Reading Rational Univariate Representations on lexicographic Gröbner bases.

Alexander Demin
National Research University, Higher School of Economics
Moscow, Russia
asdemin_2@edu.hse.ru

Fabrice Rouillier
Sorbonne Université, Paris Université, CNRS (Institut de Mathématiques de Jussieu Paris-Rive-Gauche), Inria
Paris, France
Fabrice.Rouillier@inria.fr

Joao Ruiz
Sorbonne Université, Paris Université, CNRS (Institut de Mathématiques de Jussieu Paris-Rive-Gauche), Inria
Paris, France
joao.ruiz@imj-prg.fr

ABSTRACT

In this contribution, we consider a zero-dimensional polynomial system in n variables defined over a field $\mathbb{K}$. In the context of computing a Rational Univariate Representation (RUR) of its solutions, we address the problem of certifying a separating linear form and, once certified, calculating the RUR that comes from it, without any condition on the ideal else than being zero-dimensional. Our key result is that the RUR can be read (closed formula) from lexicographic Gröbner bases of bivariate elimination ideals, even in the case where the original ideal that is not in shape position, so that one can use the same core as the well known FGLM method to propose a simple algorithm. Our first experiments, either with a very short code (300 lines) written in Maple or with a Julia code using straightforward implementations performing only classical Gaussian reductions in addition to Groebner bases for the degree reverse lexicographic ordering, show that this new method is already competitive with sophisticated state of the art implementations which do not certify the parameterizations.

ACM Reference Format:
Alexander Demin, Fabrice Rouillier, and Joao Ruiz . 2024. Reading Rational Univariate Representations on lexicographic Gröbner bases. . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Let $\mathbb{K}$ be a field with algebraic closure $\bar{\mathbb{K}}$. Consider a zero-dimensional ideal $I \subset \mathbb{K}[X_1, \ldots, X_n]$ and $V(I) = \{x \in \bar{\mathbb{K}}^n, p(x) = 0, \forall p \in I\}$ its (finite) set of zeroes in $\bar{\mathbb{K}}^n$. We assume that the characteristic of $\mathbb{K}$ is either zero or sufficiently large. A Rational Univariate Representation of $V(I)$ is a rational parameterization of the zeroes of I with coefficients in $\mathbb{K}$, that preserves the multiplicities. It is defined as follows in [20] :

- a linear form $T = \sum_{i=1}^{n} a_i X_i$ that is injective on $V(I)$ (separating form)
- $n+1$ univariate polynomials $f_T, f_{X_1}, \ldots, f_{X_n} \in \mathbb{K}[T]$ of degrees at most D for f_T and at most $\delta - 1$ for the others, where $\delta = \sharp V(I)$ denotes the number of zeros of I and D the number of zeroes of of I counted with multiplicities such that the following application defines a bijection

$$\begin{array}{ccc} V(I) & \longrightarrow & V(f_T) \\ (\alpha_1, \ldots, \alpha_n) & \mapsto & T(\alpha_1, \ldots, \alpha_n) \\ \left(\frac{f_{X_1}(\beta)}{f_1(\beta)}, \ldots, \frac{f_{X_n}(\beta)}{f_1(\beta)}\right) & \leftarrow & \beta \end{array}$$

where f_1 denotes the derivative of the squarefree part $\overline{f_T}$ of f_T. The multiplicities are preserved, say the dimension of the localization of $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ at $\alpha \in V(I)$ equals the dimension of the localization of $\frac{\mathbb{K}[T]}{\langle f_T \rangle}$ at $t(\alpha) \in V(f_T)$.

A RUR is uniquely defined (up to a multiplication of $\overline{f_T}$ by a scalar) whenever the separating linear form is fixed.

The formulas and algorithms proposed in [20] were inspired by [1] where the authors define the polynomials that correspond to each possible multiplicity by means of specialized characteristic polynomials of elements of $\frac{\mathbb{Q}[X_1,\ldots,X_n]}{I}$ and are derivations of the so-called Chow forms which can also be viewed as factors of specializations of the u-resultant of the homogeneized system.

The u-resultant of a system with a finite number of roots is the polynomial $M(U_0, \ldots, U_n) \in \mathbb{K}[U_0, \ldots, U_n]$ defined by

$$M(U_0, \ldots, U_n) = \prod_{j=1}^{l} (\alpha_{0,j} U_0 + \cdots + \alpha_{n,j} U_n)^{\mu_j}$$

so that the zeroes of I are $\left(\frac{\alpha_{1,j}}{\alpha_{0,j}}, \ldots, \frac{\alpha_{n,j}}{\alpha_{0,j}}\right), \alpha_{0,j} \neq 0, j = 1 \ldots l$ with respectives multiplicities the corresponding μ_j. It can be rewritten as $M(U_0, \ldots, U_n) = H(U_1, \ldots, U_n) R(U_0, \ldots, U_n)$, with

$$R(U_0, \ldots, U_n) = \prod_{\beta \in V(I)} (U_0 + \beta_1 U_1 + \cdots + \beta_n U_n)^{\mu(\beta)}.$$

If $T = \sum_{i=1}^{n} t_i U_i \in \mathbb{K}[U_0, \ldots, U_n]$, $f_T(U_0) = R(U_0, t_1, \ldots, t_n)$ is then the characteristic polynomial of the multiplication by T in $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$. Taking $\overline{R}(U_0, \ldots, U_n) = \prod_{\beta \in V(I)} (U_0 + \beta_1 U_1 + \cdots + \beta_n U_n)$, the squarefree part of R in $\mathbb{K}[U_0, \ldots, U_n]$, one can then define $f_{X_i} = \frac{\partial \overline{R}}{\partial U_i}(U_0, t_1, \ldots, t_n), i = 1 \ldots n$ and $f_1 = \frac{\partial \overline{R}}{\partial U_0}(U_0, t_1, \ldots, t_n)$ so that if T separates $V(I)$, $f_T, f_{X_1}, \ldots, f_{X_n}$ is the RUR of I associated to T.

As the u-resultant can be viewed as the determinant of a matrix (Macaulay matrix) whose entries are zeroes or coefficients of the polynomials defining the system, its coefficients's sizes are well controlled. For example, when $\mathbb{K} = \mathbb{Q}$, the bitsize of the coefficients of a RUR for a large class of separating forms is bounded by

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$\tilde{O}(\lambda^{n-1}\tau)$ when the polynomials defining the system are of maximal degree λ in each of the n variables and with coefficients of bitsize at most τ, thanks to Hadamard's bound on the determinant of an arbitrary matrix and Mignotte's bounds on the coefficients of factors of polynomials (see [11] for an exhaustive summary).

1.1 The problems addressed

Computing a RUR consists in computing a set of polynomials depending on a linear form that is named RUR-Candidate until it is proved/confirmed that the used linear form is separating.

It turns out that computing a separating form or checking that a linear form is a separating form seem to be as difficult as computing the RUR-Candidate itself and the certification of the separating linear form is skipped in most of the *efficient* implementations, using the fact that a randomly chosen linear form is separating with a high probability.

If the quotient algebra is supposed to be cyclic, then a linear form is separating if and only if its characteristic polynomial is also its minimal polynomial.

The checking is straightforward in the case of radical ideals, since it suffice to check that f_T is squarefree. It is still possible when the quotient algebra that $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ is cyclic (generated by powers of a unique element) by checking that the minimal and characteristic polynomial of T coincides. The checking becomes hard in the general case. Very few algorithms propose the certification in the general case. One can mention [20] or [25].

1.2 From any parameterization to the RUR of the radical

The RUR of I associated to T differs from the RUR of $\sqrt{I}$ associated to T only by the first polynomial which becomes $\overline{f_T}$ in the case of the RUR of $\sqrt{I}$ and thus the general results on the size of a RUR also cover the case of parameterizations that do not preserve the multiplicities.

Also passing from a RUR of $\sqrt{I}$ to a RUR of I can be done in $\tilde{O}(M(D))$ arithmetic operations in the ground field if $M(D)$ denotes the number of arithmetic operations in $\mathbb{K}$ to compute the product of dense matrices of dimension D.

Because of its well controlled size, it is often a good idea to transform a polynomial parametrization into a RUR and this can be done by simple operations on univariate polynomials : suppose that T is separating and that you have a parameterization of each coordinate of the roots of the system in the form $X_j = h_j(T)$ and any polynomial f_T that is a multiple of the minimal polynomial of T in $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$, then $f_{X_j} = \left(\overline{f_T}\right)' h_j \bmod \overline{f_T}$.

Algorithm 1 Parameterization to a RUR of the radical

Require: $\{f(T), h_{1,1}(T)X_1 + h_{1,0}(T), \ldots, h_{n,1}(T)X_n + h_{n,0}(T)\}$ a parameterization of a zero-dimensional Ideal.
Ensure: $\{T, f_T(T), f_{X_1}(T), \ldots, f_{X_n}(T)\}$ a RUR of $\sqrt{I}$.
Compute $f_T = \overline{f}$ the squarefree part of f
for $j = 1, \ldots, n$ **do**
$f_{X_j} = -h_{j,0}({h_{j,1}}^{-1})\overline{f}' \bmod \overline{f}$.
end for

1.3 Gröbner bases, quotient algebra and RURs

Gröbner bases offer several tools to serve the computation of a RUR. A good input for the RUR that can be provided by a Gröbner basis for any term ordeing is the knowledge of a basis $\mathcal{B}$ of $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ and of the matrices $M_{X_1}, \ldots, M_{X_n}$ of multiplication by each of the wariables in $\mathcal{B}$.

It can be noticed that, given any linear form $T = \sum_{i=1}^n t_i X_i$, $\mathcal{B}$ is also a basis of $\frac{\mathbb{K}[T,X_1,\ldots,X_n]}{I_T}$, where $I_T = I + \langle T - \sum_{i=1}^n t_i X_i\rangle$, and if $G_{>_n}$ is a the Gröbner basis of I for any term ordering $>_n$, then $G_{>_n} \cup \left\{T - \sum_{i=1}^n t_i X_i\right\}$ is a Gröbner basis for any product of orderings $(>_n, >_T)$ where $>_T$ is the canonical univariate ordering for polynomials that depend on the variable T.

When $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ is cyclic, a linear form $T = \sum_{i=1}^n t_i X_i$ separates $V(I)$ if and only if $1, T, \ldots, T^{D-1}$ is a basis of $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ and, in this particular case, a lexicographic Gröbner basis of $I_T \subset \mathbb{K}[T, X_1, \ldots, X_n]$ for any elimination ordering where $T < X_i, i = 1 \ldots n$ will have a simple shape defining directly a parameterization: $f_T(T) = 0, X_i = f_i(T), i = 1 \ldots n$.

1.4 Deterministic, Las-Vegas and Monte-Carlo algorithms

In many recent contributions for zero-dimensional systems with rational coefficients, the RUR is used in the place of a lexicographical Gröbner basis whenever the ideal is said to be in *shape position*, which means that the quotient algebra is cyclic and generated by powers of the smallest variable wrt the considered monomial ordering. This use is due to the fact that the size of a RUR is better controlled than the size of a lexicographic Gröbner basis.

As already seen above, one can consider any kind of parameterization since it can always be converted into a RUR of the solutions of the radical ideal with little additional work.

The probabilistic aspects of behavior of the various algorithms that compute parameterizations come from 3 main sources :

- the shape of the problem, mainly whether the quotient algebra $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ is cyclic or not. When it is cyclic, checking that a randomly chosen linear form is separating or not remains to checking that its minimal polynomial has degree $D = \dim_{\mathbb{K}}\left(\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}\right)$ and is thus easy.
- checking that a linear form is (or isn't) separating: as much as it is trivial in the case of a cyclic quotient algebra, it becomes hard in the general case and is almost never addressed in the litterature. In particular, it is still unclear that this checking is (or is not) the bottleneck in the general case.
- deterministic, Las Vegas or Monte-Carlo sub-algorithms such as classical Gaussian elimination vs Wiedemann algorithm for computing minimal polynomials, GCDs of univariate polynomials or strategies based on heuristics for Chinese remaindering and rational reconstructions in the case of for systems with rational coefficients.

It can thus be noticed that the complexity of computation of a RUR faces to the same problems as those conditionning the FGLM algorithm (see [12]) for computing lexicographic Gröbner basis with, in addition, the problem of finding (or at least checking) that a linear form is separating or not.

1.5 Contributions

Our main contribution is to show how to certify a separating linear form and read (say compute with a closed formula) a rational parameterization from some lexicographic Gröbner basis. In particular, a parameterization and the certification of its associated separating form t can be computed with an overhead of $\tilde{O}(nD^2)$ arithmetic operations in the ground field compared to the computation of a lexicographic Gröbner basis from any other Gröbner basis by change of ordering.

It should be stressed that, in the general case, the use of a separating linear form destroys the sparsity of the main multiplication matrix so that we do not find interesting to keep having a specific complexity setting to measure this sparsity.

In the case where $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ is cyclic, the overhead then becomes $\tilde{O}(nD)$ arithmetic operations in $\mathbb{K}$ and is thus negligible compared to the cost of FGLM algorithm, confirming that the rational parameterization can be obtained *for free* in that context, even with our general algorithm. It is also interesting to see that in the general case, whenever the FGLM engine is at least in a Las-Vegas setting (see [13]), the overhead for computing the parameterization still stays moderate.

In the case of systems with rational coefficients, when taking into account the bit operations instead of the arithmetic operations, things are not so simple due to the intermediate growth of coefficients related to the computation (or partial computation) of the lexicographic Gröbner basis. In particular, algorithms that provide the best complexity bounds are using computational technics based on the Chinese remainder theorem and their complexity studies are all in a Monte-Carlo setting. Also, in order to keep either a good control on the coefficient growth and to have a certification of the result (linear form and parameterization), one should still consider the ancestral algorithm from [20].

2 THE BIVARIATE CASE

In the case of two variables, the seminal ideas are the same as in [4], replacing the use of a subresultant sequence by the use of Gröbner basis together with some result on their specialization (see [15]).

Proposition 2.1. *Suppose that $I \subset \mathbb{K}[T, X_1]$ is zero dimensional and that $G = \left\{f(T), a_k(T, X_1) = \sum_{i=0}^{k} a_{k,i}(T)X_1^i, k = 1 \ldots n_1\right\}$, is a Gröbner basis of I for the lexicographic ordering such that $X_1 > T$, with f monic, allowing null polynomials in the list in order to keep simple notations. Denote by $\overline{f}$ the squarefree part of f.*

Define $f_0 = \overline{f}$ and for $k = 1, \ldots, n_1$, $f_k = \frac{f_{k-1}}{\gcd(f_{k-1}, a_{k,k})}$. Then,

- *T separates $V(I)$ iff*

$$(ka_{k,k})^{k-i}a_{k,i} = \begin{pmatrix} k \\ i \end{pmatrix} a_{k,k}a_{k,k-1}^{k-i} \bmod f_k,$$

  for every $k = 1, \ldots, n_1$ and $i = 0, \ldots, k-1$;
- *If T separates $V(I)$, then*

$$V(I) = \cup_{k=1}^{n_1} V(f_k, ka_{k,k}X_1 + a_{k,k-1} \bmod f_k).$$

Proof. According to [15], given any root α of f, if n_α is the first index between 1 and n_1 such that $a_{n_\alpha}(\alpha, X_1)$ is not identically zero, then $a_{n_\alpha}(\alpha, X_1) = \gcd(a_1(\alpha, X_1), \ldots, a_{n_1}(\alpha, X_1))$.

Also, if T separates $V(I)$, for each $\alpha \in V(f)$, there exists a unique β such that $(\alpha, \beta) \in V(I)$, so that $a_{n_\alpha}(\alpha, X_1) = a_{n_\alpha, n_\alpha}(\alpha)(X_1 - \beta)^{n_\alpha}$ and thus $a_{n_\alpha}(\alpha, X_1) = a_{n_\alpha,n_\alpha}(\alpha)\left(\sum_{i=0}^{n_\alpha}(-\beta)^{n_\alpha - i}\begin{pmatrix} n_\alpha \\ i \end{pmatrix} X_1^i\right)$. It is then easy to see that the roots of f_k are the factors of $f = f_0$ with roots α for which $n_\alpha = k$, and we then get that :

- T separates $\{(\alpha, \beta) \in V(I), \alpha \in V(f_k)\}$ iff , $\forall i = 0, \ldots, k-1$, $(ka_{k,k})^{k-i}a_{k,i} \equiv a_{k,k-1}^{k-i}\begin{pmatrix} k \\ i \end{pmatrix} a_{k,k} \bmod f_k$.
- $V(f_k, ka_{k,i}X_1 + a_{k,k-1} \bmod f_k) = \{(\alpha, \beta) \in V(I), \alpha \in V(f_k)\}$

To conclude, we just need to show that $\overline{f} = \prod_{k=1}^{n_1} f_k$ which is ensured by the fact that since I is zero-dimensional, a_{n_1,n_1} is a element of $\mathbb{K}$. □

The following corollary groups the parameterizations into a unique one to keep simple notations. In practice, one will better keep the decompositions for efficiency reasons.

Corollary 2.2. *Using the notations of Proposition 2.1, if T separates I, then $\{\overline{f}(T), h_{X_1}(T) = h_{X_1,1}(T)X_1 + h_{X_1,0}(T)$ $= \sum_{k=1}^{n_1}\left(ka_{k,k}(T)X_1 + a_{k,k-1}(T)\right)\prod_{i=0}^{k-1} f_i(T) \bmod \overline{f}(T)\}$ is a parameterization of $V(I)$.*

Proof. To prove this corollary, it is sufficient to remark that whenever $k < n_1$, both $a_{k,k}$ and $a_{k,k-1}$ cancel at each root of f_i, $i = k+1, \ldots, n_1$ so that $h_{X_1}(\alpha) = (ka_{k,k}(\alpha)X_1 + a_{k,k-1}(\alpha))\prod_{i=0}^{k-1} f_i(\alpha)$ $\forall \alpha \in V(f_k)$. □

Implementing the algorithm for checking the separating linear form in the bivariate case is as simple as verifying the condition of theorem 2.1 on each polynomial of the Gröbner basis. In Algorithm 2, we optimize the computation by computing the quotient $\frac{a_{k,i+1}}{a_{k,i}}$, which avoids to compute powers of polynomials.

Algorithm 2 Separation test in the bivariate setting

Require: A lexicographic Gröbner basis
1: $G_j = \{f(T), g_{j,1}(T, X_j), \ldots, g_{j,n_j}(T, X_j)\}$ with
2: $g_{j,k} = \sum_{i=0}^{k} a_{j,k,i}(T)x_j^i$ of $I \subseteq \mathbb{K}[T, X_j]$.
Ensure: *true* if T separates the zeroes of G_j, *false* otherwise
3: $f_{j,0} := \overline{f}$
4: **for** $k = 1, \ldots, n_j$ **do**
5: $\quad f_{j,k} := \frac{f_{j,k-1}}{\gcd(f_{j,k-1}a_{k,k})}$
6: $\quad$ **for** $i = 0, \ldots, k-1$ **do**
7: $\quad\quad$ **if** $\begin{pmatrix} \frac{k(k-i)}{i+1}a_{j,k,k}a_{j,k,i} \\ \neq \\ a_{j,k,k-1}a_{j,k,i+1} \bmod f_{j,k} \end{pmatrix}$ **then**
8: $\quad\quad\quad$ **return** false
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: **end for**
12: **return** true

We finally end this section with Algorithm 2 that computes the the bivariate parameterization accordingly to Corollary 2.2.

Algorithm 3 Rational parameterization in the bivariate setting

Require: A lexicographic Gröbner basis
1: $G_j = \{f(T), g_{j,1}(T, X_j), \ldots, g_{j,n_j}(T, X_j)\}$ with
2: $g_{j,k} = \sum_{i=0}^{k} a_{j,k,i}(T)x_j^i$ of $I \subseteq \mathbb{K}[T, X_j]$
3: T is known to be a separating form.
Ensure: $\{f_{X_j}\}$ such that $\{T, f_T(T), f_{X_j}\}$ is a rational parameterization of $V(I)$.
4: $f_{j,0} := \overline{f}$
5: $\rho := 1, h_{X_j,1} := 0, h_{X_j,0} := 0$
6: **for** $k = 1, \ldots, n_j$ **do**
7: $\quad f_{j,k} := \frac{f_{j,k-1}}{\gcd(f_{j,k-1}a_{k,k})}$
8: $\quad \rho := \rho \cdot f_{j,k}$
9: $\quad h_{X_j,1} := h_{X_j,1} + ia_{i,i}\rho \bmod f_{j,0}$
10: $\quad h_{X_j,0} := h_{X_j,0} + a_{i,i-1}\rho \bmod f_{j,0}$
11: **end for**
12: **return** (Algorithm 1 $(\{f(T), h_{X_j,1}(T)X_1 + h_{X_j,0}(T)\})$

3 THE GENERAL CASE

For the general case, the first remark is that if $t = \sum_{i=1}^{n} a_i X_i \in \mathbb{K}[X_1, \ldots, X_n]$, then $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ and $\frac{\mathbb{K}[T,X_1,\ldots,X_n]}{I+\langle T-t\rangle}$ are isomorphic as $\mathbb{K}$-vector spaces. Moreover, if G is a Gröbner basis of I for any ordering $>$, then $G_T = G \cup \{T - t\}$ is a Gröbner basis for the product ordering $(>, >_T)$ where $>_T$ is the canonical univariate ordering and where $T(>, >_T)X_i, i = 1..n$. A direct consequence is that a basis $\mathcal{B}$ for $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ which has been computed from G wrt $>$ is also a basis $\mathcal{B}'$ for $\frac{\mathbb{K}[T,X_1,\ldots,X_n]}{I_T=I+\langle T-(\sum_{i=1}^{n} u_i X_i)\rangle}$ wrt $(>, >_T)$.

In particular if $M_p^{\mathcal{B}}$ denotes the matrix of multiplication by a polynomial p in $\frac{\mathbb{Q}[X_1,\ldots,X_n]}{I}$ wrt a quotient basis $\mathcal{B}$, $M_T^{\mathcal{B}'} = \sum_{i=1}^{n} u_i M_{X_i}^{\mathcal{B}}$.

A second remark is that if $f(T)$ is the minimal polynomial of T in $\frac{\mathbb{K}[T,X_1,\ldots,X_n]}{I_T}$, then $f(T)$ is also the minimal polynomial of T in $\frac{\mathbb{K}[T,X_j]}{I_T \cap \mathbb{K}[T,X_j]}$, $j = 1..n$

Definition 3.1. Let $>_{T,j}$ be the lexicographic ordering on $\mathbb{K}[T, X_j]$ such that $X_j >_{T,j} T$.

The main result for the general case is a direct consequence of Corollary 2.2:

Theorem 3.2. *Let $I \subset \mathbb{K}[X_1, \ldots, X_n]$ be a zero-dimensional ideal, $T = t_1X_1 + \ldots + t_nX_n$ be a linear form with coefficients in $\mathbb{K}$ and $I_T = I + \langle T - t_1X_1 - \ldots - t_nX_n\rangle \subset \mathbb{K}[T, X_1, \ldots, X_n]$.*

For each $j = 1 \ldots n$, given $G_j = \{f_T, g_{j,1} \ldots, g_{j,n_j}\}$ with $g_{j,k} = \sum_{i=0}^{k} a_{j,k,i}(T)X_j^i$, a Gröbner basis of $I_{T,j} = I_T \cap \mathbb{Q}[T, X_j]$ for an elimination ordering $>_{T,j}$ such that $T < X_j$, we define $f_{T,j,0} = \overline{f_T}$, the monic squarefree part of f_T and for $k = 1 \ldots n_j$, the monic polynomials $f_{T,j,k} = \frac{f_{T,j,0}}{\prod_{i=1}^{k} \gcd(a_{j,i,i}, f_{T,j,i-1})}$. Then :

- *T separates the roots of $V(I_{T,j})$ iff, $\forall k = 1 \ldots n_j$,*

$$(ka_{j,k,k})^{k-i} a_{j,k,i} = \binom{k}{i} a_{j,k,k} a_{j,k,k-1}^{k-i} \bmod f_{T,j,k}$$

- *If T separates the roots of $V(I_{T,j})$, then*

$$\{f_T(T), h_{X_j}(T) = h_{X_j,1}(T)X_j + h_{X_j,0}(T) = \sum_{i=1}^{n_j} (ia_{j,i,i}X_j + a_{j,i,i-1}) \prod_{k=1}^{i-1} f_{T,j,k} \bmod \overline{f_T}\}$$

is a parameterization of $I_{T,j}$.

- *If T separates $V(I)$, denote by f_T the characteristic polynomial of T in $\frac{\mathbb{K}[X_1,\ldots,X_n]}{I}$ and set $f_{X_j} = -\tilde{f}_T' h_{X_j,0} h_{X_j,1}^{-1} \bmod \tilde{f}_T$. Then $\{T, f_T, f_{X_1}, \ldots, R_{X_n}\}$ is the RUR of $V(I)$ associated to T.*

Algorithm 4 Rational Parameterization Candidate

Require: $S_T = \{T - t, p_1, \ldots p_s\} \subseteq \mathbb{K}[T, X_1, \ldots, X_n]$, where t is a linear form in the X_i and $p_i \in \mathbb{K}[X_1, .., X_n]$;
1: A boolean *check*
Ensure: $\{T, f_T, f_{X_1}, \ldots, f_{X_n}\}$ a rational parameterization of the zeroes of $I = \langle p_1, \ldots p_s\rangle$ or *FAIL* is t is not separating.
2: If *check*$= false$ it is assumed that t is separating.
3: Compute a Gröbner basis G_T of S_T for any monomial ordering.
4: [FGLM 0] Compute from G the maximal l such that $\mathcal{T} = \{1, T, \ldots, T^{l-1}\}$ is free in $\frac{\mathbb{K}[T,X_1,\ldots,X_n]}{I}$, f_T the minimal polynomial of T modulo G_T and stor the row echelon form of the matrix $\{1, T, \ldots, T^{l-1}\}$.
5: $res := \{T, f_T\}$
6: **for** $i = 1, \ldots, n$ **do**
7: $\quad$ [FGLM 1] Compute from G_T, $\mathcal{T}$ and the row echelon form of $\mathcal{T}$ the lexicographic Gröbner basis G_i of $I \cap \mathbb{K}[T, X_i]$
8: $\quad$ **if** *check* and not(Algorithm 2(G_i)) **then**
9: $\quad\quad$ **return** FAIL
10: $\quad$ **end if**
11: $\quad res := res \cup$ Algorithm 3(G_i)
12: **end for**
13: **return** *res*

Let's now put everything together in order to propose a generic deterministic algorithm that computes a rational parameterization for any zero-dimensional ideal. The following Algorithm assumes that we have access to a Gröbner engine able to compute a Gröbner basis for any monomial ordering $>$ and to compute normal forms ($NF_>$ function in the sequel) wrt to this Gröbner basis. Algorithm 5 uses Lemma 2.1 from [20] to provide a finite familly of linear forms $\mathcal{L} = \{\sum_{j=1}^{n} i^{j-1}X_j, i = 0..\frac{(n-1)D(D-1)}{2}\}$ containing at least one element that separates the roots of the system.

Although Algorithm 5 consists in running up to $O(nD^2)$ times Algorithm 4, in practice few attempts are necessary, so that a randomized Las-Vegas variant (Algorithm 6) makes sense, replacing the loop over the elements of $\mathcal{L} = \{\sum_{j=1}^{n} i^{j-1}X_j, i = 0..\frac{(n-1)D(D-1)}{2}\}$ by some oracle proposing linear forms.

4 SOME COMPLEXITY RESULTS

If we consider systems with rational coefficients, one could use Algorithm 5 or 6 directly, but the intermediate computations would produce huge rationals, especially in the various lexicographic Gröbner bases. In that case, one should better use the algorithm

Algorithm 5 Generic parameterization - worst case

Require: $S = \{p_1, \dots p_s\} \subseteq \mathbb{K}[X_1, \dots, X_n]$
Ensure: $\{T, f_T, f_{X_1}, \dots, f_{X_n}\}$ a rational parameterization of the zeroes of $I = \langle p_1, \dots p_s \rangle$ if S is zero-dimensional, FAIL otherwise.
1: [Gröbner DRL] Compute G_{drl}, a Gröbner basis of S for the degree reverse lexicographic ordering $>_{drl}$
2: **if** G_{drl} is not zero-dimensional **then return** FAIL
3: **end if**
4: Let D be the dimension of $\frac{\mathbb{K}[X_1,\dots,X_n]}{\langle S \rangle}$
5: **for** $i = 0..\frac{(n-1)D(D-1)}{2}$ **do**
6: Compute $S_T = \{T - \sum_{j=1}^{n} i^{j-1} X_j, p_1, \dots, p_s\}$
7: Set $G_T = \{T - NF(\sum_{j=1}^{n} i^{j-1} X_j, G_{drl}), p_1, \dots, p_s\}$
the Gröbner basis of S_T for the product of ordering $(>_{drl}, >_T)$
8: rur =Algorithm 4(G_T,true)
9: **if** $rur \neq FAIL$ **then return** (rur)
10: **end if**
11: **end for**
12: **return** res

Algorithm 6 Generic parameterization - Las-Vegas

Require: $S = \{p_1, \dots p_s\} \subseteq \mathbb{K}[X_1, \dots, X_n]$, k an integer with default value 1
Ensure: $\{T, f_T, f_{X_1}, \dots, f_{X_n}\}$ a rational parameterization of the zeroes of $I = \langle p_1, \dots p_s \rangle$ if S is zero-dimensional, FAIL otherwise.
1: [Gröbner DRL] Compute G_{drl}, a Gröbner basis of S for the degree reverse lexicographic ordering $>_{drl}$
2: **if** G_{drl} is not zero-dimensional **then return** FAIL
3: **end if**
4: [Oracle separation] Chose $t = \sum_{i=1}^{n} a_i X_i$
5: Set $S_T := \{T - t, p_1, \dots, p_s\}$
6: Set $G_T = \{T - NF_>(t, G_{drl}), p_1, \dots, p_s\}$
the Gröbner basis of S_T for the product of ordering $(>_{drl}, >_T)$
7: rur =Algorithm 4(G_T,true)
8: **if** $rur \neq FAIL$ **then** Goto [Oracle separation]
9: **else return** res
10: **end if**

from [20] which also computes directly over the rationals in a Las-Vegas setting with a much better control of the size of coefficients in the intermediate computations.

Let's thus consider Monte-Carlo algorithms such as in most other algorithms computing parameterizations like [2], [17], [13], [6] or [16] for example. The strategy is straightforward : perform modular computations until being able to reconstruct a parameterization using the chinese remainder theorem coupled with a rational reconstruction (see [18]).

We are now going to show that the complexity is carried by *[Groebner DRL]*, *[FGLM-0]* and n times *[FGLM-1]*, showing that all the other parts take less than $O(nD^2)$ arithmetic operations in the worst case.

Lemma 4.1. *Let $I \subset \mathbb{K}[X_1, \dots, X_n]$ be a zero-dimensional ideal. Algorithm 1 calculates a RUR from a parameterisation of zeroes of $\sqrt{I}$ in $\tilde{O}(n\delta)$ arithmetic operations in $\mathbb{K}$, where δ denotes the degree of the minimal polynomial in $\frac{\mathbb{K}[X_1,\dots,X_n]}{I}$ of the used separating form.*

Algorithm 7 Parameterization over $\mathbb{Q}$ - Monte-Carlo

Require: $S = \{p_1, \dots p_s\} \in \mathbb{K}[X_1, \dots, X_n]$, k an integer with default value 31
Ensure: $\{T, f_T, f_{X_1}, \dots, f_{X_n}\}$ a rational parameterization of the zeroes of $I = \langle p_1, \dots p_s \rangle$ if S is zero-dimensional, FAIL otherwise.
1: Set a prime number $\pi_0 \leq 2^k$]
2: $i := 0$
3: $\{t, f_t, f_1, .., f_n\} := rur_0$:=Algorithm 6 ($S$ modulo π_0)
4: **while** [Rational Reconstruction]
5: rur:=RatRecons($rur_0, \dots, rur_i$)= $FAIL$ **do**
6: $i := i + 1$
7: $\pi_i = previousPrime(\pi_{i-1})$
8: rur_i=Algorithm 4 ($S \cup \{T - t\}$ modulo π_i,false)
9: **end while**
10: **return** rur

Proof. Firstly we need to calculate $\overline{f}$, which takes (at most) $\tilde{O}(\delta)$ operations on $\mathbb{K}$, and for each $j = 1, \dots, n$ there are 2 modular multiplication and a modular inversion with degree l polynomials giving $\tilde{O}(n\delta)$ arithmetic operations in $\mathbb{K}$. □

Lemma 4.2. *Denote by $G_j = \{f(T), g_{j,1}(T, X_j), \dots, g_{j,n_j}(T, X_j)\}$ the lexicographic Gröbner basis of $\langle G_j \rangle$ for the monomial ordering $T < X_j$ and $\sharp V(I) \leq \delta \leq D$ the degree of f*

- *Algorithm 2 verifies whether or not T separates the zeroes of G_j in at most $\tilde{O}((D-\delta)^2\delta)$ operations in $\mathbb{K}$, which is dominated by $\tilde{O}(D^2)$ in any case.*
- *Algorithm 3 returns a parameterization of the solutions in $\tilde{O}((D-\delta)\delta)$ operations in $\mathbb{K}$.*

Proof. Note that n_j is necessarily less than $D - \delta$. Calculating the expression in line 7 of Algorithm 2 involves twice the modular multiplication of polynomials of degree at most δ, say $\tilde{O}(\delta)$ operations in $\mathbb{K}$. Since we are looping $k = 1, \dots, n_j$ and $i = 0, \dots, k-1$ gives us a total of $\tilde{O}\left(\delta n_j^2\right) \leq \tilde{O}((D-\delta)^2\delta)$ operations in $\mathbb{K}$.

Calculating the parameterization using Algorithm 3 is done by a loop such that at each i one computes $f_{k,j}$ using a gcd and a modular reduction, one multiplies the product $\rho = \prod_{k=1}^{i-2} f_{j,k} \bmod \overline{f_T}$ by $f_{j,i-1}$, multiply that by $ia_{i,i}$ and by $a_{i,i-1}$ respectively, add theses polynomials to $h_{X_j,1}$ and $h_{X_j,1}$ respectively and reduce the results by $\bar{f}$. As all the involved polynomials have degree $\geq \delta$, that summing over all $i = 1, \dots, n_j$ gives $\tilde{O}((D-\delta)\delta)$ operations in $\mathbb{K}$. □

Combining the above results, one then gets :

Proposition 4.3. *Let $I \subset \mathbb{K}[X_1, \dots, X_n]$, t a linear form in the X_i and $I_T = I + \langle T - t \rangle$. If $t_{Grob}(I_T)$ denotes the number of operations in $\mathbb{K}$ computing a Gröbner basis for I, $t_{FGLM-0}(I_T)$ the one for computing the minimal polynomial of T in $\frac{K[X_1,\dots,X_n]}{I}$ and $t_{FGLM-1}(I \cap \mathbb{K}[T, X_j])$ the one for computing a lexicographic Gröbner basis for $I \cap \mathbb{K}[T, X_j]$, then Algorithm 6 returns a RUR for $\sqrt{I}$ in $\tilde{O}\left(t_{Grob} + t_{FGLM-0} + n(t_{FGLM-1} + D^2)\right)$ arithmetic operations in $\mathbb{K}$, or fails in the same complexity.*

Computing the elimination Gröbner bases is an implementation of part of the FGLM algorithm [12, 13]: at first we are obligated to

calculate the minimal polynomial f of T, but after that we can get away at each i with calculating only the polynomials in a lexicographic Gröbner basis G'_i for I_T that involve T and X_i, since we're interested not in G'_i but rather in $G'_i \cap \mathbb{K}[T, X_i]$ which by the Elimination Theorem [8, Theorem 3.1.2] is a lexicographic Gröbner basis for $I_T \cap \mathbb{K}[T, X_i]$. The FGLM algorithm is also the part where it is useful to know the multiplication matrices of the variables in the quotient $\frac{\mathbb{K}[T,X_1,\ldots,X_n]}{I_T}$, which by [12, Proposition 3.1] take $O(nD^3)$ arithmetic operations to calculate in the worst case when not taking into account the sparsity of the problem, $O(D^2 \sum_{i=1}^{n} M_i)$ if M_i denotes the number of dense columns of the matrix of multiplication by X_i by [13, Proposition 5.1].

We do not pay to much credit to the sparsity of the matrices in our complexity analysis, since, in the worst case, one will chose a linear form that will lower it dramatically.

Note that for each elimination Gröbner basis, we start calculating, no prior knowledge of the other elimination Gröbner bases is needed, and there is no part of this computation that is going to be repeated in general, so all these bases can be calculated in parallel.

5 EXPERIMENTAL RESULTS

In order to validate our result, we have made two implementations of our algorithm 7 for computing parameterizations of polynomial systems with rational coefficients :

- A simple implementation of 300 lines of Maple code that uses the available functions for computing Gröbner bases and running FGLM algorithm.
- An implementation of 1400 lines of Julia [15] code based on the package Groebner.jl (see [24]) for the computations of Gröbner bases for the degree lexicographic ordering. In this code, we we have implemented a basic FGLM algorithm using a naive handmade set of linear algebra functions, without optimizations for sparse data and following a schoolbook Gaussian reduction.

Both implementations are fully certified (including the separating linear form) when the coefficients belong to $\frac{\mathbb{Z}}{p\mathbb{Z}}$ but fall in the class of Monte-Carlo algorithms when the coefficients belong to $\mathbb{Q}$. Note that parameterizations can be checked a posteriori in the case of ideals that generate cyclic quotient algebras, for example using the strategy from [19], but there is no obvious way to check them in the general case.

The source code of our implementations is available at https://gitlab.inria.fr/newrur/code/. These prototypes are probably not bug-free, we propose them only to reproduce the experiments and, however, the possible bugs are mainly part of the probabilistic character of the implementation over the rationals.

The implementation in Julia uses the package Nemo.jl [14] for univariate polynomial arithmetic. We use multi-modular tracing to speed up the computation of the modular images of the Gröbner bases [21]. In our Gaussian elimination, we do not apply specific SIMD techniques, but rely on the compiler entirely.

We showcase the performance of our Maple and Julia implementations on a set of standard benchmark problems. We compare it to the state of the art implementation in msolve [2] accessed via the Julia package AlgebraicSolving.jl.

Table 1: Timings on a Macbook silicon M2 max

Name	Dim	Type	Maple rur	Julia rur	Julia msolve
Chandra 9	256	S	33	12	24
Chandra 10	512	S	250	106	374
Chandra 11	1024	S	2290	1153	6458
cp_3_6_2	720	S	137	110	138
cp_3_6_6	729	S	92	70	98
Eco 11	512	S	25	16	35
Eco 12	1024	S	268	191	516
Eco 13	2048	S	2862	2668	7697
Fab 4	1296	G	469	272	365
Henrion 6	720	S	20	15	95
Katsura 11	1024	S	257	110	189
Katsura 12	2048	S	3186	1846	2718
Noon 7	2173	S	1999	1692	1819
Reimer 6	576	C	10	5	35
Reimer 7	2880	C	2384	2074	29825
Schwarz 11	2048	C	1371	693	7246

The benchmark suite consists of zero-dimensional systems over the rational numbers. It includes systems in shape position, systems with a cyclic quotient algebra, as well as the most general case. The sources of benchmarks systems are available at https://gitlab.inria.fr/newrur/code/Data/Systems. We use two different computers to run the benchmarks: a Macbook running an Apple M2 Max and a PC Intel i9-13900. All computations are run using the packages *out of the box* on a single thread.

- Maple 2023 settings : `kernelopts(numcpus = 1):`
- Julia 1.10 with `Nemo v0.41.3`, `AlgebraicSolving v0.4.8`
- Macbook (MacOS) : `RAM : 64 GB CPU: 12 × Apple M2 Max`
- PC (Linux) : `RAM : 128 GB CPU: 32 × 13th Gen i9-13900`

We present the results in Tables 1 and 2. For each system, we report the type of the system and the dimension of the quotient algebra, the running time in seconds of our Maple and Julia implementations, and the running time of AlgebraicSolving.jl. Julia uses just-in-time compilation, and we do not include the compilation time in the final timings.

In the timing tables, we use the following abbreviations :

- Dim : dimension of the quotient ring
- Type : kind of the ideal
  - S : the ideal is in shape position
  - G : the ideal has no particular property
  - C : the quotient algebra is cyclic
- Maple rur : Maple implementation of our algorithm
- Julia rur : Julia implementation of our algorithm
- Julia msolve : AlgebraicSolving.jl Julia package accessing msolve.
- CRASH : the computation crashed (several attempts)

We observe that the performance is similar across the two tested machines, excepted for some examples of high dimension when using the implementation of our algorithm in Julia (see for example *Eco 13* in the above examples). We have observed some cache

Table 2: Timings on a PC corei9 13th gen

Name	Dim	Type	Maple rur	Julia rur	Julia msolve
Chandra 9	256	S	27	10	21
Chandra 10	512	S	207	85	322
Chandra 11	1024	S	1950	910	5916
cp_3_6_2	720	S	124	90	111
cp_3_6_6	729	S	86	57	70
Eco 11	512	S	20	14	29
Eco 12	1024	S	224	161	472
Eco 13	2048	S	2804	2177	7402
Fab 4	1296	G	414	210	CRASH
Henrion 6	720	S	18	12	75
Katsura 11	1024	S	218	101	152
Katsura 12	2048	S	2932	1604	2614
Noon 7	2173	S	1750	1339	1733
Reimer 6	576	C	9	5	31
Reimer 7	2880	C	2021	1864	27849
Schwarz 11	2048	C	1116	527	CRASH

phenomena during the experiments, which might explain the observation.

We also observe that the maximum ratio of computation times between the two implementations of our algorithm is quite low ($\frac{Maple-rur}{Julia-rur} < 2.5$) and that, interestingly, our short Maple implementation is competitive on all examples.

We finally observe that the Julia implementation of our algorithm outperforms $Julia - msolve$ systematically on this benchmark suite. In particular, on examples that are not in shape position, $Julia - msolve$ seems to suffer quite a lot. This is probably mainly due to the fact that our implementation uses its function for checking the separating element in order to provide better linear forms involving smaller coefficients and involving less variables which lead to smaller outputs which are easier to compute.

6 CONCLUSION

In this article, we propose a new and (very) simple algorithm for computing a rational parameterization of zero-dimensional systems which is conceptually deterministic as those in [20] or [25].

The algorithm can be implemented in few lines of code (300 in Maple) if a Gröbner engine as well as an implementation of FGLM algorithm can be reused in order to provide [FGLM-0] and [FGLM-1] easily for implementing Algorithm 4, which is the heart of the computation. In our implementation in Julia, we used a basic dense Gaussian elimination to implement [FGLM-0] and [FGLM-1], and we coupled this with a straigthforward Monte-Carlo strategy based on the chinese remainder theorem for systems with coefficients in $\mathbb{Q}$, using our deterministic algorithm on $\frac{\mathbb{Z}}{p\mathbb{Z}}$ to compute the residues.

When working with systems over the rationals in Algorithm 7, the computation of the residues is readily run on multiple threads, which we do not yet exploit. In the case when $\mathbb{K} = \frac{\mathbb{Z}}{p\mathbb{Z}}$, a more fine-grained parallelism can be applied by computing the bivariate lexicographical Gröbner bases in parallel. At the same time, the computation of the minimal polynomial is not easily parallelized, since obtaining the normal form requires the knowledge of the previous ones.

For having fair comparisons with other strategies, we currently reconstruct a unique RUR, and we do not exploit the natural splits induced by Algorithm 2 and 3 to take advantage of the situations where the system is not well separated and/or in the presence of roots of different multiplicities. Due to the uniqueness of the RUR, such splits might represent another way of computing the polynomials of the algorithm from [17].

Last but not least, as we were inspired by the work done on bivariate systems in [4] that did exploit univariate subresultants properties, one should investigate the relation with multivariate subresultants as in [7].

REFERENCES

[1] M.-E. Alonso, E. Becker, M. F. Roy, and T. Wörmann. Zeros, multiplicities, and idempotents for zero-dimensional systems. In Laureano González-Vega and Tomás Recio, editors, *Algorithms in Algebraic Geometry and Applications*, pages 1–15. Basel, 1996. Birkhäuser Basel.

[2] J. Berthomieu, C. Eder, M. Safey El Din. msolve: A Library for Solving Polynomial Systems. *ISSAC '21*, 51–58, 2021.

[15] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98. 2017.

[4] Y. Bouzidi, S. Lazard, M. Pouget and F. Rouillier. Separating linear forms and Rational Univariate Representations of bivariate systems. *Journal of Symbolic COmputation*, 68:84–119. 2015. hal-00977671

[25] C. Brand and M. Sagraloff. On the Complexity of Solving Zero-Dimensional Polynomial Systems via Projection. *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, p. 151-158, 2016*

[6] A. Bostan,B. Salvy and E. Schost Fast algorithms for zero-dimensional polynomial systems using duality. *Appl. Algebra Engrg. Comm. Comput. 14, 239–272, 2023.*

[7] David A. Cox and Carlos D'Andrea Subresultants and the Shape Lemma *Math. Comp. 92 (2023), 2355-2379*

[8] D. Cox, J. Little, D. o'Shea. Ideals, varieties and algorithms. *Springer*, 1991.

[9] D. Cox, J. Little, D. o'Shea. Using Algebraic Geometry. *Springer*, 1998.

[24] A. Demin and S. Gowda Groebner.jl: A package for Gröbner bases computations in Julia *Arxiv preprint 2304.06935v2.*

[11] I. Z. Emiris, B. Mourrain, and E. Tsigaridas. The DMM bound: multivariate (aggregate) separation bounds. In S. Watt, editor, *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 243–250. Munich, Germany, Jul 2010. ACM. Best paper award.

[12] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[13] J.-C. Faugère and C. Mou. Sparse fglm algorithms. *Journal of Symbolic Computation*, 80:538–569, 2017.

[14] C. Fieker, W. Hart, T. Hofmann, F. Johansson. Nemo/Hecke: Computer Algebra and Number Theory Packages for the Julia Programming Language. *ISSAC '17*, 157–164. 2017.

[15] P. Gianni. Properties of gröbner bases under specializations. In James H. Davenport, editor, *Eurocal '87*, pages 293–297. Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[16] M. Giusti, G. Lecerf, and B. Salvy. A gröbner free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211, 2001.

[17] S.-G. Hyun, V. Neiger, H. Rahkooy and E. Schost. Block-Krylov techniques in the context of sparse-FGLM algorithms *Journal of Symbolic Computation, vol. 98, p. 163-191, 2020*

[18] M. Monagan. Maximal quotient rational reconstruction: an almost optimal algorithm for rational reconstruction, *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, p. 243–249*

[19] B. Parisse Certifying a probabilistic parallel modular algorithm for rational univariate representation *Arxiv preprint 2106.10912v3.*

[20] F. Rouillier. Solving Zero-Dimensional Systems through the Rational Univariate Representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, May 1999. Article dans revue scientifique avec comité de lecture.

[21] C. Traverso Gröbner trace algorithms. *Symbolic and Algebraic Computation*, 125–138, 1989.