1, \ldots, f_n$ in n variables with coefficients in $\mathbb{K}$. Let Z be the zero set of the system. Assume that the mixed-volume MV_{-0} of the convex hulls of supports $A_1, \ldots, A_n$ of $f_1, \ldots, f_n$ is strictly positive. It follows from Theorem 1 that, with a suitable choice of polynomials $f_1^*, \ldots, f_n^*$, there exists a finite subset Z' of $Z \cap (\overline{\mathbb{K}}^*)^n$ such that

- Z' contains all the isolated common roots of the input system in $(\overline{\mathbb{K}}^*)^n$ as well as at least one point from every irreducible component of $Z \cap (\overline{\mathbb{K}}^*)^n$, and
- the univariate polynomial $h(T)$ in the RUR for Z' is derived from TPert $(\boldsymbol{u})$ = TPert $(u_0, u_1, \ldots, u_n)$ by setting u_0 to a variable T and specializing parameters $u_1, \ldots, u_n$ to some appropriate values in $\overline{\mathbb{K}}$.

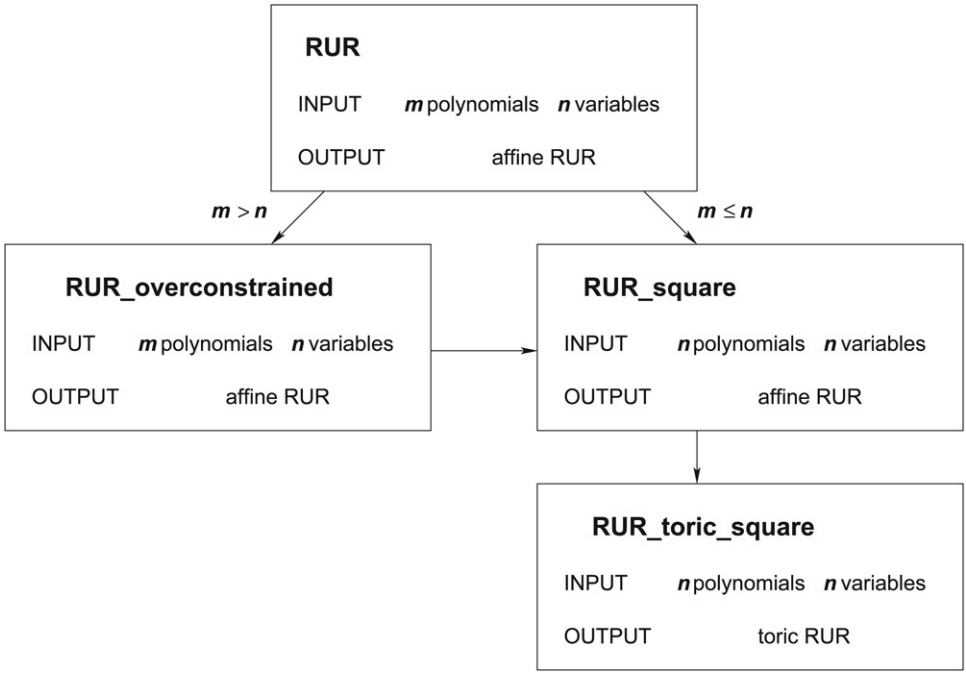

Fig. 1. Algorithms described in Section 3.

The other univariate polynomials $h_1, \ldots, h_n$ in the RUR are also derived from toric perturbations. Recall that M' is defined to be the cardinality of Z' (Section 2.2.1). Note that if the input system has only finitely many roots then $Z' = Z \cap (\overline{\mathbb{K}}^*)^n$, and thus, the input system has M' distinct common roots (in $(\overline{\mathbb{K}}^*)^n$). Auxiliary polynomials $f_1^*, \ldots, f_n^*$ can be chosen deterministically (Rojas, 1999a), though the process is costly. In practice, polynomials with random coefficients are used. The probability that random polynomials work suitably is 1 over the real numbers and unsuitable choices are detectable. (See steps **11**, **12** in Section 3.1.1.) Thus, we will describe a version of the algorithm in which choices of auxiliary polynomials remain randomized. The conditions for an appropriate specialization of parameters $u_1, \ldots, u_n$ will be clarified later. We will see that parameters $u_1, \ldots, u_n$ can be appropriately specialized.

We give an algorithm for computing the RUR for Z' (Section 3.1.1). Step-by-step details are given immediately afterward (Section 3.1.2).

The algorithm computes the RUR only when the mixed-volume MV_{-0} for the convex hulls of supports $A_1, \ldots, A_n$ of polynomials $f_1, \ldots, f_n$ is strictly positive. If MV_{-0} turns out to be 0 then the algorithm adds some points to $A_1, \ldots, A_n$ so that MV_{-0} becomes strictly positive. See Remark 3 and steps from **3** through **6** (Section 3.1.1).

3.1.1. Toric RUR for square systems: Algorithm

Algorithm RUR_toric_square

Input: $f_1, \ldots, f_n \in \mathbb{K}[X_1, \ldots, X_n]$ with supports $A_1, \ldots, A_n \subseteq \mathbb{Z}^n$.

Output: $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for some finite set Z' which contains all the isolated common roots of the input system in $(\overline{\mathbb{K}}^*)^n$ as well as at least one point from every irreducible component of the zero set of the input system in $(\overline{\mathbb{K}}^*)^n$.

1: $A_0 \leftarrow \{\boldsymbol{o}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ where $\boldsymbol{o}$ is the origin and $\boldsymbol{b}_i$ is the ith standard basis vector in $\mathbb{R}^n$
2: compute MV_{-0} of the convex hulls of $A_1, \ldots, A_n$
3: if $\mathrm{MV}_{-0} = 0$ then
4: for $i := 1, \ldots, n$ do:
5: choose a point $a \in \mathbb{Z}^n$ with random coordinates and $A_i \leftarrow \{a\} \cup A_i$
6: go to **2**
7: compute $\mathrm{MV}_{-1}, \ldots, \mathrm{MV}_{-n}$ and construct the toric resultant matrix N for a system of polynomials with supports $A_0, A_1, \ldots, A_n$
8: $(u_0, u_1, u_2, \ldots, u_n) \leftarrow (1, 0, 0, \ldots, 0)$
9: choose polynomials $f_1^*, \ldots, f_n^*$ in variables $X_1, \ldots, X_n$ with random coefficients in $\mathbb{K}$ and supports contained in $A_1, \ldots, A_n$
10: set d to be a non-negative integer such that s^d is the lowest degree term with non-zero coefficient in $\mathrm{TGCP}(s, \boldsymbol{u})$ (regarded as a polynomial in s)
11: if $\mathrm{TGCP}(s, \boldsymbol{u})$ is identically zero then
12: go to **9**
13: $u \leftarrow 0$ and $\overline{M} \leftarrow 0$
14: $(u_1, u_2, \ldots, u_n) \leftarrow \left(1, u, \ldots, u^{n-1}\right)$
15: compute $p(T) := \mathrm{TPert}(T, u_1, \ldots, u_n)$ where $\mathrm{TPert}(\boldsymbol{u})$ is the coefficient of the term s^d in $\mathrm{TGCP}(s, \boldsymbol{u})$ (regarded as a polynomial in s)
16: compute the square-free part $q(T)$ of $p(T)$
17: if $\deg_T p(T) = \deg_T q(T)$ then /* if $p(T)$ is square-free then */
18: $M \leftarrow \deg_T p(T)$
19: else /* if $p(T)$ is not square-free then */
20: if $u \leq n\binom{\mathrm{MV}_{-0}}{2}$ then
21: if $\overline{M} < \deg_T q(T)$ then
22: $\overline{M} \leftarrow \deg_T q(T)$
23: increment u and go to **14**
24: else /* if $u > n\binom{\mathrm{MV}_{-0}}{2}$ then */
25: $M \leftarrow \overline{M}$
26: $u \leftarrow 0$
27: $(u_1, u_2, \ldots, u_n) \leftarrow \left(1, u, \ldots, u^{n-1}\right)$
28: compute $p(T) := \mathrm{TPert}(T, u_1, \ldots, u_n)$
29: compute the square-free part $q(T)$ of $p(T)$
30: if $\deg_T q(T) < M$ then
31: increment u and go to **27**
32: for $i := 1, \ldots, n$ do:
33: compute $p_i^{\pm}(t) := \mathrm{TPert}(t, u_1, \ldots, u_{i-1}, u_i \pm 1, u_{i+1}, \ldots, u_n)$ [3]
34: compute the square-free part $q_i^{\pm}(t)$ of $p_i^{\pm}(t)$
35: if $\deg_t q_i^-(t) < M$ or $\deg_t q_i^+(t) < M$ then
36: increment u and go to **27**
37: $h(T) \leftarrow q(T)$
38: for $i := 1, \ldots, n$ do:

[3] Step **33** does not make sense when the characteristic of $\mathbb{K}$ is 2, but we assume that the characteristic of $\mathbb{K}$ is 0 or sufficiently large.

39: compute the first subresultants[4] $r_{i,0}$ and $r_{i,1}$ of $q_i^-(t)$ and $q_i^+(2T-t)$ regarded as polynomials in the variable t with coefficients in $\mathbb{K}[T]$
/* $r_{i,0}$ and $r_{i,1}$ are polynomials in $\mathbb{K}[T]$ */

40: compute $h_i(T) := -T - \frac{r_{i,1}(T)}{r_{i,0}(T)} \bmod h(T)$

The algorithm differs from the prior version (Rojas, 1999a) in the following:

- The loop from step **3** through step **6** handles the input system when MV_{-0} is 0.
- Step **10** uses a new criteria (see Proposition 6) to determine a non-negative integer d such that s^d is the lowest degree term with non-zero coefficient in TGCP $(s, \boldsymbol{u})$.
- The specialization of $\boldsymbol{u}$ is derandomized:
  The loop from step **13** through step **25** finds an appropriate specialization of parameters $u_1, \ldots, u_n$ for computing an upper bound M for the cardinality M' of Z' (counting without multiplicity).
  Steps **26** through **36** find an appropriate specialization of parameters $u_1, \ldots, u_n$ for computing all the univariate polynomials h and $h_1, \ldots, h_n$ in the RUR.

3.1.2. Toric RUR for square systems: Description

Step **2** computes MV_{-0}.

The loop from step **3** through step **6** adds points to $A_1, \ldots, A_n$ so that MV_{-0} is assured to be strictly positive. Those points are chosen randomly or deterministically. For efficiency, we use a randomized method.

Step **7** computes $\mathrm{MV}_{-1}, \ldots, \mathrm{MV}_{-n}$ and constructs the toric resultant matrix N for a system of $n+1$ polynomials with supports $A_0, A_1, \ldots, A_n$. Entries of matrix N remain undetermined, and will be specialized to some values later at steps **10**, **15**, **29** and **33**. Step **7** needs to be performed once and only once for any square system of n polynomials in n variables with given supports $A_1, \ldots, A_n$.

The loop from step **8** through step **12** determines a non-negative integer d such that TPert $(\boldsymbol{u})$ is the coefficient of the term s^d in TGCP $(s, \boldsymbol{u})$.

We will show, in Proposition 6, that if s^d is the lowest degree term with non-zero coefficient in TGCP $(s, 1, 0, 0, \ldots, 0)$ then TPert $(\boldsymbol{u})$ is the coefficient of the term s^d in TGCP $(s, \boldsymbol{u})$. Thus, parameters $u_0, u_1, \ldots, u_n$ are specialized to $1, 0, \ldots, 0$ at step **8** and fixed throughout the loop.

Step **9** chooses auxiliary polynomials $f_1^*, \ldots, f_n^*$. While randomly chosen f_i^*'s could turn out not to be suitable, this is almost never the case, and is detected at step **11** if they are. As mentioned earlier, there is a deterministic method for choosing suitable auxiliary polynomials (Rojas, 1999a), but the method is costly, and suffers from expression swell. Thus, we stick with the randomized method.

Step **10** determines a non-negative integer d such that s^d is the lowest degree term with non-zero coefficient in TGCP $(s, 1, 0, 0, \ldots, 0)$. From (5), $\deg_s \mathrm{TGCP}(s, 1, 0, 0, \ldots, 0) = \sum_{i=1}^{n} \mathrm{MV}_{-i}$, the right-hand side of which can easily be calculated from the quantities computed at step **7**, and thus, all the coefficients of TGCP $(s, 1, 0, 0, \ldots, 0)$ can be computed via interpolation. More precisely, choose $\sum_{i=1}^{n} \mathrm{MV}_{-i} + 1$ many values for s, specialize the entries of N with the coefficients of f_0 (which is the constant polynomial 1 here), $f_1 - sf_1^*, \ldots, f_n - sf_n^*$, evaluate TGCP $(s, 1, 0, 0, \ldots, 0)$, and interpolate TGCP $(s, 1, 0, 0, \ldots, 0)$ from these values.

[4] (Canny (1990) and Gonzalez-Vega (1991)).

Recall that the determinant of the resultant matrix N is some non-trivial multiple of the toric resultant. In order to calculate the explicit value of TGCP $(s, \boldsymbol{u})$ at fixed s and $\boldsymbol{u}$, the contribution of the extraneous factor must be eliminated. An elimination of the extraneous factor is done by another level of interpolation through the values of the determinant of N whose entries are specialized in several ways. One such method is called the division method (Canny and Emiris, 2000), which applies to both cases — when the characteristic of $\mathbb{K}$ is 0 or positive.

If the characteristic of $\mathbb{K}$ is 0 then d can be determined by scanning the coefficients of some non-trivial multiple of TGCP $(s, 1, 0, 0, \ldots, 0)$ instead of the coefficients of TGCP $(s, 1, 0, 0, \ldots, 0)$ without any extraneous factor. From (5), $\deg_s$ TGCP $(s, 1, 0, 0, \ldots, 0)$ is known to be bounded from above by $\dim N - \mathrm{MV}_{-0}$, and thus, some non-trivial multiple of TGCP $(s, 1, 0, 0, \ldots, 0)$ can be computed via interpolation from the values of $\det N$. More precisely, choose $\dim N - \mathrm{MV}_{-0} + 1$ many values for s, specialize the entries of N with the coefficients of f_0 (which is the constant polynomial 1 here), $f_1 - sf_1^*, \ldots, f_n - sf_n^*$, evaluate $\det N$, and interpolate.

Step **11** checks whether or not the randomly chosen f_i^*'s at step **9** are suitable.

The loop from step **13** through step **25** finds an appropriate specialization of parameters $u_1, \ldots, u_n$ for computing an upper bound M for the cardinality M' of Z' (counting without multiplicity).

Step **14** specializes parameters $u_1, \ldots, u_n$ to some integer values.

Step **15** computes $p(T) := \mathrm{TPert}(T, u_1, \ldots, u_n)$ introducing a variable T. From (4), $\deg_T \mathrm{TPert}(T, u_1, \ldots, u_n) = \mathrm{MV}_{-0}$, the right-hand side of which has been computed at step **7**, and thus, $\mathrm{TPert}(T, u_1, \ldots, u_n)$ can be computed via interpolation: choose $\mathrm{MV}_{-0} + 1$ many values for u_0, evaluate the coefficient $\mathrm{TPert}(\boldsymbol{u}) = \mathrm{TPert}(u_0, u_1, \ldots, u_n)$ of the term s^d in TGCP $(s, \boldsymbol{u})$, and interpolate $\mathrm{TPert}(T, u_1, \ldots, u_n)$ from these values. From (5), $\deg_s$ TGCP $(s, \boldsymbol{u})$ is known, and thus, the coefficient of the term s^d in TGCP $(s, \boldsymbol{u})$ can be computed via another level of interpolation: choose $\sum_{i=1}^{n} \mathrm{MV}_{-i} + 1$ many values for s, specialize the entries of N with the coefficients of $f_0, f_1 - sf_1^*, \ldots, f_n - sf_n^*$, calculate the explicit values of TGCP $(s, \boldsymbol{u})$, and interpolate TGCP $(s, \boldsymbol{u})$ from these values. Note that calculation of the explicit values of TGCP $(s, \boldsymbol{u})$ requires an elimination of the extraneous factor from $\det N$.

The above paragraph holds when the characteristic of $\mathbb{K}$ is 0 or positive. If the characteristic of $\mathbb{K}$ is 0 then $\mathrm{TPert}(T, u_1, \ldots, u_n)$ can be computed via interpolation from the values of the coefficient of the term s^d in some non-trivial multiple of TGCP $(s, \boldsymbol{u})$ instead of the values of the coefficient of the term s^d in TGCP $(s, \boldsymbol{u})$ without any extraneous factor. This is possible because, by (4), the contributions of the extraneous factor are independent of $\boldsymbol{u}$, in particular u_0, and will be canceled out during interpolation. More precisely, choose $\mathrm{MV}_{-0} + 1$ many values for u_0, evaluate the coefficient of the term s^d in some non-trivial multiple of TGCP $(s, \boldsymbol{u})$, and interpolate $\mathrm{TPert}(T, u_1, \ldots, u_n)$ from these values. The coefficient of the term s^d in some non-trivial multiple of TGCP $(s, \boldsymbol{u})$ is computed via another level of interpolation: choose $\dim N - \mathrm{MV}_{-0} + 1$ many values for s, specialize the entries of N with the coefficients of $f_0, f_1 - sf_1^*, \ldots, f_n - sf_n^*$, evaluate $\det N$, and interpolate.

Step **16** computes the square-free part $q(T)$ of $p(T) := \mathrm{TPert}(T, u_1, \ldots, u_n)$ by dividing $p(T)$ by the greatest common divisor of $p(T)$ and its derivative $p'(T)$ found using the Euclidean algorithm.

Steps **17** through **25** find M. If, for some specialization of parameters $u_1, \ldots, u_n$, $p(T) := \mathrm{TPert}(T, u_1, \ldots, u_n)$ is square-free then $M = \deg_T p(T)$ and the computation immediately exits from the loop. On the contrary, if $p(T)$ remains non-square-free for all $n\binom{\mathrm{MV}_{-0}}{2} + 1$ many

specializations of parameters $u_1, \ldots, u_n$ then M is set to be the maximum degree of the square-free part $q(T)$ of $p(T)$. The correctness of this part of the algorithm will be shown later.

Steps **26** through **36** find an appropriate specialization of parameters $u_1, \ldots, u_n$ for computing all the univariate polynomials h and $h_1, \ldots, h_n$ in the RUR.

Step **27** specializes parameters $u_1, \ldots, u_n$ to some integer values.

Steps **28** and **29** are the same as steps **15** and **16**, respectively. In the loop from step **13** through step **25**, we have already tried several specializations of parameters $u_1, \ldots, u_n$, and have found at least one appropriate specialization (possibly more if step **25** has been reached) for computing M and possibly some inappropriate ones. For those specializations of parameters $u_1, \ldots, u_n$ that have been tried in the previous loop, steps **28** and **29** do not need to be performed. If a specialization of parameters $u_1, \ldots, u_n$ has been found inappropriate then the computation immediately goes back to step **27**. On the other hand, if a specialization of parameters $u_1, \ldots, u_n$ has been found appropriate for computing M, which means that it is appropriate for computing h, then the computation jumps to step **32** and checks whether or not it is also appropriate for computing $h_1, \ldots, h_n$.

Similar to step **15**, at step **33**, $p_i^{\pm}(t)$ are computed via interpolations, and similar to step **16**, at step **34**, $q_i^{\pm}(t) := p_i^{\pm}(t) \,/\, \gcd\left(p_i^{\pm}(t)\,, \left(p_i^{\pm}\right)'(t)\right)$.

Step **37** determines the univariate polynomial h in the RUR, and the loop from step **38** through step **40** determines the univariate polynomials $h_1, \ldots, h_n$ in the RUR.

Step **39** computes the *first subresultants* r_0 and r_1 of $q_i^-(t)$ and $q_i^+(2T - t)$ regarded as polynomials in the variable t with coefficients in $\mathbb{K}[T]$ (Canny, 1990; Gonzalez-Vega, 1991). By definition, r_0 and r_1 are the determinants of some submatrices of the Sylvester matrix for $q_i^-(t)$ and $q_i^+(2T - t)$ in $(\mathbb{K}[T])[t]$. Each of these submatrices consists of $M - 1$ rows whose entries are the coefficients of $q_i^-(t)$ (or 0) and $M - 1$ rows whose entries are the coefficients of $q_i^+(2T - t)$ (or 0). Since the coefficients of $q_i^+(2T - t)$ are actually polynomials in $\mathbb{K}[T]$ of degree M, r_0 and r_1 are also polynomials in $\mathbb{K}[T]$ and both are of degree $M(M - 1)$. Thus, they can be computed via interpolation: choose $M(M - 1)$ many values for T, specialize the entries of these submatrices with the coefficients of $q_i^-(t) \in \mathbb{K}[t]$ and $q_i^+(2T - t) \in (\mathbb{K}[T])[t]$, evaluate the determinant of these submatrices and interpolate $r_0(T)$ and $r_1(T)$ from these values of the determinant.

Step **40** computes univariate polynomials $h_1(T), \ldots, h_n(T)$ with coefficients in $\mathbb{K}$. We will show that whenever parameters $u_1, \ldots, u_n$ are appropriately specialized, if $\theta \in \overline{\mathbb{K}}$ is a root of $h(T) \in \mathbb{K}[T]$ then $(h_1(\theta), \ldots, h_n(\theta))$ is either a root of the input system or not toric.

3.1.3. Toric RUR for square systems: Proof for correctness

The following proposition completes the proof of the correctness of the loop from step **8** though step **12** of the algorithm.

Proposition 6. *If s^d is the lowest degree term with non-zero coefficient in* TGCP $(s, 1, 0, 0, \ldots, 0)$ *then* TPert $(\boldsymbol{u})$ *is the coefficient of the term s^d in* TGCP $(s, \boldsymbol{u})$.

Proof. Suppose otherwise. Then, there exists a non-negative integer $e < d$ such that TPert $(\boldsymbol{u})$ is the non-zero coefficient of the term s^e in TGCP $(s, \boldsymbol{u})$. By Theorem 1, TPert $(\boldsymbol{u})$ splits into (not necessarily distinct) linear factors:

$$\text{TPert}\,(\boldsymbol{u}) = c \prod_{j=1}^{M} \left(u_0 + \sum_{l=1}^{n} u_l \zeta_l^{(j)} \right)^{\mu^{(j)}} \tag{6}$$

where c is a non-zero constant belonging to $\mathbb{K}$ and $\mu^{(1)}, \ldots, \mu^{(M)}$ are positive integers. Thus, TPert $(1, 0, 0, \ldots, 0) \neq 0$. This is a contradiction, since the coefficient TPert $(1, 0, 0, \ldots, 0)$ of the term s^e in TGCP $(s, 1, 0, \ldots, 0)$ is not identically zero but $e < d$. □

In the rest of this section, we describe the conditions for an appropriate specialization of parameters $u_1, \ldots, u_n$, the existence of appropriate specializations and the remaining proofs of the correctness of the algorithm.

We use the famous concept of separating polynomials and their properties. For more details, see textbooks like Basu et al. (2003).

A polynomial f in n variables with coefficients in a field $\mathbb{L}$ is said to *separate* two distinct points α and β in $\mathbb{L}^n$ if $f(\alpha) \neq f(\beta)$. A polynomial f in n variables with coefficients in $\mathbb{L}$ is said to *separate* a finite subset A of $\mathbb{L}^n$ if f separates every pair of two distinct points in A.

Lemma 7. *Let $\mathbb{L}$ be a field of characteristic 0 or a finite field of characteristic at least $n+1$. Furthermore, let α and β be two distinct points in $\mathbb{L}^{n+1}$. Then, at least one of the linear polynomials in $n+1$ variables $X_0, X_1, \ldots, X_n$ with integer coefficients*

$$v_u = X_0 + uX_1 + \cdots + u^n X_n, \quad u = 0, 1, \ldots, n, \tag{7}$$

separates α and β.

We describe the conditions for an appropriate specialization of parameters $u_1, \ldots, u_n$.

Recall that Z' is some finite subset of the zero set of the input system of polynomials with coefficients in $\mathbb{K}$ such that the univariate polynomial $h(T)$ in the RUR for Z' is derived from TPert $(\boldsymbol{u})$ by setting u_0 to a variable T and specializing parameters $u_1, \ldots, u_n$ to some appropriate values in $\overline{\mathbb{K}}$. Let M' be the cardinality of Z' so that

$$Z' = \left\{ \left(\zeta_1^{(1)}, \ldots, \zeta_n^{(1)} \right), \ldots, \left(\zeta_1^{(M')}, \ldots, \zeta_n^{(M')} \right) \right\}. \tag{8}$$

By Bernstein's theorem (Bernstein, 1975)

$$M' \leq M \leq \mathrm{MV}_{-0} = \deg_T \mathrm{TPert}\,(T, u_1, \ldots, u_n)\,. \tag{9}$$

We say that parameters $u_1, \ldots, u_n$ are appropriately specialized if the linear polynomials in n variables

$$u_1 X_1 + \cdots + u_n X_n \tag{10}$$

and

$$u_1 X_1 + \cdots + u_{i-1} X_{i-1} + (u_i \pm 1)\, X_i + u_{i+1} X_{i+1} + \cdots + u_n X_n, \quad i = 1, \ldots, n \tag{11}$$

separate Z', or equivalently, the following conditions are satisfied:

$$j \neq k \Rightarrow \sum_{l=1}^{n} u_l \zeta_l^{(j)} \neq \sum_{l=1}^{n} u_l \zeta_l^{(k)} \tag{12}$$

and

$$j \neq k \Rightarrow \sum_{l=1}^{n} u_l \zeta_l^{(j)} \pm \zeta_i^{(j)} \neq \sum_{l=1}^{n} u_l \zeta_l^{(k)} \pm \zeta_i^{(k)}, \quad i = 1, \ldots, n. \tag{13}$$

We show that there always exists an appropriate specialization of parameters $u_1, \ldots, u_n$.

Proposition 8. *At least one of the n-tuples in*

$$\left\{(u_1,\ldots,u_n)=\left(u,\ldots,u^n\right)\;\middle|\; u=0,1,\ldots,n\binom{\mathrm{MV}_{-0}}{2}\right\}$$

satisfies (12).

Proof. Let u be a non-negative integer and $v_u=\sum_{i=0}^{n}u^iX_i$ as in (7). Define

$$Y'=\left\{\left(0,\zeta_1^{(j)},\ldots,\zeta_n^{(j)}\right)\;\middle|\; j=1,\ldots,M'\right\}. \tag{14}$$

Condition (12) holds if there exists a non-negative integer u such that v_u separates $\binom{M'}{2}$ pairs of distinct points in Y'. Now, apply Lemma 7 and (9). □

Proposition 9. *At least one of the n-tuples in*

$$\left\{(u_1,\ldots,u_n)=\left(u,\ldots,u^n\right)\;\middle|\; u=0,1,\ldots,(2n+1)\,n\binom{M}{2}\right\}$$

satisfies (12) *and* (13).

Proof. Let u be a non-negative integer and $v_u=\sum_{i=0}^{n}u^iX_i$ as in (7). Define Y' as in (14) and define

$$Y'^{\pm}_i=\left\{\left(\pm\zeta_i^{(j)},\zeta_1^{(j)},\zeta_2^{(j)},\ldots,\zeta_n^{(j)}\right)\;\middle|\; j=1,\ldots,M'\right\},\quad i=1,\ldots,n.$$

Note that the cardinality of each $Y'^{\pm}_i$ is M'. Conditions (12) and (13) hold if there exists a non-negative integer u such that v_u separates $\binom{M'}{2}$ pairs of distinct points in Y', $\binom{M'}{2}$ pairs of distinct points in Y'^{+}_i for $i=1,\ldots,n$, and $\binom{M'}{2}$ pairs of distinct points in Y'^{-}_i for $i=1,\ldots,n$, in total, $(2n+1)\binom{M'}{2}$ pairs of distinct points. Now, apply Lemma 7. □

Remark 10. Parameters $u_1,\ldots,u_n$ are appropriately specialized to some integers.[5]

We complete the proof of the correctness of the algorithm.

First, we show that condition (12) holds iff, at step **18** or step **25**, M is correctly set.

By Theorem 1, TPert $(T,u_1,\ldots,u_n)$ splits into (not necessarily distinct) linear factors:

$$\mathrm{TPert}\,(T,u_1,\ldots,u_n)=c\prod_{j=1}^{M}\left(T+\sum_{l=1}^{n}u_l\zeta_l^{(j)}\right)^{\mu^{(j)}} \tag{15}$$

where $c\in\mathbb{K}^*$ and $\mu^{(1)},\ldots,\mu^{(M)}$ are some positive integers.

Suppose TPert $(u_1,\ldots,u_n)$ is not square-free. The possible situations are

(1) $\mu^{(j)}\geq 2$ for some j, and/or
(2) parameters $u_1,\ldots,u_n$ are inappropriately specialized so that condition (12) does not hold.

[5] Since we assume that the characteristic of $\mathbb{K}$ is 0 or sufficiently large, $\mathbb{K}$ always contains integers.

If $\mu^{(1)} = \cdots = \mu^{(M)} = 1$ then, by Proposition 8, within finitely many attempts, a specialization of parameters $u_1, \ldots, u_n$ satisfying condition (12) will be found eventually to compute TPert $(T, u_1, \ldots, u_n)$ which becomes square-free. Thus, the computation reaches step **25** only if $\mu^{(j)} \geq 2$ for some j. In this case, $n\binom{\mathrm{MV}-0}{2} + 1$ many specializations of parameters $u_1, \ldots, u_n$ are tried. Again, by Proposition 8, at least one of them must satisfy condition (12). Whenever a specialization of parameters $u_1, \ldots, u_n$ satisfying condition (12) is used, $\deg_T q(T)$ at step **21**, which precisely matches the number of distinct linear factors of TPert $(T, u_1, \ldots, u_n)$, is maximized, since, with any inappropriate specialization of parameters $u_1, \ldots, u_n$ breaching condition (12), TPert $(T, u_1, \ldots, u_n)$ must have fewer distinct linear factors.

Next, we show that conditions (13) hold iff, at step **30**, $\deg_T q(T) = M$ and simultaneously, at step **35**, $\deg_t q_i^{\pm}(t) = M$ for $i = 1, \ldots, n$.

By (15), TPert $(t, u_1, \ldots, u_{i-1}, u_i \pm 1, u_{i+1}, \ldots, u_n)$ splits into (not necessarily distinct) linear factors:

$$\begin{aligned}&\text{TPert}\,(t, u_1, \ldots, u_{i-1}, u_i \pm 1, u_{i+1}, \ldots, u_n)\\ &\quad = c \prod_{j=1}^{M} \left(t + \sum_{l=1}^{n} u_l \zeta_l^{(j)} \pm \zeta_i^{(j)} \right)^{\mu^{(j)}}, \quad i = 1, \ldots, n.\end{aligned} \tag{16}$$

If conditions (13) hold then every $q_i^{\pm}(t)$ computed at step **34** is a product of M distinct linear factors:

$$q_i^{\pm}(t) = c \prod_{j=1}^{M} \left(t + \sum_{l=1}^{n} u_l \zeta_l^{(j)} \pm \zeta_i^{(j)} \right), \quad i = 1, \ldots, n. \tag{17}$$

Thus, $\deg_t q_i^{\pm}(t) = M$ for $i = 1, \ldots, n$.

On the other hand, if not all conditions (13) hold then, for some i, $q_i^{-}(t)$ or $q_i^{+}(t)$ has strictly fewer than M distinct linear factors. Thus, $\deg_t q_i^{-}(t) < M$ or $\deg_t q_i^{+}(t) < M$ for some i.

Furthermore, by Proposition 9, an appropriate specialization of parameters $u_1, \ldots, u_n$ satisfying conditions (12) and (13) will be found eventually, which results in $\deg_T q(T) = \deg_t q_i^{\pm}(t) = M$ for $i = 1, \ldots, n$. This shows that the computation eventually exits from the loop from step **26** through step **36**.

Finally, we show that whenever parameters $u_1, \ldots, u_n$ are appropriately specialized, if $\theta \in \overline{\mathbb{K}}$ is a root of $h(T) \in \mathbb{K}[T]$ then $(h_1(\theta), \ldots, h_n(\theta))$ is either a root of the input system or not toric.

It follows that, provided condition (12) holds, at step **37**, $h(T)$ is a product of M distinct linear factors in $\overline{\mathbb{K}}[T]$:

$$h(T) = c \prod_{j=1}^{M} \left(T + \sum_{l=1}^{n} u_l \zeta_l^{(j)} \right), \tag{18}$$

and thus, $h(T)$ has precisely M distinct roots $\theta^{(1)}, \ldots, \theta^{(M)}$ in $\overline{\mathbb{K}}$ where

$$\theta^{(j)} = -\sum_{l=1}^{n} u_l \zeta_l^{(j)}, \quad j = 1, \ldots, M. \tag{19}$$

Substituting (19) into (17), we see that, provided (12) and (13) hold, $q_i^-(t)$ and $q_i^+(2T-t)$ split into linear factors over $\overline{\mathbb{K}}$:

$$\begin{aligned} q_i^-(t) &= c\prod_{j=1}^{M}\left(t-\left(\theta^{(j)}+\zeta_i^{(j)}\right)\right),\\ q_i^+(2T-t) &= c\prod_{j=1}^{M}\left(2T-t-\left(\theta^{(j)}-\zeta_i^{(j)}\right)\right), \end{aligned} \qquad i=1,\ldots,n. \tag{20}$$

Fixing $T=\theta^{(j)}$ in (20), we claim that $q_i^-(t)$ and $q_i^+\left(2\theta^{(j)}-t\right)$ have a common linear factor $t-\left(\theta^{(j)}+\zeta_i^{(j)}\right)$ in $\overline{\mathbb{K}}[t]$:

$$q_i^-(t)=q_i^+\left(2\theta^{(j)}-t\right)=0\Leftrightarrow t=\theta^{(j)}+\zeta_i^{(j)},\quad \begin{matrix} i=1,\ldots,n,\\ j=1,\ldots,M. \end{matrix} \tag{21}$$

In fact, any root of $q_i^-(t)$ in $\overline{\mathbb{K}}$ is of the form $\theta^{(k)}+\zeta_i^{(k)}$ for some k. But, if $k\neq j$ then $q_i^+\left(2\theta^{(j)}-t\right)$ is not square-free as a polynomial in $\overline{\mathbb{K}}[t]$.

We now show that, for every $h_i(T)$ computed at step **40**, $h_i\left(\theta^{(j)}\right)=\zeta_i^{(j)}$ for $j=1,\ldots,M'$.

Fix any $i\in\{1,\ldots,n\}$ and $j\in\{1,\ldots,M\}$. It is known (Canny, 1990; Gonzalez-Vega, 1991) that $r_{i,0}^*$ and $r_{i,1}^*$ are the first subresultants of a pair of univariate polynomials $q_i^-(t)$ and $q_i^+\left(2\theta^{(j)}-t\right)$ in the variable t with coefficients in $\overline{\mathbb{K}}$ then the ratio $\frac{r_{i,1}^*}{r_{i,0}^*}$ matches the ratio of the constant term $-\left(\theta^{(j)}+\zeta_i^{(j)}\right)$ and the coefficient 1 of the linear term of the common linear factor $t-\left(\theta^{(j)}+\zeta_i^{(j)}\right)$ of $q_i^-(t)$ and $q_i^+\left(2\theta^{(j)}-t\right)$ in $\overline{\mathbb{K}}[t]$: $\frac{r_{i,1}^*}{r_{i,0}^*}=-\left(\theta^{(j)}+\zeta_i^{(j)}\right)$.

Then

$$h_i\left(\theta^{(j)}\right)=-\theta^{(j)}-\frac{r_{i,1}\left(\theta^{(j)}\right)}{r_{i,0}\left(\theta^{(j)}\right)}=-\theta^{(j)}-\frac{r_{i,1}^*}{r_{i,0}^*}=-\theta^{(j)}+\left(\theta^{(j)}+\zeta_i^{(j)}\right)=\zeta_i^{(j)}.$$

Of course, the above argument does not make sense if some root $\theta^{(j)}$ of $h(T)$ corresponds to some common root of the input system outside of the torus $(\mathbb{C}^*)^n$. In such a case, some of the univariate polynomials $h_1,\ldots,h_n$ identically vanish, and we can effectively ignore them. The number of common roots of the input system outside of the torus $(\mathbb{C}^*)^n$ is the difference between M and M'.

3.2. RUR

Recall that Algorithm **RUR_toric_square** computes the RUR for the toric zero set (the zero set in $(\overline{\mathbb{K}}^*)^n$) of a square system. We would like to develop an algorithm for computing the RUR for the affine zero set (the zero set in $\overline{\mathbb{K}}^n$) for a system which is not necessarily square.

3.2.1. RUR for square systems

Consider a square system of polynomials $f_1,\ldots,f_n$ in n variables with coefficients in $\mathbb{K}$. Let Z be the zero set of the system. Write $A_1,\ldots,A_n$ for the supports of $f_1,\ldots,f_n$, respectively. It is known (Li and Wang, 1996; Rojas, 1999a, 2000) that if, in Algorithm **RUR_toric_square**, instead of $A_1,\ldots,A_n$, the sets $\{\boldsymbol{o}\}\cup A_1,\ldots,\{\boldsymbol{o}\}\cup A_n$ are used then the algorithm computes the RUR for some finite set $\overline{Z}'$ that contains all the isolated common roots of the input system (in

$\overline{\mathbb{K}}^n$) as well as at least one point from every irreducible component of $Z(\subseteq \overline{\mathbb{K}}^n)$. Note that $\overline{Z}'$ may contain some extraneous points which do not belong to Z.

Algorithm RUR_square

Input: $f_1, \ldots, f_n \in \mathbb{K}[X_1, \ldots, X_n]$.

Output: $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for some finite set $\overline{Z}'$ which contains all the isolated common roots of the input system as well as at least one point from every irreducible component of the zero set of the input system.

1: for $i := 1, \ldots, n$ do:
2: set A_i to be the support of f_i
3: $A_i \leftarrow \{\boldsymbol{o}\} \cup A_i$ where $\boldsymbol{o}$ is the origin in $\mathbb{R}^n$
4: call Algorithm **RUR_toric_square** on the input $f_1, \ldots, f_n$ and $A_1, \ldots, A_n$ to compute $h, h_1, \ldots, h_n \in \mathbb{K}[T]$.

3.2.2. RUR for overdetermined systems

The method described here is different from Keyser et al. (2005). In the previous paper, linear algebra is used while, in this paper, algebraic geometry is used. The previous approach produces some extra points other than the common roots of the input system, and, in order to get rid of them, the extra work is required. However, the resultant matrix used in the previous approach is generically smaller than the resultant matrix used in the new approach.

Throughout this paragraph, let $\mathbb{L}$ be an algebraically closed field containing $\mathbb{K}$.

Let $f_1, \ldots, f_m$ be polynomials in n variables with coefficients in $\mathbb{L}$. Write $\mathbf{Z}^{(n)}(f_1, \ldots, f_m)$ for the zero set of polynomials $f_1, \ldots, f_m$ in $\mathbb{L}^n$:

$$\mathbf{Z}^{(n)}(f_1, \ldots, f_m) = \left\{(z_1, \ldots, z_n) \in \mathbb{L}^n \mid f_1(z_1, \ldots, z_n) = \cdots = f_m(z_1, \ldots, z_n) = 0\right\}.$$

We will use the following facts. The proofs are found in textbooks of algebraic geometry, e.g., Cox et al. (1996).

- Let f_1 and f_2 be polynomials in n variables with coefficients in $\mathbb{L}$. Then

$$\mathbf{Z}^{(n)}(f_1, f_2) = \mathbf{Z}^{(n)}(f_1) \cap \mathbf{Z}^{(n)}(f_2). \tag{22}$$

  and

$$\mathbf{Z}^{(n)}(f_1 \cdot f_2) = \mathbf{Z}^{(n)}(f_1) \cup \mathbf{Z}^{(n)}(f_2). \tag{23}$$

- An algebraic set Z in $\mathbb{L}^n$ is written as a finite union of irreducible algebraic sets in $\mathbb{L}^n$:

$$Z = V_1 \cup \cdots \cup V_l \tag{24}$$

  where $V_1, \ldots, V_l$ are irreducible algebraic sets in $\mathbb{L}^n$. A decomposition (24) of Z is said to be minimal if $V_i \not\subseteq V_j$ for $i \neq j$. A minimal decomposition of Z always exists and is unique up to the order in which $V_1, \ldots, V_l$ are written.
- Let Z_1 and Z_2 be irreducible algebraic sets in $\mathbb{L}^m$ and $\mathbb{L}^n$, respectively. Then, $Z_1 \times Z_2$ is an irreducible algebraic set in $\mathbb{L}^{m+n}$.

In the rest of this paragraph, assume that $m > n$.

Let $f_1, \ldots, f_m$ be polynomials in n variables $X_1, \ldots, X_n$ with coefficients in $\mathbb{L}$. Introducing $m - n$ variables $X_{n+1}, \ldots, X_m$, $f_1, \ldots, f_m$ are seen as polynomials in m variables $X_1, \ldots, X_n, X_{n+1}, \ldots, X_m$ with coefficients in $\mathbb{L}$. Thus, $\mathbf{Z}^{(m)}(f_1, \ldots, f_m)$ is well-defined and

$$\mathbf{Z}^{(m)}(f_1, \ldots, f_m) = \left\{ (z_1, \ldots, z_n, z_{n+1}, \ldots, z_m) \in \mathbb{L}^m \;\middle|\; \begin{array}{l} (z_1, \ldots, z_n) \in \mathbf{Z}^{(n)}(f_1, \ldots, f_m), \\ (z_{n+1}, \ldots, z_m) \in \mathbb{L}^{m-n} \end{array} \right\}.$$

Let Π denote the projection from $\mathbb{L}^m$ onto $\mathbb{L}^n$ which ignores the last $m - n$ coordinates:

$$\Pi : \mathbb{L}^m \ni (z_1, \ldots, z_n, z_{n+1}, \ldots, z_m) \mapsto (z_1, \ldots, z_n) \in \mathbb{L}^n.$$

Proposition 11. *Let $f_1, \ldots, f_m$ be polynomials in n variables $X_1, \ldots, X_n$ with coefficients in $\mathbb{L}$. Furthermore, let*

$$\mathbf{Z}^{(n)}(f_1, \ldots, f_m) = V_1 \cup \cdots \cup V_l \tag{25}$$

be the unique minimal decomposition of $\mathbf{Z}^{(n)}(f_1, \ldots, f_m)$ into a union of distinct irreducible algebraic sets in $\mathbb{L}^n$. For $k = 1, \ldots, l$, define

$$W_k = V_k \times \mathbb{L}^{m-n} = \left\{ (z_1, \ldots, z_n, z_{n+1}, \ldots, z_m) \in \mathbb{L}^m \;\middle|\; \begin{array}{l} (z_1, \ldots, z_n) \in V_k, \\ (z_{n+1}, \ldots, z_m) \in \mathbb{L}^{m-n} \end{array} \right\}.$$

Then

$$\mathbf{Z}^{(m)}(f_1, \ldots, f_m) = W_1 \cup \cdots \cup W_l \tag{26}$$

is the unique minimal decomposition of $\mathbf{Z}^{(m)}(f_1, \ldots, f_m)$ into a union of distinct irreducible algebraic sets in $\mathbb{L}^m$.

Proof. It is easy to see that equality (26) holds. It remains to be shown that (26) is a decomposition of $\mathbf{Z}^{(m)}(f_1, \ldots, f_m)$ into a union of irreducible algebraic sets in $\mathbb{L}^m$ and is actually the unique minimal decomposition.

For $k = 1, \ldots, l$, a product W_k of an irreducible algebraic set V_k in $\mathbb{L}^n$ and an irreducible algebraic set $\mathbb{L}^{m-n}$ is an irreducible algebraic set in $\mathbb{L}^m$.

Because of the minimality of the decomposition (25) of $\mathbf{Z}^{(n)}(f_1, \ldots, f_m)$, $V_i \not\subseteq V_j$ for $i \neq j$, i.e., there exists a point $(z_1, \ldots, z_n) \in V_i \setminus V_j$. Then, for every $m - n$-tuple $(z_{n+1}, \ldots, z_m) \in \mathbb{L}^{m-n}$, $(z_1, \ldots, z_n, z_{n+1}, \ldots, z_m)$ is a point in $W_i \setminus W_j$, i.e., $W_i \not\subseteq W_j$ for $i \neq j$. Thus, (26) is a minimal decomposition of $\mathbf{Z}^{(m)}(f_1, \ldots, f_m)$ into a union of distinct irreducible algebraic sets in $\mathbb{L}^m$, and its uniqueness follows from the minimality. □

Corollary 12. *Let $f_1, \ldots, f_m$ be polynomials in n variables $X_1, \ldots, X_n$ with coefficients in $\mathbb{L}$. Furthermore, let*

$$\mathbf{Z}^{(m)}(f_1, \ldots, f_m) = W_1 \cup \cdots \cup W_l \tag{27}$$

be the unique minimal decomposition of $\mathbf{Z}^{(m)}(f_1, \ldots, f_m)$ into a union of distinct irreducible algebraic sets in $\mathbb{L}^m$. Then

$$\mathbf{Z}^{(n)}(f_1, \ldots, f_m) = \Pi(W_1) \cup \cdots \cup \Pi(W_l) \tag{28}$$

is the unique minimal decomposition of $\mathbf{Z}^{(n)}(f_1, \ldots, f_m)$ *into a union of distinct irreducible algebraic sets in* $\mathbb{L}^n$.

Proof. Let $\mathbf{Z}^{(n)}(f_1, \ldots, f_m) = V_1 \cup \cdots \cup V_{l'}$ be the unique minimal decomposition of $\mathbf{Z}^{(n)}(f_1, \ldots, f_m)$ into a union of distinct irreducible algebraic sets in $\mathbb{L}^n$. By Proposition 11, $\mathbf{Z}^{(m)}(f_1, \ldots, f_m) = V_1 \times \mathbb{L}^{m-n} \cup \cdots \cup V_{l'} \times \mathbb{L}^{m-n}$ is the unique minimal decomposition of $\mathbf{Z}^{(m)}(f_1, \ldots, f_m)$ into a union of distinct irreducible algebraic sets in $\mathbb{L}^m$. Thus, $l' = l$ and, relabeling if necessary, for $k = 1, \ldots, l$, $W_k = V_k \times \mathbb{L}^{m-n}$ which implies $V_k = \Pi(W_k)$. □

Consider a system of m polynomials $f_1, \ldots f_m$ in n variables $X_1, \ldots, X_n$ with coefficients in $\mathbb{K}$. Introducing $m - n$ variables $X_{n+1}, \ldots, X_m$, construct a square system of m polynomials $g_1, \ldots, g_m$ in m variables $X_1, \ldots, X_m$ with coefficients in $\mathbb{K}$:

$$g_i(X_1, \ldots, X_m) = f_i(X_1, \ldots, X_n) \cdot (X_{n+1} - a_{n+1}) \cdot \cdots \cdot (X_m - a_m) \qquad i = 1, \ldots, m, \tag{29}$$

where $a_{n+1} \ldots, a_m$ are some constants in $\mathbb{K}$.

Let

$$\mathbf{Z}^{(m)}(f_1, \ldots, f_m) = W_1 \cup \cdots \cup W_l \tag{30}$$

be the unique minimal decomposition of $\mathbf{Z}^{(m)}(f_1, \ldots, f_m)$ into a union of distinct irreducible algebraic sets in $\overline{\mathbb{K}}^m$.

Proposition 13. *Following the notations above,*

$$\mathbf{Z}^{(m)}(g_1, \ldots, g_m) = W_1 \cup \cdots \cup W_l \cup \mathbf{Z}^{(m)}(X_{n+1} - a_{n+1}) \cup \cdots \cup \mathbf{Z}^{(m)}(X_m - a_m) \tag{31}$$

is the unique minimal decomposition of $\mathbf{Z}^{(m)}(g_1, \ldots, g_m)$ *into a union of distinct irreducible algebraic sets in* $\overline{\mathbb{K}}^m$.

Proof. Equality (31) holds since

$$\begin{aligned}
\mathbf{Z}^{(m)}(g_1, \ldots, g_m) &= \bigcap_{i=1}^{m} \mathbf{Z}^{(m)}(g_i) \\
&= \bigcap_{i=1}^{m} \left(\mathbf{Z}^{(m)}(f_i) \cup \bigcup_{j=n+1}^{m} \mathbf{Z}^{(m)}\left(X_j - a_j\right) \right) \\
&= \bigcap_{i=1}^{m} \mathbf{Z}^{(m)}(f_i) \cup \bigcup_{j=n+1}^{m} \mathbf{Z}^{(m)}\left(X_j - a_j\right) \\
&= \mathbf{Z}^{(m)}(f_1, \ldots, f_m) \cup \bigcup_{j=n+1}^{m} \mathbf{Z}^{(m)}\left(X_j - a_j\right) \\
&= \bigcup_{k=1}^{l} W_k \cup \bigcup_{j=n+1}^{m} \mathbf{Z}^{(m)}\left(X_j - a_j\right)
\end{aligned}$$

where the first and the fourth equalities follow from (22), the second equality follows from (23) and the last equality follows from (30).

All the components appearing on the right-hand side of (31) are irreducible. By assumption, $W_1, \ldots, W_l$ are all irreducible. Each $\mathbf{Z}^{(m)}\left(X_j - a_j\right)$ is the zero set of a linear polynomial $X_j - a_j$, and thus, is irreducible.

Furthermore, all the components appearing on the right-hand side of (31) are distinct. By assumption, $W_i \not\subseteq W_j$ for $i \neq j$. Any pair of W_k and $\mathbf{Z}^{(m)}\left(X_j - a_j\right)$ are distinct because W_k contains a point whose jth coordinate is not a_j. If $i \neq j$ then $\mathbf{Z}^{(m)}\left(X_i - a_i\right) \not\subseteq \mathbf{Z}^{(m)}\left(X_j - a_j\right)$ since the former contains a point whose jth coordinate is not a_j. Hence, the decomposition (31) is minimal, and its uniqueness follows from the minimality. □

Note that all the irreducible components of the unique minimal decomposition of $\mathbf{Z}^{(m)}\left(g_1, \ldots, g_m\right)$ are of positive dimension; each W_k contains a copy of $\overline{\mathbb{K}}^{m-n}$ and each $\mathbf{Z}^{(m)}\left(X_j - a_j\right)$ contains a copy of $\overline{\mathbb{K}}^{m-1}$.

Suppose Algorithm **RUR_square** will be applied to the square system of polynomials $g_1, \ldots, g_m$ in m variables with coefficients in $\mathbb{K}$ and univariate polynomials h and $h_1, \ldots, h_m$ with coefficients in $\mathbb{K}$ are returned. These polynomials form the RUR for some finite set $\overline{Y}'$ that contains at least one point from every irreducible component of $\mathbf{Z}^{(m)}\left(g_1, \ldots, g_m\right)$. By Proposition 13, $\overline{Y}'$ contains at least one point from each W_k: for $k = 1, \ldots, l$, there exists a root θ (in $\overline{\mathbb{K}}$) of h such that $W_k \ni \left(h_1(\theta), \ldots, h_m(\theta)\right)$. By Proposition 11 and Corollary 12, there is a bijective correspondence between irreducible components $V_1, \ldots, V_l$ of $\mathbf{Z}^{(n)}\left(f_1, \ldots, f_m\right)$ and irreducible components $W_1, \ldots, W_l$ of $\mathbf{Z}^{(m)}\left(f_1, \ldots, f_m\right)$, and $V_k = \Pi(W_k)$ for $k = 1, \ldots, l$. Thus, for $k = 1, \ldots, l$, there exists a root θ (in $\overline{\mathbb{K}}$) of h such that $V_k \ni \left(h_1(\theta), \ldots, h_n(\theta)\right)$. Hence, the set

$$\overline{Z}' = \Pi\left(\overline{Y}'\right) = \left\{\left(h_1(\theta), \ldots, h_n(\theta)\right) \mid \theta \in \overline{\mathbb{K}} \text{ with } h(\theta) = 0\right\}$$

contains at least one point from every irreducible component of $\mathbf{Z}^{(n)}\left(f_1, \ldots, f_m\right)$. In particular, $\overline{Z}'$ contains all the isolated roots of the input system of polynomials $f_1, \ldots, f_m$ in n variables.

Algorithm RUR_overconstrained

Input: $f_1, \ldots, f_m \in \mathbb{K}[X_1, \ldots, X_n]$.

Output: $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for some finite set $\overline{Z}'$ which contains all the isolated common roots of the input system as well as at least one point from every irreducible component of the zero set of the input system.

1: for $i := 1, \ldots, m$ do:

2: $g_i(X_1, \ldots, X_m) \leftarrow f_i(X_1, \ldots, X_n) \cdot (X_{n+1} - a_{n+1}) \cdot \cdots \cdot (X_m - a_m)$ where $a_{n+1}, \ldots, a_m$ are some constants in $\mathbb{K}$.

3: call Algorithm **RUR_square** on the input $g_1, \ldots, g_m \in \mathbb{K}[X_1, \ldots, X_m]$ to compute $h, h_1, \ldots, h_m \in \mathbb{K}[T]$ forming the RUR for some finite set $\overline{Y}'$ which contains at least one point from every irreducible component of the zero set (in $\overline{\mathbb{K}}^m$) of the system of polynomials $g_1, \ldots, g_m$

4: discard $h_{n+1}, \ldots, h_m$ to obtain $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for $\overline{Z}'$

One may suspect that the RUR for (some finite subset of) $\mathbf{Z}^{(m)}\left(f_1, \ldots, f_m\right)$ may be computed via Algorithm **RUR_square** by simply treating polynomials $f_1, \ldots, f_m$ as polynomials in m variables $X_1, \ldots, X_m$ instead of generating $g_1, \ldots, g_m$. Unfortunately, this is not so. Recall that all irreducible components $W_1, \ldots, W_l$ of $\mathbf{Z}^{(m)}\left(f_1, \ldots, f_m\right)$ are of positive dimension. In order to compute the RUR for (some finite subset of) the zero set of positive dimension, the input system of polynomials $f_1, \ldots, f_m$ must be perturbed by auxiliary polynomials $f_1^*, \ldots, f_m^*$ with the conditions (1) the support of f_i^* is contained in the support of f_i for $i = 1, \ldots, m$,

and (2) $f_1^*, \ldots, f_m^*$ have only finitely many common roots in $\overline{\mathbb{K}}^m$. (See step **9** of Algorithm **RUR_toric_square**.) Suppose condition (1) is satisfied. Since variables $X_{n+1}, \ldots, X_m$ do not appear in the supports of $f_1, \ldots, f_m$, they are not in the supports of $f_1^*, \ldots, f_m^*$. Then, the zero set of $f_1^*, \ldots, f_m^*$ never becomes finite; it always contains $\overline{\mathbb{K}}^{m-n}$. Thus, there is no system of polynomials $f_1^*, \ldots, f_m^*$ satisfying the above two conditions simultaneously, and step **9** of Algorithm **RUR_toric_square** always fails.

3.2.3. RUR for underdetermined systems

Consider a system of m polynomials $f_1, \ldots, f_m$ in n variables with coefficients in $\mathbb{K}$. Assume that $m < n$. Then, we can construct a square system by adding $n - m$ copies of f_m to the input system and use Algorithm **RUR_square**.

In general, the zero set of an underdetermined system has some positive-dimensional components. Our algorithm cannot find them, but just picks up finitely many points on them. This is not very interesting, as the result is nearly meaningless in any real application.

3.2.4. Algorithm RUR

Putting the results from the previous sections together, given a system of m polynomials $f_1, \ldots, f_m$ in n variables with coefficients in $\mathbb{K}$, even though $m \neq n$, we can compute the RUR for some set $\overline{Z}'$ which contains all the isolated common roots of the input system in $\overline{\mathbb{K}}^n$ (rather than in $(\overline{\mathbb{K}}^*)^n$) as well as at least one point from every irreducible component of the zero set of the input system.

Algorithm RUR

Input: $f_1, \ldots, f_m \in \mathbb{K}[X_1, \ldots, X_n]$.

Output: $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for some finite set $\overline{Z}'$ which contains all the isolated common roots of the input system as well as at least one point from every irreducible component of the zero set of the input system.

1: if $m > n$ then
2: call Algorithm **RUR_overconstrained** to compute $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for $\overline{Z}'$
3: else if $m = n$ then
4: call Algorithm **RUR_square** to compute $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for $\overline{Z}'$
5: else /* if $m < n$ then */
6: $g_1 \leftarrow f_1, \ \ldots, \ g_m \leftarrow f_m, \ g_{m+1} \leftarrow f_m, \ \ldots, \ g_n \leftarrow f_m$
7: call Algorithm **RUR_square** on the input $g_1, \ldots, g_n$ to compute $h, h_1, \ldots, h_n \in \mathbb{K}[T]$ forming the RUR for $\overline{Z}'$

3.3. Real solving via RUR

Consider a square system of polynomials $f_1, \ldots, f_n$ in n variables with coefficients in $\mathbb{Q}$. Assume that $\mathrm{MV}_{-0} > 0$. We have seen that we are able to compute the RUR for some set $\overline{Z}' \subseteq (\overline{\mathbb{C}}^*)^n$. The value of the ith coordinate of a point in $\overline{Z}'$ can be obtained by evaluating the univariate polynomial h_i with coefficients in $\mathbb{Q}$ at some root θ of the univariate polynomial h with coefficients h. If $\theta \in \mathbb{R}$ then, obviously, $h_i(\theta) \in \mathbb{R}$. In this section, we will show that, under

a certain condition, the converse is also true. We would like to point out that the converse is not a direct consequence of the existence of the RUR, but instead follows from the fact that a certain type of random choices can always be made in algorithm **RUR_square**.

Let the RUR for the zero set of a given system of polynomials with rational coefficients be

$$\left\{(h_1(\theta), \ldots, h_n(\theta)) \in \overline{\mathbb{K}}^n \mid \theta \in \overline{\mathbb{K}} \text{ with } h(\theta) = 0\right\}.$$

In order to approximate all the coordinates of all the roots of the input system in the RUR to any given precision, we should be able to approximate all the roots of h to any given precision simultaneously. We will show that the point $(h_1(\theta), \ldots, h_n(\theta))$ is real iff θ is real. Thus, in order to approximate all the coordinates of all the real roots of the input system in the RUR to any given precision, we should be able to approximate all the real roots of h simultaneously, but this can be done by using, e.g., Sturm's method. On the other hand, it is not trivial to enumerate all the roots of h.

Consider a square system of n polynomials $f_1, \ldots, f_n$ in n variables with rational coefficients. Let Z be the zero set of the input system (in $\mathbb{C}^n$). Suppose that Algorithm **RUR_square** is called on the input $f_1, \ldots, f_n$ and returns the univariate polynomials h and $h_1, \ldots, h_n$ with rational coefficients forming the RUR for some finite set $\overline{Z}'$. The set $\overline{Z}'$ contains all the isolated common roots of the input system.

Proposition 14. *For any root θ of h,*

$$\theta \in \mathbb{R} \Leftrightarrow (h_1(\theta), \ldots, h_n(\theta)) \in \mathbb{R}^n. \tag{32}$$

Proof. The sufficient condition is obvious. Thus, we only need to show the necessary condition.

Rewriting (19), $\theta = -\sum_{l=1}^{n} u_l \cdot h_l(\theta)$. By Remark 10, $u_1, \ldots, u_l \in \mathbb{Z}$. Hence, $h_1(\theta), \ldots, h_n(\theta) \in \mathbb{R} \Rightarrow \theta \in \mathbb{R}$. □

If Z is of dimension zero then all the common roots of the input system are isolated, and thus, $\overline{Z}' \supseteq Z$. Hence, the set

$$\overline{Z}'_{\mathbb{R}} = \{(h_1(\theta), \ldots, h_n(\theta)) \mid \theta \in \mathbb{R} \text{ with } h(\theta) = 0\} \subseteq \mathbb{R}^n$$

contains all the real roots of the input system. Therefore, our algorithm can be used for the real solving of zero-dimensional square systems.

On the other hand, if Z is of positive dimension then there may be some real roots of the input system which are not contained in $\overline{Z}'_{\mathbb{R}}$. Algorithm **RUR_square** picks up least one point from each of the positive-dimensional components of Z. These positive-dimensional components of Z contain finitely or infinitely many real points. However, there is no guarantee that the points picked up by the algorithm are real, even if there are only finitely many real roots on some positive-dimensional components. Note that Proposition 14 still holds, though it is not useful in this case.

4. Examples

In this section, we give examples that illustrate the results and shortcomings of our algorithms. Due to the limited space available to display results, all the examples listed are of low dimension.

Example F_1:

Consider a system F_1 of 2 polynomials in 2 variables with integer coefficients:

$$\begin{aligned} f_1 &= 1 + 2X - 2X^2Y - 5XY + X^2 + 3X^3Y, \\ f_2 &= 2 + 6X - 6X^2Y - 11XY + 4X^2 + 5X^3Y. \end{aligned} \tag{33}$$

The zero set of F_1 consists of 2 isolated points $(1, 1)$, $\left(\frac{1}{7}, \frac{7}{4}\right)$ and 1 irreducible component $X = -1$ of dimension 1.

The RUR for the zero set of F_1 is computed as follows:

$$\begin{aligned} h(T) &= 84T^4 + 306T^3 - 574T^2 - 1545T + 1989, \\ h_1(T) &= -T - \frac{r_{1,1}(T)}{r_{1,0}(T)}, \\ h_2(T) &= -T - \frac{r_{2,1}(T)}{r_{2,0}(T)} \end{aligned} \tag{34}$$

where

$$\begin{aligned} r_{1,1}(T) &= -13822794943768419 84T^3 - 5914280670220800T^2 \\ &\quad + 94587294494414113 92T - 8992070973148449600, \\ r_{1,0}(T) &= -3963147148947893 76T^3 - 1262397960878976T^2 \\ &\quad + 27177582113166804 48T - 2585831836226329728, \\ r_{2,1}(T) &= -\frac{5412043022435784482792945336 32}{3176523}T^3 \\ &\quad - \frac{3120374299759092128055296}{27}T^2 \\ &\quad + \frac{1069598558052109058795873435648}{1058841}T \\ &\quad - \frac{26253101608553522011667959808 0}{352947}, \\ r_{2,0}(T) &= -\frac{9874040237517291328323911680}{151263}T^3 \\ &\quad + \frac{1904111379091981805699072}{27}T^2 \\ &\quad + \frac{9016074028903097271713937817 6}{151263}T \\ &\quad - \frac{3610984918268547057817776947 2}{50421}. \end{aligned}$$

The univariate polynomial h has 4 roots θ and the values of the real and imaginary parts of $h_1(\theta)$ and $h_2(\theta)$ are approximated as follows:

(Re $h_1(\theta)$,	Im $h_1(\theta)$),	(Re $h_2(\theta)$,	Im $h_2(\theta)$)
$(-1,$	$-1.0561 \times 10^{-46})$,	$(-0.32019,$	$-1.6753 \times 10^{-47})$
$(1,$	$1.6658 \times 10^{-47})$,	$(1,$	$8.0503 \times 10^{-48})$
$(0.14286,$	$1.0519 \times 10^{-51})$,	$(1.75,$	$1.0512 \times 10^{-51})$
$(-1,$	$-5.2409 \times 10^{-43})$,	$(0.19519,$	$-3.5186 \times 10^{-40})$

The table above suggests that, for the RUR computed as (34), we find 2 isolated roots along with 2 points on the positive-dimensional component.

Example F_2:

Consider a system F_2 of 3 polynomials in 3 variables with integer coefficients:

$$\begin{aligned} f_1 &= X^2 + Y^2 + Z - 1, \\ f_2 &= X^2 + Y^2 - Z + 1, \\ f_3 &= Z - 1. \end{aligned} \tag{35}$$

It is easy to see that F_2 has one and only one real root $(0, 0, 1)$ which actually lies on the intersection of 2 complex positive-dimensional components $\left\{\left(-\sqrt{-1}Y, Y, 1\right)\right\}$ and $\left\{\left(\sqrt{-1}Y, Y, 1\right)\right\}$ of the zero set of the input system.

The univariate polynomial h in the RUR for the zero set of F_2 is computed as follows:

$$h = -T^4 - 4T^3 - 6T^2 - 4T - 5. \tag{36}$$

Since the degree of h is 4, the RUR for the set determines 4 points lying on the positive-dimensional components. By using Sturm's method, we can easily see that h has no real roots. Thus, by Proposition 14, none of those 4 points are real.

In general, if the zero set of the input system has some positive-dimensional components on which there are only finitely many real points then our algorithm often will not pick up (some or all of) these real points. See Section 3.3.

Example F_3:

Let L_3 be a system of 3 linear polynomials in 3 variables with integer coefficients:

$$\begin{aligned} l_1 &= 3X - Y - 1, \\ l_2 &= X - Y + 1, \\ l_3 &= X + Y - 3. \end{aligned} \tag{37}$$

The zero set of L_3 consists of a single point $(1, 2)$.

Now, consider a system F_3 of 3 polynomials in 2 variables with integer coefficients:

$$\begin{aligned} f_1 = l_2 \cdot l_3 &= X^2 - 2X - Y^2 + 4Y - 3, \\ f_2 = l_1 \cdot l_3 &= 3X^2 + 2XY - 10X - Y^2 + 2Y + 3, \\ f_3 = l_1 \cdot l_2 &= 3X^2 - 4XY + 2X + Y^2 - 1. \end{aligned} \tag{38}$$

By construction, we immediately see that the zero set of F_3 consists of a single point $(1, 2)$. Note, however, that the zero sets of any subsystem consisting of 2 polynomials has a positive-dimensional component. (The subsystems (f_1, f_2), (f_1, f_3) and (f_2, f_3) have positive-dimensional components l_3, l_2 and l_1, respectively.) Thus, an approach such as finding solutions for one pair of equations and then "checking" the third equation would not be sufficient.

Since F_3 is an overdetermined system, **Algorithm_RUR_overconstrained** constructs a square system G_3 of 3 polynomials in 3 variables with rational coefficients:

$$\begin{aligned} g_1 = f_1 \cdot (Z - 1) &= \left(X^2 - 2X - Y^2 + 4Y - 3\right)(Z - 1), \\ g_2 = f_2 \cdot (Z - 1) &= \left(3X^2 + 2XY - 10X - Y^2 + 2Y + 3\right)(Z - 1), \\ g_3 = f_3 \cdot (Z - 1) &= \left(3X^2 - 4XY + 2X + Y^2 - 1\right)(Z - 1). \end{aligned} \tag{39}$$

The univariate polynomial h in the RUR for the zero set of G_3 is computed as follows:

$$\begin{aligned} h \;=\; & -5505024000T^{10} - 391643136000T^9 \\ & - 9566787993600T^8 - 434913253785607^7 \\ & + 2475168513392640T^6 + 58123559884554240T^5 \\ & + 57118479152578560T^4 + 22768913951490048007^3 \\ & - 4405394155933532160T^2 - 704116623899885568007 \\ & - 176330303770208501760. \end{aligned} \tag{40}$$

Likewise, but not shown here, we compute h_1, h_2, and h_3. The RUR for the zero set of F_3 is obtained from the RUR of the zero set of G_3 by ignoring the last coordinate (i.e. ignoring h_3). Evaluating h_1 and h_2 at roots of h gives us multiple points at the single location $(1, 2)$.

5. Complexity analysis

In this section, we give a worst-case asymptotic complexity analysis of Algorithm **RUR_toric_square** described in Section 3.1.

The computational model used here is either the Turing machine or the BSS machine (Blum et al., 1997). If $\mathbb{K}$ is $\mathbb{Q}$ or some finite field then the algorithm can be implemented on Turing machines (or existing computers). On the other hand, if $\mathbb{K}$ is $\mathbb{R}$ or $\mathbb{C}$ then the algorithm cannot be implemented (exactly) on Turing machines. In this case, the BSS machine over $\mathbb{R}$ or $\mathbb{C}$ is used. On the BSS machine over a field $\mathbb{K}$, an arithmetic operation over $\mathbb{K}$ is done in constant time, and thus, roughly speaking, the time complexity of a given algorithm matches the number of arithmetic operations over $\mathbb{K}$. In order to make a valid argument on either of those computational models, in this section, we only consider the arithmetic complexity (the number of arithmetic operations) of the algorithm. The bit-length of the quantities appearing in the algorithm is not discussed here, but some discussion of the practical performance can be found in Section 6.1.2.

The following notations are used:

Let $\boldsymbol{O}^*(\,)$ denote a big oh notation in which a polylog factor is ignored: $\boldsymbol{O}^*(n) = \boldsymbol{O}\left(n \log^r n\right)$ for some $r \geq 0$. Also, let ω be the constant so that the matrix multiplication of two square matrices of dimension l takes $\boldsymbol{O}(l^\omega)$ arithmetic operations. It is well-known that $\omega < 2.376$.

5.1. Arithmetic complexity analysis

Consider a square system of polynomials $f_1, \ldots, f_n$ in n variables with coefficients in $\mathbb{K}$. Let A_i be the support of f_i for $i = 1, \ldots, n$. Suppose Algorithm **RUR_toric_square** is called on the input $f_1, \ldots, f_n$ and $A_1, \ldots, A_n$ and returns the RUR for some finite subset Z' of the zero set Z of the input system. The algorithm sets the support A_0 of f_0 so that f_0 is a linear polynomial.

We introduce two quantities $\mathcal{M}$ and $\mathcal{N}$.

As before, let MV_{-i} denote the mixed-volume of the convex hulls of $A_0, A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n$. (See Section 2.2.2.) Define $\mathcal{M} = \sum_{i=0}^{n} \mathrm{MV}_{-i}$. Thus, $\mathcal{M}$ is the degree of the toric resultant.

Step **7** of the algorithm constructs the toric resultant matrix N whose determinant is some non-trivial multiple of the toric resultant. Let $\mathcal{N} = \dim N$.

Step **18** or step **26** of the algorithm determines the cardinality M of Z' which matches the degree of the univariate polynomial h in the RUR.

Recall the following facts (von zur Gathen and Gerhard, 2003):

- Given $l+1$ distinct values in $\mathbb{K}$, a unique univariate polynomial of degree at most l with coefficients in $\mathbb{K}$ that takes those values at $l+1$ distinct points in $\mathbb{K}$ can be computed via interpolation using $\boldsymbol{O}^*(l)$ arithmetic operations over $\mathbb{K}$.
- Given two univariate polynomials with coefficients in $\mathbb{K}$ of degree at most l, their GCD is computed using $\boldsymbol{O}^*(l)$ arithmetic operations over $\mathbb{K}$. Thus, given a univariate polynomial with coefficients in $\mathbb{K}$ of degree at most l, its square-free part is computed using $\boldsymbol{O}^*(l)$ arithmetic operations over $\mathbb{K}$.

The value of the toric resultant of a system of $n+1$ polynomials in n variables with supports $A_0, A_1, \ldots, A_n$ with arbitrary but fixed coefficients in $\mathbb{K}$ is calculated using $\boldsymbol{O}^*(n\mathcal{M}\mathcal{N}^\omega)$ arithmetic operations over $\mathbb{K}$ (Emiris and Canny, 1995; Canny and Emiris, 2000).

The arithmetic complexity of the algorithm is governed by the loop from step **13** through step **25** or the loop from step **26** through **36**.

By (5), $\deg_s \mathrm{TGCP}(s, \boldsymbol{u}) = \sum_{i=1}^{n} \mathrm{MV}_{-i} = \mathcal{M} - \mathrm{MV}_{-0}$, and thus, the coefficients of $\mathrm{TGCP}(s, \boldsymbol{u})$ at fixed $\boldsymbol{u}$ (regarded as a univariate polynomial in s) are computed via interpolation using $\boldsymbol{O}^*\left(n\mathcal{M}^2\mathcal{N}^\omega\right)$ arithmetic operations over $\mathbb{K}$.

By (4), $\deg_T \mathrm{TPert}(T, u_1, \ldots, u_n) = \mathrm{MV}_{-0}$. Thus, at step **15** or step **28**, $\mathrm{TPert}(T, u_1, \ldots, u_n)$ at fixed $(u_1, \ldots, u_n)$ is computed via interpolation using $\boldsymbol{O}^*\left(n\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^\omega\right)$ arithmetic operations over $\mathbb{K}$, and at step **16** or step **29**, the square-free part of $\mathrm{TPert}(T, u_1, \ldots, u_n)$ is computed using $\boldsymbol{O}^*(\mathrm{MV}_{-0})$ arithmetic operations over $\mathbb{K}$.

By the same argument as in the previous paragraph, for $i = 1, \ldots, n$, at step **33**, $\mathrm{TPert}(t, u_1, \ldots, u_{i-1}, u_i \pm 1, u_{i+1}, \ldots, u_n)$ at fixed $(u_1, \ldots, u_n)$ are computed using $\boldsymbol{O}^*\left(n\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^\omega\right)$ arithmetic operations over $\mathbb{K}$, and at step **34**, $q_i^{\pm}(t)$ are computed using $\boldsymbol{O}^*(\mathrm{MV}_{-0})$ arithmetic operations over $\mathbb{K}$. Hence, the **for** loop from step **33** through step **36** is executed using $\boldsymbol{O}^*\left(n^2\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^\omega\right)$ arithmetic operations over $\mathbb{K}$.

By Proposition 8, the loop from step **13** through step **25** is repeated $\boldsymbol{O}\left(n\mathrm{MV}_{-0}^2\right)$ times, and each iteration uses $\boldsymbol{O}^*\left(n\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^\omega\right)$ arithmetic operations over $\mathbb{K}$. Thus, in total, the number of arithmetic operations over $\mathbb{K}$ needed to process this loop is $\boldsymbol{O}^*\left(n^2\mathrm{MV}_{-0}^3\mathcal{M}^2\mathcal{N}^\omega\right)$.

By Proposition 9, the loop from step **26** through step **36** is repeated $\boldsymbol{O}\left(n^2M^2\right)$ times, and each iteration uses $\boldsymbol{O}^*\left(n^2\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^\omega\right)$ arithmetic operations over $\mathbb{K}$. Thus, in total, the number of arithmetic operations over $\mathbb{K}$ needed to process this loop is $\boldsymbol{O}^*\left(n^4M^2\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^\omega\right)$.

Putting all of this together, the univariate polynomials h and $h_1, \ldots, h_n$ forming the RUR are computed using $\boldsymbol{O}^*\left(n^2\left(\mathrm{MV}_{-0}^2 + n^2M^2\right)\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^\omega\right)$ arithmetic operations over $\mathbb{K}$.

By (9), $\mathrm{MV}_{-0} \geq M$. The equality holds if Z' does not contain any multiple root of the input system. In this case, the loop from step **26** through step **36** governs the complexity of the algorithms. On the other hand, if Z' contains some multiple roots of the input system then $\mathrm{TPert}(T, u_1, \ldots, u_n)$ is not square-free and

$$\deg_T \mathrm{TPert}(T, u_1, \ldots, u_n) = \mathrm{MV}_{-0} > M = \deg h. \tag{41}$$

In this case, there is a slight chance that the loop from step **13** through **25** takes more arithmetic operations over $\mathbb{K}$ than the loop from step **26** through step **36**. When the loop from step **13** through **25** is executed, M has not yet been determined. On the other hand, the loop from step **26** through step **36** is executed after M is correctly determined. Thus, the loop from step **26** through step **36** does not have to be repeated unnecessarily.

5.1.1. $\mathcal{M}$ and $\mathcal{N}$

We have seen that the arithmetic complexity of the algorithm is expressed in terms of $\mathcal{M}$ and $\mathcal{N}$. In this section, we consider how the relation of those two quantities affects the complexity of the algorithm.

When the characteristic of $\mathbb{K}$ is 0, at step **15** or step **28** or step **33**, TPert $(T, u_1, \ldots, u_n)$ at fixed $(u_1, \ldots, u_n)$ (regarded as a polynomial in $\mathbb{K}[T]$) is computed via interpolation through the values of the coefficient of the term s^d in some non-trivial multiple of TGCP $(s, \boldsymbol{u})$ instead of the values of the coefficient of the term s^d in TGCP $(s, \boldsymbol{u})$ without any extraneous factor. The value of any coefficient of some non-trivial multiple of TGCP $(s, \boldsymbol{u})$ is interpolated from the values of $\det N$ and N has been constructed at step **7**. On the other hand, the value of any coefficient of TGCP $(s, \boldsymbol{u})$ without any extraneous factor is interpolated from the values of TRes $\left(f_0, f_1 - sf_1^*, \ldots, f_n - sf_n^*\right)$, but evaluation of TRes $\left(f_0, f_1 - sf_1^*, \ldots, f_n - sf_n^*\right)$ requires additional steps to eliminate the contribution of the extraneous factor from the value of $\det N$. The use of $\det N$ instead of TRes $\left(f_0, f_1 - sf_1^*, \ldots, f_n - sf_n^*\right)$ allows us to avoid executing these additional steps. (See Section 3.1.) In such a case, TPert $(T, u_1, \ldots .u_n)$ at fixed $(u_1, \ldots, u_n)$ is computed using $\boldsymbol{O}^*\left(\mathrm{MV}_{-0}\mathcal{N}^{1+\omega}\right)$ arithmetic operations over $\mathbb{K}$, and thus, the number of arithmetic operations over $\mathbb{K}$ needed to compute univariate polynomials h and $h_1, \ldots, h_n$ forming the RUR becomes $\boldsymbol{O}^*\left(n\left(\mathrm{MV}_{-0}^2 + n^2M^2\right)\mathrm{MV}_{-0}\mathcal{N}^{1+\omega}\right)$.

The quantity $\mathcal{N}$ actually depends on the algorithm used to construct the resultant matrix N. From (2) and (3), $\mathcal{N} \geq \mathcal{M}$. However, no algorithm that constructs an optimal N (i.e., N satisfying $\mathcal{N} = \mathcal{M}$) has been found except for very small n (Khetan, 2003, 2005). Even if the best algorithm currently known is used, there is a risk that $\mathcal{N}$ becomes exponentially bigger than $\mathcal{M}$ (Emiris and Canny, 1995; Canny and Emiris, 2000): $\mathcal{N} = \boldsymbol{O}\left(\frac{e^n}{\sqrt{n}}\mathcal{M}\right)$. Thus, asymptotically, the cost of additional steps to eliminate the extraneous factor will be negligible compared to the cost of interpolations through the values of the determinant of a bigger matrix. Hence, even if the characteristic of $\mathbb{K}$ is 0, the "best" worst-case arithmetic complexity of Algorithm **RUR_toric_square** remains $\boldsymbol{O}^*\left(n^2\left(\mathrm{MV}_{-0}^2 + n^2M^2\right)\mathrm{MV}_{-0}\mathcal{M}^2\mathcal{N}^{\omega}\right)$. In practice, $\mathcal{N}$ rarely becomes exponentially bigger than $\mathcal{M}$. This matter is discussed more later in Section 6. Nevertheless, developing an algorithm for computing a resultant matrix of smaller (or the smallest) size is still an active area of research.

6. Implementation

In this section, we describe implementations of the algorithms.

Our goal is to develop a library for computing the RUR for the zero set of a system of m polynomials with rational coefficients such that (1) the exact RUR is computed, meaning that all the rational coefficients of the univariate polynomials forming the RUR will be computed to full precision, and (2) for small m, the library runs in an acceptable amount of time in practice.

Since all the algorithms reduce to Algorithm **RUR_toric_square** (see Fig. 1), we mainly discuss the implementation of Algorithm **RUR_toric_square**.

We also show some experimental results.

6.1. Implementation of **RUR_toric_square**

We discuss here an implementation of Algorithm **RUR_toric_square** for computing the exact RUR for the zero set of a square system of n polynomials $f_1, \ldots, f_n$ in n variables with rational

coefficients. Let A_i be the support of f_i for $i = 1, \ldots, n$. The algorithm sets, at step **1**, the support A_0 of f_0 so that f_0 is a linear polynomial.

Step **7** constructs the toric resultant matrix N of a system of $n+1$ polynomials in n variables with supports $A_0, A_1, \ldots, A_n$. We implement Emiris's incremental algorithm (Emiris and Canny, 1995). This algorithm computes, as byproducts, the convex hulls Q_i of A_i and the quantities MV_{-i} for $i = 0, 1, \ldots, n$ where MV_{-i} is the mixed-volume of $Q_0, Q_1, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_n$. The computation of Q_i and the computation of MV_{-i} both reduce to some linear programming problems (Emiris and Canny, 1995; Canny and Emiris, 2000) where all the linear constraints have rational coefficients. These linear programming problems are solved via a standard two-phase simplex method that is implemented with multi-precision rational number arithmetic in order to help deal with instability issues. Further discussion of alternative toric resultant implementations is given below in Section 6.1.1.

Note that, for the resultant matrix constructed by Emiris's incremental algorithm, equality (2) holds (Emiris and Canny, 1995). Thus, equality (4) also holds, on which the correctness of Algorithm **RUR_toric_square** relies.

Recall that, in Section 5.1.1, when the characteristic of $\mathbb{K}$ is 0, in particular, $\mathbb{K} = \mathbb{Q}$, there are two options for determining d at step **10** and computing $\mathrm{TPert}(T, u_1, \ldots, u_n)$ at fixed $(u_1, \ldots, u_n)$ at step **15**, step **28** and step **33**. We implement a version in which d at step **10** is determined by scanning the coefficients of some non-trivial multiples of $\mathrm{TGCP}(s, 1, 0, \ldots, 0)$ instead of the coefficients of $\mathrm{TGCP}(s, 1, 0, \ldots, 0)$ without any extraneous factor. Also, we compute $\mathrm{TPert}(T, u_1, \ldots, u_n)$ via interpolation through the values of the coefficient of the term s^d in some non-trivial multiple of $\mathrm{TGCP}(s, \boldsymbol{u})$ instead of the values of the coefficient of the term s^d in $\mathrm{TGCP}(s, \boldsymbol{u})$ without any extraneous factor. That is, we do not eliminate the contribution of the extraneous factor from $\det N$ before interpolating some non-trivial multiple of $\mathrm{TGCP}(s, \boldsymbol{u})$. (See Section 3.1.) In Section 5.1.1, we have seen that, asymptotically, the cost of eliminating the extraneous factor is negligible compared to the cost of interpolations through the values of the determinant of a bigger resultant matrix. However, this is not true in practice. The resultant matrix constructed by Emiris's incremental algorithm is generically not too big (Emiris and Canny, 1995). In particular, for small n, we usually gain significant speedup by avoiding the costly process of elimination of the extraneous factors, even though the cost of interpolations slightly increases.

Whenever $\det N$ is evaluated, the non-zero entries of N are specialized to the coefficients of the linear u polynomial $f_0 = u_0 + u_1 X_1 + \cdots + u_n X_n$ and polynomials in the perturbed system $f_1 - s f_1^*, \ldots, f_n - s f_n^*$. By Proposition 6 and Remark 10, parameters $u_0, u_1, \ldots, u_n$ are always specialized to some integers. The coefficients of the input polynomials $f_1, \ldots, f_n$ are rational numbers. Since the characteristic of $\mathbb{Q}$ is 0, the coefficients of auxiliary polynomials $f_1^*, \ldots, f_n^*$ can be chosen from rational numbers at step **9**, and at any interpolation, we can assign rational values to s. Thus, the entries of N are always specialized to rational numbers. Hence, all the coefficients of some non-trivial multiple of $\mathrm{TGCP}(s, \boldsymbol{u})$ are rational numbers. It immediately follows that all the coefficients of $\mathrm{TPert}(T, u_1, \ldots, u_n)$ are rational numbers and they can be computed to full precision.

The rest of the algorithm involves arithmetic operations and the Euclidean algorithm over the ring of univariate polynomials with rational coefficients, and the computation of the first subresultant of two univariate polynomials. Therefore, by the use of multi-precision rational number arithmetic, all the steps of Algorithm **RUR_toric_square** can be implemented exactly and the exact RUR will be computed.

6.1.1. Alternative toric resultant computations

In order to have a practically efficient implementation, it is important to choose a fast algorithm that constructs resultant matrices of reasonable size. There are several algorithms for computing the toric resultant of a square system of polynomials (Canny and Emiris, 2000; Emiris and Canny, 1995; D'Andrea, 2002; Emiris, 2003; Khetan, 2003). While we implement Emiris's incremental algorithm, the other algorithms can be used if the prerequisite conditions are met.

D'Andrea's formula (D'Andrea, 2002) computes the toric resultant as a quotient of two determinants. The matrix whose determinant becomes the numerator is as big as the resultant matrix constructed by Emiris's algorithms. Thus, evaluating the toric resultant using this formula costs at least as much as evaluating the determinant of the toric resultant matrix (with some extraneous factor) constructed by Emiris's algorithms.

Khetan's formula (Khetan, 2003, 2005) computes the toric resultant as the determinant of a single matrix. Formulas have been found for unmixed systems of 3 polynomials in 2 variables (Khetan, 2003) and 4 polynomials in 3 variables (Khetan, 2005), but it is probably impossible to find such formulas for general systems of $n+1$ polynomials in n variables. If we apply the formula to a mixed system with supports $A_0, A_1, \ldots, A_n$ then we must treat the input system as unmixed by pretending that all the input polynomials have the identical support $\bigcup_{i=0}^{n} A_i$. The degree of the toric resultant of this "fake" unmixed system could be much larger than that of the original system. Also, the resultant matrix contains a block whose entries themselves are the determinants of some other matrices. Thus, the cost of evaluating the resultant matrix constructed using Khetan's formula is more than the cost of evaluating the optimal resultant matrix.

Besides Emiris's incremental algorithm, we could instead use his mixed-subdivision-based algorithm (Canny and Emiris, 2000). The mixed-subdivision-based algorithm constructs a single resultant matrix that works, but the size of the resultant matrix constructed is often much larger than the optimal one. In fact, the difference might become exponential in n (Canny and Emiris, 2000; Emiris and Canny, 1995). On the other hand, the incremental algorithm (Emiris and Canny, 1995) tries several matrices. Starting at a matrix of the smallest possible size, the algorithm keeps enlarging matrices until one that works is found. If none of these trials are successful, the incremental algorithm constructs the same resultant matrix as the mixed-subdivision-based algorithm does. Thus, in the worst case, the incremental algorithm requires much more computation and still ends up returning a big matrix. However, we observe in practice that the incremental algorithm usually constructs a resultant matrix of reasonable size within only a few iterations.

In terms of the arithmetic complexity, the argument above is rephrased as follows: letting $\mathcal{N}_S$ and $\mathcal{N}_I$ be the size of the resultant matrices constructed by the mixed-subdivision-based algorithm and the incremental algorithm, respectively, the arithmetic complexity for these algorithms is $\boldsymbol{O}^*\left(\mathcal{N}_S^{\omega}\right)$ and $\boldsymbol{O}^*\left(\mathcal{N}_I^{1+\omega}\right)$, respectively. In the worst case, $\mathcal{N}_I = \mathcal{N}_S$, however, usually, $\mathcal{N}_I$ is much smaller than $\mathcal{N}_S$.

Note that, for both versions, the number of rows of the resultant matrix whose entries are specialized to the coefficients of f_0 is fixed to MV_{-0}. Thus, equality (2) holds (Canny and Emiris, 2000; Emiris and Canny, 1995).

The SYNAPS library[6] provides an alternative implementation of Emiris's toric resultant algorithm that is nearly identical to ours, with some small differences. SYNAPS computes the mixed-volume of the convex hulls of given sets of points (steps **2** and **7** in Algorithm

[6] http://www-sop.inria.fr/galaad/software/synaps/.

RUR_toric_square). In contrast to the SYNAPS implementation, which uses floating point numbers to implement the simplex method for solving linear programming problems, our implementation uses multi-precision arithmetic in order to avoid problems due to numerical inaccuracies. SYNAPS also provides the algorithm for computing the stable mixed-volume (Huber and Sturmfels, 1997).

SYNAPS provides an implementation for computing the toric resultant of a given system of $n+1$ polynomials in n variables using a generic programming formulation that can be instantiated over the field to which the coefficients of the input polynomials belong. Our implementation is more limited, assuming that the coefficients of the input polynomials are rational numbers. Also, in contrast to SYNAPSE (but for greater efficiency) we implement only a portion of the algorithm (step **7** in Algorithm **RUR_toric_square**); we stop once the resultant matrix is constructed. The determinant of the matrix is some multiple of the resultant, but the extraneous factor will be canceled out later.

6.1.2. Expression swell

The algorithms given suffer from expression swell, thus slowing the performance. While large input coefficients are a concern, even if the coefficients of the input polynomials are small, the intermediate and final quantities can grow quite large.

Algorithm **RUR_toric_square** consists of exact evaluation of the determinant of a given square matrix with rational entries, polynomial interpolations over rational numbers and operations over the ring of univariate polynomials with rational coefficients. We can thus use modular arithmetic in order to avoid expression swell occurring in exact evaluation of determinants.

At step **10**, step **15**, step **28** and step **33**, we evaluate the determinant of the toric resultant matrix N whose entries are specialized in several ways, while, at step **39**, we evaluate the determinant of the first subresultant matrices of size $2M-1$ where $M = \deg h$. We have seen that $\dim N > \mathrm{MV}_{-0} \geq M$, but $\dim N$ is not necessarily larger than $2M-1$. The size of the entries of N depends on the input and could be large or small, while the entries of the first subresultant matrices are usually large because of expression swell of intermediate quantities. Thus, we always use modular arithmetic to compute the first subresultants, while modular arithmetic is used to evaluate $\det N$ only when $\dim N$ is large and/or the size of the entries of N is large. For more about exact evaluation of determinants, see Emiris (1998) and Kaltofen and Villard (2004).

Modular arithmetic can also be used for interpolations and operations over the polynomial ring. See von zur Gathen and Gerhard (2003).

Recall that, at step **40**, $h_i(T) = -T - r_{i,1}(T) \cdot r_{i,0}(T)^{-1} \bmod h(T)$ where $r_{i,0}(T) + r_{i,1}(T)\, t$ is the first subresultant of $q_i^-(t)$ and $q_i^+(2T-t)$. Given $r_{i,0}(T)$, its inverse modulo h is computed using the extended Euclidean algorithm, which usually causes significant expression swell. In order to avoid this problem, we instead could compute $h_i(T)$ as a rational representation: $h_i(T) = -T - \frac{r_{i,1}(T)}{r_{i,0}(T)} \bmod h(T)$. In most applications, rational representations with significantly smaller coefficients are preferable to polynomials with large coefficients.

6.2. Experiments

We have implemented Algorithm **RUR** exactly. In this section, we show some experimental results of our implementation. The implementation is compiled with GNU C++. The GNU Multi-Precision (GMP) arithmetic library is used to support multi-precision rational number arithmetic.

Table 1
Timing breakdown for several examples $F_1, \ldots, F_6$

Input System	F_1	F_2	F_3	F_4	F_5	F_6
# of polynomials	2	3	3	2	2	2
# of variables	2	3	2	2	2	2
max. degree of monomials	4	2	2	2	2	2
max. bit-length of coefficients	4	1	4	307	51	95
# of roots of system	∞	∞	∞	2	∞	4
MV_{-0}	4	4	12	2	4	4
$\mathcal{M} = \sum_{i=0}^{n} \mathrm{MV}_{-i}$	12	12	36	5	8	8
$\mathcal{N} = \dim N$	12	17	42	6	10	10
$M =$ # of roots of h	4	4	10	2	4	4
max. bit-length of coefficients of h	18	20	68	844	159	292
max. bit-length of coefficients of $r_{i,j}$	173	96	1549	1686	2061	2820
max. bit-length of coefficients of h_i	44	11	273	358	8563	416
total time (sec)	.333	1.87	111	.0662	.672	.628
computing resultant matrix (%)	27	41	0	70	10	11
computing h (%)	14	13	9	2	4	1
computing $q_i^{\pm}$ (%)	48	44	51	6	13	3
computing h_i (%)	11	2	40	22	73	86

Rows 2 through 6 characterize the input systems. Rows 7 through 10 characterize the complexity of the toric resultant algorithm (See Section 5). N is the toric resultant matrix computed by the incremental algorithm. Rows 11 through 13 characterize the outputs. Rows 14 through 18 show the timing. Row 15 shows the percentage of computing the resultant matrix (Steps **1** through **7** in Algorithm **RUR_toric_square**). Row 16 shows the percentage of computing h (Steps **8** through **31** in Algorithm **RUR_toric_square**). Row 17 shows the percentage of computing $q_i^{\pm}$ (Steps **32** through **36** in Algorithm **RUR_toric_square**). Row 18 shows the percentage of computing h_i (Steps **38** through **40** in Algorithm **RUR_toric_square**).

All the experiments shown in this section are performed on a 3 GHz Intel Pentium CPU with 6 GB memory using Linux Kernel 2.6.

In Table 1, we show timing breakdowns for the application of the exact RUR to a few sample systems. We give a brief discussion of each case, and summarize the results.

Systems F_1 through F_3 are all drawn from examples described in Section 4, while systems F_4 through F_6 are all drawn from cases encountered in an actual geometric boundary evaluation computation. The source data is real-world data provided by the BRL-CAD (Dykstra and Muuss, 1989) solid modeling system.

For the overconstrained system F_3 we present the results for **RUR_toric_square** applied to the modified system G_3, as discussed in Section 4.

System F_4 consists of an intersection of a line with an ellipse. There are 2 intersections and both are real.

System F_5 consists of two ellipses. Rather than real intersections, these ellipses have 2 complex intersections.

System F_6 consists of two ellipses with supports

$$(2, 0)\,,\ (1, 0)\,,\ (0, 2)\,,\ (0, 1)\,,\ (0, 0)$$

and

$$(2, 0)\,,\ (1, 1)\,,\ (1, 0)\,,\ (0, 2)\,,\ (0, 1)\,,\ (0, 0)\,,$$

respectively. System F_6 has 4 roots and all of them are real.

For this example, we spent the most time computing polynomials h_1 and h_2. This was because the coefficients of the polynomial h, h_1 and h_2 become huge.

6.2.1. Summary of timing breakdowns

While the examples shown above are not comprehensive, from these and other cases we have examined, we can draw the following conclusions:

- The performance of our algorithm is reasonable for lower dimension/degree systems. However, for higher dimension/degree systems, the implementation tends not to be very practical.
- Constructing the toric resultant matrix takes up an insignificant portion of the time. Evidently, the use of Emiris's incremental algorithm rather than the mixed-subdivision-based algorithm is justified.
- For lower dimension/degree systems, the most time consuming part of the algorithm is repeated evaluation of the determinant of the toric resultant matrix. By the use of the incremental algorithm, we are able to construct resultant matrices of reasonably small size and sometimes even the optimal one (e.g. F_1). However, the size of the toric resultant matrix grows quite rapidly with respect to the dimension/degree of the input system.
- For positive-dimensional systems, the resultant evaluation contributes a certain amount to the total time, while for the zero-dimensional systems, the resultant evaluation is insignificant.
- For higher dimension/degree systems, the most time consuming part is computing univariate polynomials forming the exact RUR, mainly because of their huge coefficients. Further optimization such as extending the use of modular arithmetic (already used in determinant computation) to polynomial operations should be a target of future speedup efforts.
- For these examples, except for F_5, we did not find any benefit in using rational representations for h_i instead of univariate polynomials. For higher dimension/degree systems, though, significant improvement might be seen.

7. Conclusion

We have given a detailed description of algorithms for computing the RUR for the zero set of a given system of multivariate polynomial equations. We briefly summarize our major results here:

- Our algorithm for computing the RUR for the zero set of a square system improves on the algorithm originally introduced in Rojas (1999a). In both algorithms, the univariate polynomial h in the RUR is derived from the toric perturbation, which is a generalization of the toric u-resultant, by specializing the indeterminates to some appropriate values. We have described a deterministic way to specialize those indeterminates appropriately. Our algorithm correctly counts the number of roots of a given zero-dimensional system without multiplicity.
- We have developed a new algorithm for computing the RUR for the zero set of an overdetermined system. We construct a square system of higher dimension so that the projection of the RUR for the zero set of the square system will become the RUR for the zero set of the input system. In contrast to the algorithm for an overdetermined system described in Rojas (2000), where a square system of the same dimension is constructed from the input system with some random choices, our algorithm is deterministic. For small dimension, a single execution of our algorithm usually takes less time than several executions of the randomized algorithm.

- As a consequence of derandomization, we have developed a simple algorithm for computing real roots of a given system of multivariate polynomial equations.
- We have analyzed the arithmetic complexity of our algorithm. Since the derandomized algorithm correctly counts the number of distinct roots of the input system, we are able to give a tighter bound on the arithmetic complexity of the algorithm. Also, we have observed that the size of the resultant matrices governs the complexity of the algorithm.
- We have described an implementation of the algorithms for the case when all the coefficients of the input polynomials are rational numbers. The implementation is optimized for relatively small n. The implementation is exact; all the rational coefficients of the univariate polynomials forming the RUR will be computed to full precision. We have shown some experimental results. Tested systems include examples of a degenerate system and an overdetermined system as well as some problems picked from real-world industry.

7.1. Future work

There are several avenues of future work open, some of which we list here.

Our algorithm computes the RUR for some finite set that contains all the isolated roots of the input system as well as at least one point from each irreducible component of the zero set of the system. Under certain circumstances, this finite set may be redundant — it may contain some points that are not roots of the input system. There are other techniques (presented elsewhere) that can be used to eliminate those redundant points.

We could easily develop a randomized algorithm to detect whether or not the zero set of the input system has some (complex) positive-dimensional components. Namely, we run the algorithm **RUR** a few times. If the zero set of the input system is of dimension zero then, at each execution of the algorithm **RUR**, the same set of points are returned. On the other hand, if the zero set of the input system has some positive-dimensional components then, at each execution, different points would be picked from those positive-dimensional components. While it would also be nice to develop means of actually finding a representation for the positive-dimensional components of the zero set of the input system, this would be a much harder problem.

We have not analyzed the asymptotic arithmetic complexity of our algorithm for an overdetermined system. We strongly believe that the arithmetic complexity of our algorithm is worse compared to the algorithm described in Rojas (2000), although the new algorithm behaves better for small n. A bit-complexity analysis of these algorithms would also be useful.

Although we have included some efficiency improvements, the implementation of our algorithms can be further optimized. One avenue in particular would be to use better algorithms for constructing the resultant matrices or evaluating the toric resultant. Any improvement here would be helpful, since the resultant computation governs the performance of the algorithms both in theory and practice. Also, faster subroutines for linear algebra operations and polynomial ring operations would be useful. For instance, we should take advantage of the sparse structure of the resultant matrix when its determinant is evaluated (Emiris and Pan, 2002).

Also, it would be nice to extend our implementation to include coefficients belonging to a field other than the field of rational numbers. Over a finite field, an implementation must look different since we cannot use techniques that work over a field of characteristic 0. Also, an implementation for coefficients that are real or complex algebraic numbers would be interesting. Even more generally, there are practical reasons to consider polynomials whose coefficients are not given exactly. In contrast to Gröbner basis approaches to determining the RUR, the toric approach is continuous over perturbations in the coefficients, thus there is some hope that it

would offer a method for dealing with such input. Developing theory and implementation to support such polynomials would be very valuable.

Finally, this work was motivated by work on robust geometric computation. We are currently exploring application of our algorithms and implementation in this direction.

Acknowledgements

The authors were partially supported by NSF ITR award CCR-0220047 and NSF/DARPA CARGO award DMS-0138446.

References

Aubry, P., Rouillier, F., El Din, M.S., 2002. Real solving for positive dimensional systems. Journal of Symbolic Computation 34 (6), 543–560.

Basu, S., Pollack, R., Roy, M.-F., 2003. Algorithms in Real Algebraic Geometry. Springer.

Bernstein, D.N., 1975. The number of roots of a system of equations. Functional Analysis and its Applications 9 (2), 183–185.

Blum, L., Cucker, F., Shub, M., Smale, S., 1997. Complexity and Real Computation. Springer.

Canny, J., 1988. Some algebraic and geometric computations in pspace. In: Proc. of 20th Annual Symposium on Theory of Computation. ACM, pp. 460–467.

Canny, J.F., 1990. Generalized characteristic polynomials. Journal of Symbolic Computation 9 (3), 241–250.

Canny, J.F., Emiris, I.Z., 1993. An efficient algorithm for the sparse mixed resultant. In: Proc. of 10th AAECC. In: LNCS, vol. 673. Springer, pp. 89–104.

Canny, J.F., Emiris, I.Z., 2000. A subdivision-based algorithm for the sparse resultant. Journal of ACM 47 (3), 417–451.

Cox, D., 2003. What is a toric variety? In: Goldman, R., Krasauskas, R. (Eds.), Topics in Algebraic Geometry and Geometric Modeling. In: Contemporary Mathematics, vol. 334. AMS, pp. 203–223.

Cox, D., Little, J., O'Shea, D., 1996. Ideals, Varieties, and Algorithms, second edition. Springer.

Cox, D., Little, J., O'Shea, D., 1998. Using Algebraic Geometry. Springer.

D'Andrea, C., 2002. Macaulay style formulas for sparse resultants. Transactions of the AMS 354 (7), 2595–2629.

D'Andrea, C., Emiris, I.Z., 2001. Computing sparse projection operators. In: Green, E.L., et al. (Eds.), Symbolic Computation: Solving equations in Algebra, Geometry and Engineering. In: Contemporary Mathematics, vol. 286. AMS, pp. 121–139.

D'Andrea, C., Emiris, I.Z., 2003. Sparse resultant perturbation. In: Joswig, M., Takayama, N. (Eds.), Algebra, Geometry, and Software. Springer, pp. 93–107.

Dickenstein, A., Emiris, I.Z., 2003. Multihomogeneous resultant formulae by means of complexes. Journal of Symbolic Computation 36 (3–4), 317–342.

Din, M.S.E., Shost, E., 2004. Properness defects of projections and computation of at least one point in each connected component of a real algebraic set. Discrete and Computational Geometry 32 (3), 417–430.

Dykstra, P.C., Muuss, M.J., 1989. The BRL-CAD package an overview. Tech. rep., Advanced Computer Systesms Team, Ballistics Research Laboratory, Aberdeen Proving Ground, MD http://ftp.arl.mil/brlcad/.

Emiris, I.Z., 1996. On the complexity of sparse elimination. Journal of Complexity 12 (2), 134–166.

Emiris, I.Z., 1998. A complete implementation for computing general dimensional convex hulls. International Journal of Computational Geometry and Applications 8 (2), 223–253.

Emiris, I.Z., 2002. Enumerating a subset of the integer points inside a Minkowski sum. Computational Geometry 22 (1–3), 143–166.

Emiris, I.Z., 2003. Discrete geometry for algebraic elimination. In: Joswig, M., Takayama, N. (Eds.), Algebra, Geometry, and Software. Springer, pp. 77–91.

Emiris, I.Z., Canny, J.F., 1995. Efficient incremental algorithm for the sparse resultant and the mixed volume. Journal of Symbolic Computation 20 (2), 117–149.

Emiris, I.Z., Pan, Y.V., 2002. Symbolic and numeric methods for exploiting structure in constructing resultant matrices. Journal of Symbolic Computation 33 (4), 393–413.

Gelfand, I.M., Kapranov, M.M., Zelevinsky, A.V., 1994. Discriminants, Resultants, and Multidimensional Determinants. Birkhauser.

Giusti, M., Lecerf, G., Salvy, B., 2001. A Gröbner free alternative for polynomial system solving. Journal of Complexity 17 (1), 154–211.

Gonzalez-Vega, L., 1991. A subresultant theory for multivariate polynomials. In: Proc. of ISSAC '91. ACM, pp. 79–85.
Gonzalez-Vega, L., Rouillier, F., Roy, M.-F., 1999. Symbolic recipes for polynomial system solving. In: Cohen, A.M., Cuypers, H., Sterk, H. (Eds.), Some Tapas of Computer Algebra. In: Algorithms and Computation in Mathematics, vol. 4. Springer, pp. 34–65.
Hong, H., Minimair, M., 2002. Sparse resultant of composed polynomials I mixed-unmixed case. Journal of Symbolic Computation 33 (4), 447–465.
Huber, B., Sturmfels, B., 1997. Bernstein's theorem in affine space. Discrete and Computational Geometry 17 (2), 137–141.
Jeronimo, G., Krick, T., Sabia, J., Sombra, M., 2004. The computational complexity of the chow form. Foundations of Computational Mathematics 4 (1), 41–117.
Kaltofen, E., Villard, G., 2004. Computing the sign or the value of the determinant of an integer matrix, a complexity survey. Journal of Computational and Applied Mathematics 162 (1), 133–146.
Keyser, J., Ouchi, K., Rojas, J.M., 2005. The exact rational univariate representation and its application. In: Janardan, R., Smid, M., Dutta, D. (Eds.), Geometric and Algorithmic Aspects of Computer-Aided Design and Manufacturing. In: AMS/DIMACS, vol. 67. AMS, pp. 299–328.
Khetan, A., 2003. The resultant of an unmixed bivariate system. Journal of Symbolic Computation 36 (3–4), 425–442.
Khetan, A., 2005. Exact matrix formula for the unmixed resultant in three variables. Journal of Pure and Applied Algebra 198 (1–3), 237–256.
Kronecker, L., 1895–1931. Leopold Kronecker's Werke. Teubner.
Lecerf, G., 2002. Quadratic Newton iteration for systems with multiplicity. Journal of Foundations of Computational Mathematics 2 (3), 247–293.
Li, T.Y., Wang, X., 1996. The BKK root count in C^n. Mathematics of Computation 65 (216), 1477–1484.
Minimair, M., 2002. Sparse resultant of composed polynomials II mixed-unmixed case. Journal of Symbolic Computation 33 (4), 467–478.
Mourrain, B., 1999. A new criterion for normal form algorithms. In: Proc. of 13th International Symposium, AAECC 13 '99. In: LNCS, vol. 1719. Springer, pp. 430–443.
Pedersen, P., Sturmfels, B., 1993. Product formulas for resultants and chow forms. Mathematische Zeitschrift 214 (3), 377–396.
Rojas, J.M., 1997. Toric laminations, sparse generalized characteristic polynomials, and a refinement of Hilbert's tenth problem. In: Cucker, F., Shub, M. (Eds.), Foundations of Computational Mathematics. Springer, pp. 369–381.
Rojas, J.M., 1999a. Solving degenerate sparse polynomial systems faster. Journal of Symbolic Computation 28 (1–2), 155–186.
Rojas, J.M., 1999b. Toric intersection theory for affine root counting. Journal of Pure and Applied Algebra 136 (1), 67–100.
Rojas, J.M., 2000. Algebraic geometry over four rings and the frontier to tractability. In: J., D., et al. (Eds.), Hilbert's Tenth Problem: Relations with Arithmetic and Algebraic Geometry. In: Contemporary Mathematics, vol. 270. AMS, pp. 275–321.
Rojas, J.M., 2003. Why polyhedra matter in non-linear equation solving. In: Goldman, R., Krasauskas, R. (Eds.), Topics in Algebraic Geometry and Geometric Modeling. In: Contemporary Mathematics, vol. 334. AMS, pp. 293–320.
Rojas, J.M., Wang, X., 1996. Counting affine roots of polynomial systems via pointed Newton poly topes. Journal of Complexity 12 (2), 116–133.
Rouillier, F., 1999. Solving zero-dimensional systems through the rational univariate representation. Applicable Algebra in Engineering, Communication and Computing 9 (5), 433–461.
Sturmfels, B., 1994. On the Newton polytope of the resultant. Journal of Algebraic Combinatorics 3 (2), 207–236.
Sturmfels, B., 2002. Solving Systems of Polynomial Equations. AMS.
von zur Gathen, J., Gerhard, J., 2003. Modern Computer Algebra, second edition. Cambridge University Press.