A Gröbner Free Alternative for Polynomial System Solving

Grégoire Lecerf

Laboratoire GAGE, École polytechnique

July 5, 1999

[summary by Éric Schost]

Abstract

Let $f_1, \ldots, f_n$ and g be polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$, such that the system $f_1 = \cdots = f_n = 0$ and $g \neq 0$ has only a finite number of solutions. Following a long series of theoretical papers, G. Lecerf, M. Giusti and B. Salvy propose a new algorithm to obtain a *geometric resolution* of the zero-set of the system. This algorithm is valid under the hypothesis that the system $f_1, \ldots, f_n$ forms a reduced, regular sequence outside $V(g)$. This talk presents the complexity of the algorithm, details some crucial steps and finally compares the implementation in Magma called Kronecker[1] to other available softwares. It is based on [3].

1. Introduction

The algorithm presented here is devoted to the resolution of zero-dimensional systems. For a zero-dimensional variety $\mathcal{V}$, a geometric resolution consists in the following elements:

- a *primitive element* u of the extension $\mathbb{Q} \to \mathbb{Q}[\mathcal{V}]$;
- its *minimal polynomial* $q \in \mathbb{Q}[U]$;
- the *parametrization* of the coordinates of the form $q'(u)x_i = w_i(u)$, $1 \leq i \leq n$

such that $\{q(u) = 0, x_i = w_i(u)/q'(u)\}$ is a description of the points of $\mathcal{V}$.

The algorithm presented by G. Lecerf lies in the continuation of earlier, theoretical work by the TERA group [1, 2] that gave algorithms to obtain geometric resolutions of reduced, regular systems.

The present work inherits the specifics of these papers: the algorithm takes into account the evaluation properties of the input system and can work outside a given hypersurface. It also includes refinements that simplify previous algorithms and improve their complexity. Finally, note that it is a probabilistic algorithm. The author proposes an implementation in Magma called Kronecker that validates this approach.

In the specific case of zero-dimensional systems, a geometric resolution also bears the name Rational Univariate Representation, following F. Rouillier's denomination. In [8], he gives an algorithm to compute this object that relies on the precomputation of a Gröbner basis. This computation is avoided here, whence the title.

The presentation is organized as follows. First, a rough sketch of the resolution algorithm is given. The complexity of this algorithm is then stated in terms of some suitably defined quantities. The most relevant steps of the resolution are detailed. The implementation is compared with

[1] See also the Kronecker homepage: `http://www.gage.polytechnique.fr/~lecerf/software/kronecker`

implementations of Gröbner bases and Rational Univariate Representation (Gb by J.-C. Faugère, RealSolving by F. Rouillier), and of Gröbner bases in Magma.

2. Outlook of the Algorithm

Notation. $\mathcal{V}_i$ will denote $\overline{V(f_1,\ldots,f_i)\backslash V(g)}$.

Input. The input system $f_1,\ldots,f_n$ must form a regular, reduced sequence outside $V(g)$, that is:

- $V(f_{i+1})$ intersects $\mathcal{V}_i$ regularly, so that $\dim \mathcal{V}_i = n-i$;
- $\mathcal{V}_i$ is reduced.

These polynomials will be evaluated on many objects, so they are thought (and given) as Straight-Line Programs.

Output. The output of the algorithm is a geometric resolution of $\mathcal{V}_n$.

Sketch of the Algorithm. The algorithm is an iterative intersection process. It works the following way:

- Apply a generic linear change of coordinates.
- Compute incrementally a resolution in $\mathbb{Z}/p\mathbb{Z}$, for some prime p. The i-th step consists in:

$$\text{Resolution of } \mathcal{V}_i \text{ at } x_1,\ldots,x_{n-i}=0$$

$$\downarrow \text{Lifting of } x_{n-i}$$

$$\text{Resolution of } \mathcal{V}_i \text{ at } x_1,\ldots,x_{n-i-1}=0$$

$$\downarrow \text{Intersection}$$

$$\text{Resolution of } \mathcal{V}_i \cap V(f_{i+1}) \text{ at } x_1,\ldots,x_{n-i-1}=0$$

$$\downarrow \text{Cleaning}$$

$$\text{Resolution of } \mathcal{V}_{i+1} \text{ at } x_1,\ldots,x_{n-i-1}=0$$

- Unapply the change of variables.
- Lift the resolution to a big enough precision p^k and reconstruct a rational resolution.

Why a Generic Change of Variables? The algorithm heavily relies on specializations and unspecializations of variables, so that we need generic enough coordinates. Thus, the linear change of variables must ensure that all the intermediate varieties $\mathcal{V}_i$ are in Noether position, that their fiber above $x_1,\ldots,x_{n-i}=0$ is reduced and that the first dependent variable x_{n-i+1} is a separating (primitive) element for this fiber.

Are There SLP's Left ? The rough algorithm shown above handles only zero- and one-dimensional varieties, as the reader will easily check, with the consequence that all computations are performed on uni- or bi-variate polynomials, which are not represented by SLP's. This follows the deforestation technique introduced in [9] and already used in [4]. Only the input polynomials are coded as SLP's, as they are evaluated on various data throughout the resolution.

Probabilistic Aspects. The algorithm is probabilistic. The success relies on a lucky prime number p and a lucky linear change of coordinates. The unlucky change of coordinates are enclosed in a strict algebraic subset of $\mathrm{GL}(n)$.

The algorithm is not Las Vegas. Still, we can test whether the solution it outputs satisfies the input system. If it does, we might only miss some of the solutions. In the special case when the output contains $\prod \deg(f_i)$ solutions, we know we have all of them.

3. Complexity

The quantities that appear in the complexity are of two kinds. First, the syntactic complexity of the input system:

- d, the maximum of the degrees of the polynomial $f_1, \ldots, f_n$;
- h, the maximum of the heights of these polynomials;
- L, the number of binary arithmetic operations required to evaluate the polynomials $f_1, \ldots, f_n$ and g; that is the size of the SLP that encodes them.

The complexity also depends on some parameters that are intrinsic to the varieties $\mathcal{V}_i$:

- δ, the maximum of the degrees of the varieties $\mathcal{V}_1, \ldots, \mathcal{V}_{n-1}$;
- D, the degree of $\mathcal{V}_n$. It is the number of solutions of the system;
- η, the height of the integers of the resolution (that depends on the choice of the primitive element).

The following theorem establishes the time-complexity of the whole resolution process. As usual, the term $O_{\log}(X)$ indicates a complexity of X up to logarithmic factors. The constant Ω is related to the complexity of linear algebra operations over a commutative ring. Note that $\Omega < 3$ is valid on a field. On a ring, we may take $\Omega \leq 4$, using Berkowitz' algorithm.

Theorem 1. *The time-complexity of the modular computations is*

$$n(nL + n^{\Omega}) O_{\log}\left((d\delta)^2\right).$$

The time necessary to lift the integers is

$$(nL + n^{\Omega}) O_{\log}(D) O_{\log}(\eta).$$

This complexity improves upon earlier results. The papers [1] gave algorithms with complexity $L(nd\delta\eta h)^{O(1)}$. More recently, [5] refines these algorithms, so that the power of δ that appears is 3.

The following three sections detail the iterative step of the modular resolution, and in particular:

- the lifting step that now relies on a global Newton process;
- the intersection step that uses methods that go back to Kronecker [6], and are detailed in [7].

The minimal polynomial of a resolution will always be noted q (or $\tilde{q}$, Q, ...). The situation is slightly more complicated regarding parametrizations. The reader should be aware that a resolution can bear many forms, depending on the choice of the denominator of the parametrization. The so-called "Kronecker" form has the derivative of the minimal polynomial as denominator; the associated parametrization will be noted $\mathbf{w}$. The polynomial parametrization (i.e., with denominator 1) will be noted $\mathbf{v}$.

It is always possible to go from one type of parametrization to another one, by inverting the denominator modulo the minimal polynomial. This will often be done, without further notice.

4. **Lifting**

We enter the i-th step with a resolution of the variety $\mathcal{V}_i$ specialized at $x_1, \ldots, x_{n-i} = 0$, which is a zero-dimensional set. The lifting step "unspecializes" the last free variable x_{n-i} and thus builds a resolution of the one-dimensional set $\mathcal{V}_i$ specialized at $x_1, \ldots, x_{n-i-1} = 0$, which will be called a *lifted curve*.

Input. A geometric resolution modulo p of $\mathcal{V}_i$ at $x_1, \ldots, x_{n-i} = 0$, with primitive element x_{n-i+1}:

$$q(x_{n-i+1}) = 0, \qquad \begin{cases} x_{n-i+1} & = v_{n-i+1}(x_{n-i+1}), \\ & \vdots \\ x_n & = v_n(x_{n-i+1}). \end{cases}$$

Output. The lifted curve is represented as a parametrized version of the previous representation, where now both the minimal polynomial and the parametrizations have coefficients in $\mathbb{Z}/p\mathbb{Z}[x_{n-i}]$. The minimal polynomial and the parametrizations in Kronecker form have total degree less than the degree of q.

$$Q(x_{n-i}, x_{n-i+1}) = 0, \qquad \begin{cases} x_{n-i+1} & = V_{n-i+1}(x_{n-i}, x_{n-i+1}), \\ & \vdots \\ x_n & = V_n(x_{n-i}, x_{n-i+1}). \end{cases}$$

How Does it Work? The core of the routine is a global Newton iteration, which is summarized in the following scheme.

$$\begin{array}{c} q(x_{n-i+1}) = 0, \\ \mathbf{f}(\mathbf{v}) = 0 + O(x_{n-i}) \mod q, \end{array} \qquad \begin{cases} x_{n-i+1} & = v_{n-i+1}(x_{n-i+1}), \\ & \vdots \\ x_n & = v_n(x_{n-i+1}). \end{cases}$$

$$\downarrow \text{Global Newton}$$

$$\begin{array}{c} \tilde{q}(x_{n-i}, x_{n-i+1}) = 0 + O(x_{n-i}^2), \\ \mathbf{f}(x_{n-i}, \tilde{\mathbf{v}}) = 0 + O(x_{n-i}^2) \mod \tilde{q}, \end{array} \qquad \begin{cases} x_{n-i+1} & = \tilde{v}_{n-i+1}(x_{n-i}, x_{n-i+1}), \\ & \vdots \\ x_n & = \tilde{v}_n(x_{n-i}, x_{n-i+1}). \end{cases}$$

This makes sense, since at the beginning of the i-th step, the system can be seen as polynomials over $\mathbb{Z}/p\mathbb{Z}[x_{n-i}]$ for which we have a resolution modulo $I = (x_{n-i})$. A global Newton iteration enables to compute a resolution to the precision I^2, and, by successive applications, to any I^{2^k}. The bound on the degrees in the Kronecker form indicates the number of steps to perform. Let's detail the first pass.

The point is to consider that we are given a representation modulo I of an underlying resolution over the ring $\mathbb{Z}/p\mathbb{Z}[x_{n-i}]$. This resolution has a separating element T which is x_{n-i+1} modulo I, and such that $\mathbf{f}(x_{n-i}, \mathbf{v}(T)) = 0 + O(x_{n-i})$ modulo $q(T)$—recall that this only means that the input is a resolution for the specialization $x_{n-i} = 0$.

First, an iteration of the classical Newton iterator yields new parametrizations V that satisfy $\mathbf{f}(x_{n-i}, \mathbf{V}(T)) = 0 + O(x_{n-i}^2)$ modulo $q(T)$. This is not satisfying yet, for we seek expressions that involve x_{n-i}, not T. The trick is to see that the new parametrization of x_{n-i+1} is

$x_{n-i+1} = T + x_{n-i}\Delta(T) + O(x_{n-i}^2)$, for some polynomial Δ with coefficients in $\mathbb{Z}/p\mathbb{Z}$, according to the remark above. Furthermore, $x_{n-i}\Delta(T)$ is $x_{n-i}\Delta(x_{n-i+1})$ modulo I^2 (this is a first-order expansion). The corresponding substitution $T \leftarrow x_{n-i+1} - x_{n-i}\Delta(x_{n-i+1})$ thus yields the desired resolution modulo I^2.

5. Intersection

We go back to the description of a step of the iterative resolution process. This second part consists in intersecting $\mathcal{V}_i$ specialized at $x_1, \ldots, x_{n-i-1} = 0$ with the hypersurface defined by f_{i+1}.

Input. The output of the lifting step and f_{i+1}

Output. A geometric resolution of $\mathcal{V}_i \cap V(f_{i+1})$ at $x_1, \ldots, x_{n-i-1} = 0$ with x_{n-i} as primitive element.

To this effect, we perform the following linear change of variables: $x_{n-i} = X - tx_{n-i+1}$. The corresponding minimal polynomial and parametrizations are noted $Q_t(X, x_{n-i+1})$ and $\mathbf{V}_t(X, x_{n-i+1})$. This idea naturally leads to what we called Kronecker parametrizations, and is already to be found in his papers.

We compute $A := \text{Resultant}_{x_{n-i+1}}(Q_t, f_{i+1}(X - tx_{n-i+1}, \mathbf{V}_t))$ at order $O(t^2)$, that we shall write $A = A_0 + tA_1 + O(t^2)$. As $A(x_{n-i} + tx_{n-i+1}) = 0$, it follows that the eliminating polynomial of x_{n-i} is A_0 and that the the parametrization of x_{n-i+1} is $-A_1(x_{n-i})/A_0'(x_{n-i})$.

6. Cleaning Step

This is the last part of each iteration, that consists in removing the unwanted components lying in $V(g)$.

Input. A resolution of $\mathcal{V}_i \cap V(f_{i+1})$ at $x_1, \ldots, x_{n-i-1} = 0$ as produced by the intersection routine.

Output. A resolution of $\mathcal{V}_{i+1}$ at $x_1, \ldots, x_{n-i-1} = 0$.

The computation is straightforward: compute the gcd p of q and $g(v)$ and replace q by q/p to obtain a new resolution.

7. Experimental Results

The algorithm is implemented in Magma, in a package called Kronecker. It has been compared with existing softwares, namely Gb, RealSolving and Magma on different examples. All computations were done on the MEDICIS machines (`http://www.medicis.polytechnique.fr`).

The first series of examples consists in n generic polynomials of degree 2, with coefficients of h decimal digits, for various n and h. The systems have then full degree $D = 2^n$. This example aims at illustrating the good behaviour of Kronecker regarding the growth of the coefficients. The timings are given in Table 1. The computations were performed on a Compaq Alpha EV6, 500 MHz, 128 Mb.

The second example (Table 2) is inspired by the Kruppa equations. It consists in 7 equations in 7 variables with integers coefficients of size 18, each equation is a product of two linear forms minus a constant coefficient. The computations were performed on a DEC Alpha EV56, 400 Mhz, OSF/1 4.0b, 1024 Mb.

n	h	Kronecker	Gb Grevlex + Real Solving
4	4	6 s	0.5s +0.5s
4	8	8 s	1s + 1.3s
4	16	11s	2.5s + 3.7s
4	32	21s	7s + 9.3s
5	4	36s	5s + 18s
5	8	50s	17s + 57s
5	16	88s	65s + 180s
5	32	208s	244s + 592s
6	4	260s	209s + >317s
6	8	412s	773s + ∞
6	16	875s	2999s + ∞
6	32	2312s	5652s + ∞

TABLE 1

Kronecker	Magma Grevlex
5h	13.6h

TABLE 2

Bibliography

[1] Giusti (M.), Hägele (K.), Heintz (J.), Morais (J. E.), Montaña (J. L.), and Pardo (L. M.). – Lower bounds for Diophantine approximation. *Journal of Pure and Applied Algebra*, vol. 117/118, 1997, pp. 277–317. – Proceedings MEGA'96.

[2] Giusti (M.), Heintz (J.), Morais (J. E.), Morgenstern (J.), and Pardo (L. M.). – Straight-line programs in geometric elimination theory. *Journal of Pure and Applied Algebra*, vol. 124, 1998, pp. 101–146.

[3] Giusti (M.), Lecerf (G.), and Salvy (B.). – A Gröbner free alternative for polynomial system solving. In *Proceedings FOCM'99*. – 1999.

[4] Giusti (Marc), Hägele (Klemens), Lecerf (Grégoire), Marchand (Joël), and Salvy (Bruno). – *Computing the Dimension of a Projective Variety: the Projective Noether Maple Package*. – Research report n° 3224, Institut National de Recherche en Informatique et en Automatique, July 1997.

[5] Heintz (J.), Matera (G.), and Waissbein (A.). – On the time-space complexity of geometric elimination procedures. – 1999. Manuscript of Universidad Favaloro, Buenos Aires, Argentina.

[6] Kronecker (L.). – Grundzüge einer arithmetischen Theorie der algebraischen Grössen. *Journal für die reine und angewandte Mathematik*, vol. 92, 1882, pp. 1–122.

[7] Macaulay (F. S.). – *The Algebraic Theory of Modular Systems*. – Cambridge University Press, 1916.

[8] Rouillier (F.). – *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux*. – PhD thesis, Université de Rennes I, may 1996.

[9] Wadler (P.). – Deforestation: transforming programs to eliminate trees. *Theoretical Computer Science*, vol. 73, 1990, pp. 231–248. – Special issue of selected papers from 2nd ESOP.