HAL
archives-ouvertes.fr

Block-Krylov techniques in the context of sparse-FGLM algorithms

Seung Gyu Hyun, Vincent Neiger, Hamid Rahkooy, Éric Schost

▶ **To cite this version:**

Seung Gyu Hyun, Vincent Neiger, Hamid Rahkooy, Éric Schost. Block-Krylov techniques in the context of sparse-FGLM algorithms. 2017. <hal-01661690>

HAL Id: hal-01661690
https://hal-unilim.archives-ouvertes.fr/hal-01661690

Submitted on 12 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Block-Krylov techniques in the context of sparse-FGLM algorithms

Seung Gyu Hyun[1] Vincent Neiger[2] Hamid Rahkooy[1]

Éric Schost[1]

[1]Cheriton School of Computer Science, University of Waterloo

[2]Univ. Limoges, CNRS, XLIM, UMR 7252, F-87000 Limoges, France

December 12, 2017

Abstract

Consider a zero-dimensional ideal I in $\mathbb{K}[X_1, \ldots, X_n]$. Inspired by Faugère and Mou's Sparse FGLM algorithm, we use Krylov sequences based on multiplication matrices of I in order to compute a description of its zero set by means of univariate polynomials.

Steel recently showed how to use Coppersmith's block-Wiedemann algorithm in this context; he describes an algorithm that can be easily parallelized, but only computes parts of the output in this manner. Using generating series expressions going back to work of Bostan, Salvy, and Schost, we show how to compute the entire output for a small overhead, without making any assumption on the ideal I other than it having dimension zero. We then propose a refinement of this idea that partially avoids the introduction of a generic linear form. We comment on experimental results obtained by an implementation based on the C++ libraries LinBox, Eigen and NTL.

1 Introduction

Computing the Gröbner basis of an ideal with respect to a given term ordering is an essential step in solving systems of polynomials. Certain term orderings, such as the degree reverse lexicographical ordering (*degrevlex*), tend to make the computation of the Gröbner basis faster. This has been observed empirically since the 1980's and is now supported by theoretical results, at least for some "nice" families of inputs, such as complete intersections or certain determinantal systems [11, 16, 2]. On the other hand, other orderings, such as the lexicographical ordering (*lex*), make it easier to find the coordinates of the solutions, or to perform arithmetic operations in the corresponding residue class ring. For instance, for

a zero-dimensional radical ideal I in generic coordinates in $\mathbb{K}[X_1, \ldots, X_n]$, for some field $\mathbb{K}$, the Gröbner basis of I for the lexicographic order with $X_1 > \cdots > X_n$ has the form

$$\{X_1 - R_1(X_n), \ldots, X_{n-1} - R_{n-1}(X_n), R_n(X_n)\}, \tag{1}$$

with all R_i's, for $i = 1, \ldots, n-1$, of degree less than $\deg(R_n)$ (and R_n squarefree); this is known as the *shape lemma* [19]. The points in the variety $V(I) \subset \overline{\mathbb{K}}^n$ are then

$$\{(R_1(\tau), \ldots, R_{n-1}(\tau), \tau) \mid \tau \in \overline{\mathbb{K}} \text{ is a root of } R_n\}.$$

As a result, the standard approach to solve a zero-dimensional system by means of Gröbner basis algorithms is to first compute a Gröbner basis for a degree ordering and then convert it to a more readable output, such as a lexicographic basis. As pointed out in [15], the latter step, while of polynomial complexity, can now be a bottleneck in practice. This paper will thus focus on this step; in order to describe our contributions, we first discuss previous work on the question.

Let I be a zero-dimensional ideal in $\mathbb{K}[X_1, \ldots, X_n]$. As input, we assume that we know a monomial basis $\mathscr{B}$ of $\mathscr{Q} = \mathbb{K}[X_1, \ldots, X_n]/I$, together with the multiplication matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ of respectively $X_1, \ldots, X_n$ in this basis. We denote by D the degree of I, which is the vector space dimension of $\mathscr{Q}$. We should stress that starting from a degree Gröbner basis of I, computing the multiplication matrices efficiently is not a straightforward task. Faugère *et al.* showed how to do it in time $O(nD^3)$ in [14]; more recently, algorithms have been given with cost bound $O^{\tilde{}}(nD^\omega)$ [12, 13, 36], at least for some favorable families of inputs. Here, the notation $O^{\tilde{}}$ hides polylogarithmic factors and ω is a feasible exponent for matrix multiplication over a ring (commutative, with 1). While improving these results is an interesting question in itself, we will not address it in this paper.

Given such an input, including the multiplication matrices, the FGLM algorithm [14] computes the lexicographic Gröbner basis of I in $O(nD^3)$ operations in $\mathbb{K}$. While the algorithm has an obvious relation to linear algebra, lowering the runtime to $O^{\tilde{}}(nD^\omega)$ was only recently achieved [12, 13, 36].

Polynomials as in Eq. (1) form a very useful data structure, but there is no guarantee that the lexicographic Gröbner basis of I has such a shape (even in generic coordinates). When it does, we will say that I is in *shape position*; some sufficient conditions for being in shape position are detailed in [3]. As an alternative, one may then use Rouillier's Rational Univariate Representation algorithm [40] (see also [1, 4] for related considerations). The output is a description of the zero-set $V(I)$ by means of univariate rational functions

$$\left\{F(T) = 0, \quad X_1 = \frac{G_1(T)}{G(T)}, \ldots, X_n = \frac{G_n(T)}{G(T)}\right\}, \tag{2}$$

where the multiplicity of a root τ of F coincides with that of I at the corresponding point $(G_1(\tau)/G(\tau), \ldots, G_n(\tau)/G(\tau)) \in V(I)$. The fact that we use rational functions makes it possible to control precisely the bit-size of their coefficients, if working over $\mathbb{K} = \mathbb{Q}$.

The algorithms of [1, 4, 40] rely on *duality*, which will be at the core of our algorithms as well. Indeed, these algorithms compute sequences of values of the form $v_s = (\mathrm{trace}(X^s))_{s \geq 0}$

and $v_{s,i} = (\mathrm{trace}(X^s X_i))_{s\geq 0}$, where trace : $\mathscr{Q} \to \mathbb{K}$ is the trace form and $X = t_1X_1 + \cdots + t_nX_n$ is a $\mathbb{K}$-linear combination of the variables. From these values, one may then recover the output in Eq. (2) by means of structured linear algebra calculations.

A drawback of this approach is that we need to know the trace of all elements of the basis $\mathscr{B}$; while feasible in polynomial time, this is by no means straightforward. In [8], Bostan *et al.* introduced randomization to alleviate this issue. They show that computing values such as $\ell(X^s)$ and $\ell(X^s X_i)$, where X is as above and ℓ is a random $\mathbb{K}$-linear form $\mathscr{Q} \to \mathbb{K}$, allows one to deduce a description of $V(I)$ of the form

$$\{Q(T) = 0, \quad X_1 = V_1(T), \ldots, X_n = V_n(T)\}, \tag{3}$$

where Q is a monic squarefree polynomial in $\mathbb{K}[T]$ and V_i is in $\mathbb{K}[T]$ of degree less than $\deg(Q)$ for all i. The tuple $((Q, V_1, \ldots, V_n), X)$ computed by such algorithms will be called a *zero-dimensional parametrization* of $V(I)$. In particular, it generally differs from the description in Eq. (2), since the latter keeps trace of the multiplicities of the solutions (the algorithm in [8] actually computes the *nil-indices* of the solutions). Remark that starting from such an output, we can reconstruct the local structure of I at its roots, using algorithms from [32], [35] or [37].

The most costly part of the algorithm of [8] is the computation of the values $\ell(X^s)$ and $\ell(X^s X_i)$; the rest essentially boils down to applying the Berlekamp-Massey algorithm and univariate polynomial arithmetic. In [15], Faugère and Mou pointed out that the multiplication matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ can be expected to be sparse; for generic inputs, they gave precise estimates on the sparsity of these matrices, assuming the validity of a conjecture by Moreno-Socías [33]. On this basis, they designed several forms of *sparse FGLM* algorithms. For instance, if I is in shape position, the algorithms in [15] recover its lexicographic basis, which is as in (1), by also considering values of a linear form $\ell : \mathscr{Q} \to \mathbb{K}$. For less favorable inputs, the algorithm falls back on the Berlekamp-Massey-Sakata algorithm [41] or plain FGLM.

The ideas at play in these algorithms are essentially based on Krylov subspace methods, using projections and Berlekamp-Massey techniques. These techniques have also been widely used in integer factorization or discrete logarithm calculations, going back to [30]. To easily parallelize the bottleneck of the algorithm, it is then natural to adapt to our situation the block version of such algorithms, as pioneered by Coppersmith [10] in the context of integer factorization. This was already discussed by Steel in [45], where he showed how to compute the analogue of the polynomial Q in Eq. (3) using such techniques. In that reference, one is only interested in the solutions in the base field $\mathbb{K}$ ($\mathbb{K}$ being a finite field in that context): the algorithm computes the roots of Q in $\mathbb{K}$ and substitutes them in the input system, before computing a Gröbner basis in $n-1$ variables for each of them.

Our first contribution is to give a block version of the algorithm in [8] that extends the approach introduced in [45] to compute all polynomials in Eq. (3) for essentially the same cost as the computation of Q. More precisely, the bottleneck for the algorithm of [45] is the computation of a block-Krylov sequence; we show that once this sequence has been computed, not only Q but also all other polynomials in the zero-dimensional parametrization

can be efficiently obtained. Compared with the algorithms of [15], a notable difference is that our algorithm deals with any ideal zero-dimensional I (for instance, we do not require shape position), but the base field must have high enough characteristic (and our output is somewhat weaker than a Gröbner basis, since multiplicities are not computed). While we focus on the case where the multiplication matrices are sparse, we also give a cost analysis for the case of dense multiplication matrices.

Our second contribution is a refinement of our first algorithm, where we try to avoid computation with a generic linear form $X = t_1X_1 + \cdots + t_nX_n$ to the extent possible (this is motivated by the fact that the multiplication matrix of X may be denser than those of the variables X_i). The algorithm first computes zero-dimensional parametrization of a subset of $V(I)$ for which we can take X equal to (say) X_1, and falls back on the previous approach for the residual part; if the former set has large cardinality, this is expected to provide a speed-up over the generalist algorithm.

Our algorithms have been implemented in C++ on the basis of LinBox [22], Eigen [23] and NTL [42]; an experimental implementation can be found at `https://git.uwaterloo.ca/sghyun/Block-sparse-FGLM`.

The paper is organized as follows. The next section mainly reviews known results on scalar and matrix recurrent sequences, and introduces a simple but useful algorithm to compute a so-called *scalar numerator* for such sequences. Section 3 describes sequences that arise in the context of FGLM-like algorithms; we prove slightly more precise versions of results from [8] that will be used throughout. The main algorithm is given in Section 4, and the refinement mentioned above is in Section 5.

Complexity model. We measure the cost of our algorithms by counting basic operations in $\mathbb{K}$ at unit cost. Most algorithms are randomized; they involve the choice of a vector $\gamma \in \mathbb{K}^S$ of field elements, for an integer S that depends on the size of our input, and success is guaranteed if the vector γ avoids a hypersurface of the parameter space $\mathbb{K}^S$.

Suppose that the input ideal I is generated by polynomials $F_1, \ldots, F_t$. Given a zero-dimensional parametrization $((Q, V_1, \ldots, V_n), X)$ computed by our algorithms, one can always evaluate $F_1, \ldots, F_t$ at $X_1 = V_1, \ldots, X_n = V_n$, doing all computations modulo Q. This allows us to verify whether the output describes a subset of $V(F_1, \ldots, F_t)$, but not whether we have found all solutions. If $\deg(Q)$ coincides with the dimension of $\mathscr{Q} = \mathbb{K}[X_1, \ldots, X_n]/I$, we can infer that we have all solutions (and that I is radical), so our output is correct.

In what follows, we assume $\omega > 2$, in order to simplify a few cost estimates. We use a time function $d \mapsto \mathsf{M}(d)$ for the cost of univariate polynomial multiplication over $\mathbb{K}$, for which we assume the super-linearity properties of [18, Section 8.4]. Then, two $m \times m$ matrices over $\mathbb{K}[T]$ whose degree is less than d can be multiplied using $O(m^\omega \mathsf{M}(d))$ operations in $\mathbb{K}$.

Acknowledgments. We wish to thank Chenqi Mou and Dave Saunders for useful discussions. This research was partially supported by Schost's NSERC Discovery Grant.

2 Linearly recurrent sequences

This section starts with a review of known facts on linearly recurrent sequences; we first discuss the scalar case, and then we show how the ideas carry over to matrix sequences.

The main results we will need from the first three subsections are Corollary 2.10 and Theorem 2.11, which give cost estimates for the computation of a *minimal matrix generator* of a linearly recurrent matrix sequence, as well as a degree bound for such generators, when the matrices we consider are obtained from a Krylov sequence. These results are for the most part not new (see [49, 50, 27, 47]), but the cost analysis we give uses results not available when those references were first written. The fourth and last subsection presents a useful result for our main algorithm that allows us to compute a "numerator" for a scalar sequence from a similar object obtained for a matrix sequence.

2.1 Scalar sequences

Let $\mathbb{K}$ be a field and consider a sequence $\mathcal{L} = (\ell_s)_{s\geq 0} \in \mathbb{K}^{\mathbb{N}}$. We say that a degree d polynomial $P = p_0 + \cdots + p_d T^d \in \mathbb{K}[T]$ *cancels* the sequence $\mathcal{L}$ if $p_0\ell_s + \cdots + p_d\ell_{s+d} = 0$ for all $s \geq 0$. The sequence $\mathcal{L}$ is *linearly recurrent* if there exists a non-zero polynomial that cancels it. The *minimal polynomial* of a linearly recurrent sequence $\mathcal{L} = (\ell_s)_{s\geq 0}$ is the monic polynomial of lowest degree that cancels it; the *order* of $\mathcal{L}$ is the degree of this polynomial P.

One can rephrase these properties in terms of the generating series $S = \sum_{s\geq 0} \ell_s T^s \in \mathbb{K}[[T]]$ associated to $\mathcal{L}$. Then, P cancels $\mathcal{L}$ if and only if $\mathrm{rev}(P)S$ is a polynomial, where $\mathrm{rev}(P) = T^d P(1/T)$; in this case, $\mathrm{rev}(P)S$ must have degree less than d. The sequence $\mathcal{L}$ is linearly recurrent if and only if there exist polynomials A, B in $\mathbb{K}[T]$ such that $S = A/B$; these polynomials are unique if we assume $\gcd(A, B) = 1$ and $B(0) = 1$. Given these A and B, the minimal polynomial of $\mathcal{L}$ is $P = T^{\max(\deg(A)+1,\deg(B))}B(1/T)$.

It is often easier to work with a closed form that has the actual minimal polynomial as its denominator, rather than its reverse; this is done by working with generating series in the variable $1/T$. Let $Z = \sum_{s\geq 0} \ell_s/T^{s+1}$ and P be any polynomial; then, P cancels the sequence $\mathcal{L}$ if and only if $Q = PZ$ is a polynomial, in which case Q must have degree less than $\deg(P)$. Using generating series in $1/T$, the minimal polynomial of $\mathcal{L}$ is thus the monic polynomial P of lowest degree for which there exists $Q \in \mathbb{K}[T]$ such that $Z = Q/P$.

In particular, given a sequence $\mathcal{L} = (\ell_s)_{s\geq 0}$ and a polynomial that cancels it, there is a well-defined notion of associated numerator. This is formalized in the next definition:

Definition 2.1. *Let $\mathcal{L} = (\ell_s)_{s\geq 0} \in \mathbb{K}^{\mathbb{N}}$ be a sequence and P be a polynomial that cancels $\mathcal{L}$. Then, the* numerator *of $\mathcal{L}$ with respect to P is denoted by $\Omega(\mathcal{L}, P)$ and defined as*

$$\Omega(\mathcal{L}, P) = PZ, \quad \text{where} \quad Z = \sum_{s\geq 0} \frac{\ell_s}{T^{s+1}}.$$

In view of the discussion above, $\Omega(\mathcal{L}, P)$ is a polynomial of degree less than $\deg(P)$.

Remark 2.2.

- We may write this definition using expressions in variable T instead of $1/T$ as well: given $\mathcal{L} = (\ell_s)_{s\geq 0}$ and P as above, we recover $\Omega(\mathcal{L}, P)$ by considering $S = \sum_{s\geq 0} \ell_s T^s$ and computing $A = \mathrm{rev}(P)S$, with $\mathrm{rev}(P) = T^d P(1/T)$ and $d = \deg(P)$; then
$$\Omega(\mathcal{L}, P) = T^{d-1} A(1/T).$$
- We only need to know the first $d = \deg(P)$ coefficients $\ell_0, \ldots, \ell_{d-1}$ to compute $\Omega(\mathcal{L}, P)$. Explicitly, we have
$$\Omega(\mathcal{L}, P) = \left(P \sum_{s=0}^{d-1} \ell_{d-1-s} T^s \right) \text{ div } T^d,$$

where div denotes the quotient in the Euclidean division. □

Now, we give an example of this construction. Consider the Fibonacci sequence $\mathcal{L} = (1, 1, 2, 3, 5, 8, \ldots)$, which is linearly recurrent with minimal polynomial $P = T^2 - T - 1$, and define $S = \sum_{s\geq 0} \ell_s T^s$ and $\mathrm{rev}(P) = 1 - T - T^2$. Then, we can write S in closed form as
$$S = \frac{A}{\mathrm{rev}(P)} = \frac{1}{1 - T - T^2},$$
so that $\Omega(\mathcal{L}, P) = T^{2-1} \cdot 1 = T$. Equivalently, defining $Z = \sum_{s\geq 0} \ell_s / T^{s+1}$, we recover
$$\Omega(\mathcal{L}, P) = (T^2 - T - 1)\left(\frac{1}{T} + \frac{1}{T^2} + \frac{2}{T^3} + \frac{3}{T^4} + \cdots \right) = T.$$

The numerators thus defined are related to the Lagrange interpolation polynomials; this will explain why they play an important role in our main algorithm. Explicitly, suppose that $\mathcal{L} = (a_1^s + \cdots + a_n^s)_{s\geq 0}$ for some pairwise distinct elements $a_1, \ldots, a_n \in \mathbb{K}$. Then, its minimal polynomial is $P = \prod_{i=1}^n (T - a_i)$, and the numerator $\Omega(\mathcal{L}, P)$ is $\sum_{i=1}^n \prod_{i' \neq i} (T - a_{i'})$.

Let us go back to the general case of a sequence $\mathcal{L} = (\ell_s)_{s\geq 0} \in \mathbb{K}^{\mathbb{N}}$. In terms of complexity, assuming that $\mathcal{L}$ is known to have order at most d and that we are given its first $2d$ terms, we can recover its minimal polynomial P by means of the Berlekamp-Massey algorithm, or of Euclid's algorithm applied to rational function reconstruction; this can be done in $O(\mathsf{M}(d)\log(d))$ operations in $\mathbb{K}$ [9]. Given P and $\ell_0, \ldots, \ell_{\deg(P)-1}$, we can deduce $\Omega(\mathcal{L}, P)$ for the cost $\mathsf{M}(\deg(P))$ of one polynomial multiplication in degree $\deg(P)$.

2.2 Linearly recurrent matrix sequences

Next, we discuss the analogue of these ideas for matrix sequences; our main goal is to give a cost estimate for the computation of a suitable *matrix generator* for a matrix sequence $\boldsymbol{\mathcal{F}}$, obtained by means of recent algorithms for approximant bases. A similar discussion, but relying on the approximant basis algorithm in [5], can be found in [47, Chapter 4]. Note also that the latter reference uses the generating series in T while here we use that in $1/T$, thus obtaining the sought generator directly as a submatrix of the computed approximant basis.

We first define the notion of linear recurrence for matrix sequences over a field $\mathbb{K}$ as in [27, Section 3] or [47, Definition 4.2], hereby extending the above notions for scalar sequences.

Definition 2.3. *Let $\boldsymbol{\mathcal{F}} = (\boldsymbol{F}_s)_{s\geq 0} \subset \mathbb{K}^{m\times m'}$ be a matrix sequence. Then,*

- *a polynomial $P = p_0 + \cdots + p_d T^d \in \mathbb{K}[T]$ is a* scalar relation for $\boldsymbol{\mathcal{F}}$ *if the identity $p_0\boldsymbol{F}_s + \cdots + p_d\boldsymbol{F}_{s+d} = \boldsymbol{0}$ holds for all $s \geq 0$;*
- *a polynomial vector $\boldsymbol{p} = \boldsymbol{p}_0 + \cdots + \boldsymbol{p}_d T^d \in \mathbb{K}[T]^{1\times m}$ is a* (left, vector) relation for $\boldsymbol{\mathcal{F}}$ *if $\boldsymbol{p}_0\boldsymbol{F}_s + \cdots + \boldsymbol{p}_d\boldsymbol{F}_{s+d} = \boldsymbol{0}$ holds for all $s \geq 0$;*
- $\boldsymbol{\mathcal{F}}$ *is* linearly recurrent *if it admits a non-zero scalar relation.*

For designing efficient algorithms, it will be useful to rely on operations on polynomials, that is, truncated power series; hence the following characterization of vector relations.

Lemma 2.4. *Consider a matrix sequence $\boldsymbol{\mathcal{F}} = (\boldsymbol{F}_s)_{s\geq 0} \subset \mathbb{K}^{m\times m'}$ and its generating series $\boldsymbol{Z} = \sum_{s\geq 0} \boldsymbol{F}_s/T^{s+1} \in \mathbb{K}[\![T^{-1}]\!]^{m\times m'}$. Then, $\boldsymbol{p} \in \mathbb{K}[T]^{1\times m}$ is a vector relation for $\boldsymbol{\mathcal{F}}$ if and only if the entries of $\boldsymbol{q} = \boldsymbol{p}\boldsymbol{Z}$ are in $\mathbb{K}[T]$; furthermore, in this case,* $\deg(\boldsymbol{q}) < \deg(\boldsymbol{p})$.

Proof. Write $\boldsymbol{p} = \sum_{0\leq k\leq d} \boldsymbol{p}_k T^k$. For $s \geq 0$, the coefficient of $\boldsymbol{q}$ of degree $-s-1 < 0$ is $\sum_{0\leq k\leq d} \boldsymbol{p}_k\boldsymbol{F}_{s+k}$. Hence the equivalence, by definition of a relation. The degree comparison is clear since $\boldsymbol{Z}$ has only terms of negative degree. □

Concerning the algebraic structure of the set of vector relations, we have the following basic result, which can be found for example in [49, 27, 47].

Lemma 2.5. *The sequence $\boldsymbol{\mathcal{F}}$ is linearly recurrent if and only if the set of left vector relations for $\boldsymbol{\mathcal{F}}$ is a $\mathbb{K}[T]$-submodule of $\mathbb{K}[T]^{1\times m}$ of rank m.*

(Note however that in general a matrix sequence may admit nontrivial vector relations and have no scalar relation, and therefore not be linearly recurrent with the present definition; in this case the module of vector relations has rank less than m.)

Definition 2.6. *Let $\boldsymbol{\mathcal{F}} \subset \mathbb{K}^{m\times m'}$ be linearly recurrent. A* (left) matrix generator $\boldsymbol{P}$ *for $\boldsymbol{\mathcal{F}}$ is a matrix in $\mathbb{K}[T]^{m\times m}$ whose rows form a basis of the module of left vector relations for $\boldsymbol{\mathcal{F}}$. This basis is said to be*

- minimal *if $\boldsymbol{P}$ is row reduced [51, 25];*
- canonical *if $\boldsymbol{P}$ is in Popov form [38, 25].*

Note that the canonical generator is also a minimal generator. Besides, all matrix generators $\boldsymbol{P} \in \mathbb{K}[T]^{m\times m}$ have the same determinantal degree $\deg(\det(\boldsymbol{P}))$, which we denote by $\Delta(\boldsymbol{\mathcal{F}})$.

We now point out that matrix generators are denominators in some irreducible fraction description of the generating series of the sequence; this is a direct consequence of Lemmas 2.4 and 2.5 and of basic properties of polynomial matrices. As suggested in the previous subsection, working with generating series in $1/T$ turns out to be the most convenient choice here.

In what follows, for an $r \times s$ matrix $\boldsymbol{P}$ with entries in $\mathbb{K}[T]$, we denote by $\mathrm{rdeg}(\boldsymbol{P})$ its row degree, that is, the size-r vector of the degrees of its rows; similarly, we denote by $\mathrm{cdeg}(\boldsymbol{P})$ its column degree, which is a size-s vector.

Corollary 2.7. *Let $\boldsymbol{P}$ be a nonsingular matrix in $\mathbb{K}[T]^{m\times m}$. The rows of $\boldsymbol{P}$ are relations for a matrix sequence $\boldsymbol{\mathcal{F}} = (\boldsymbol{F}_s)_{s\geq 0} \subset \mathbb{K}^{m\times m'}$ if and only if the generating series $\boldsymbol{Z} = \sum_{s\geq 0} \boldsymbol{F}_s/T^{s+1} \in \mathbb{K}[\![T^{-1}]\!]^{m\times m'}$ can be written as a matrix fraction $\boldsymbol{Z} = \boldsymbol{P}^{-1}\boldsymbol{Q}$, with $\boldsymbol{Q} \in \mathbb{K}[T]^{m\times m'}$. In this case, we have $\operatorname{rdeg}(\boldsymbol{Q}) < \operatorname{rdeg}(\boldsymbol{P})$ termwise and $\deg(\det(\boldsymbol{P})) \geq \Delta(\boldsymbol{\mathcal{F}})$; furthermore, $\boldsymbol{P}$ is a matrix generator for $\boldsymbol{\mathcal{F}}$ if and only if $\deg(\det(\boldsymbol{P})) = \Delta(\boldsymbol{\mathcal{F}})$.*

Given a nonsingular matrix of relations $\boldsymbol{P}$ for $\boldsymbol{\mathcal{F}}$, we will thus write $\boldsymbol{\Omega}(\boldsymbol{\mathcal{F}}, \boldsymbol{P}) = \boldsymbol{P}\boldsymbol{Z} \in \mathbb{K}[T]^{m\times m'}$, generalizing Definition 2.1.

By the previous corollary, this is a polynomial matrix, whose ith row has degree less than the ith row of $\boldsymbol{P}$ for $1 \leq i \leq m$. As in the scalar case, if $\boldsymbol{P}$ has degree d, we only need to know $\boldsymbol{F}_0, \ldots, \boldsymbol{F}_{d-1}$ to recover $\boldsymbol{\Omega}(\boldsymbol{\mathcal{F}}, \boldsymbol{P})$ as

$$\boldsymbol{\Omega}(\boldsymbol{\mathcal{F}}, \boldsymbol{P}) = \left(\boldsymbol{P} \cdot \sum_{s=0}^{d-1} \boldsymbol{F}_{d-1-s} T^s \right) \operatorname{div} T^d, \tag{4}$$

where div T^d means we keep the quotient of each entry by T^d. The cost of computing it then depends on the cost of rectangular matrix multiplication in sizes (m, m) and (m, m'). For simplicity, we only give the two most useful cases: if $m' = m$, this takes $O(m^\omega \mathsf{M}(d))$ operations in $\mathbb{K}$; if $m' = 1$, the cost becomes $O(m^2\mathsf{M}(d))$.

In all the previous discussion, we remark that we may also consider vector relations operating on the right: in particular, Lemma 2.4 shows that if the sequence is linearly recurrent then these right relations form a submodule of $\mathbb{K}[T]^{m'\times 1}$ of rank m'. Thus, a linearly recurrent sequence also admits a canonical right generator.

Now, we focus on our algorithmic problem: given a linearly recurrent sequence, find a minimal matrix generator. We assume the availability of bounds (d_ℓ, d_r) on the degrees of the canonical left and right generators, which allow us to control the number of terms of the sequence we will access during the algorithm. Since taking the Popov form of a reduced matrix does not change the degree, any minimal left matrix generator $\boldsymbol{P}$ has the same degree $\deg(\boldsymbol{P})$ as the canonical left generator: thus, d_ℓ is also a bound on the degree of any minimal left generator. The same remark holds for d_r and minimal right generators. The bounds (d_ℓ, d_r) correspond to (γ_1, γ_2) in [47, Definitions 4.6 and 4.7] and (δ_l, δ_r) in [50, Section 4.2]. The next result is similar to [47, Theorem 4.5].

Lemma 2.8. *Let $\boldsymbol{\mathcal{F}} = (\boldsymbol{F}_s)_{s\geq 0} \subset \mathbb{K}^{m\times m'}$ be linearly recurrent and let $d_r \in \mathbb{N}$ be such that the canonical right matrix generator of $\boldsymbol{\mathcal{F}}$ has degree at most d_r. Then, a vector $\boldsymbol{p} = \boldsymbol{p}_0 + \cdots + \boldsymbol{p}_d T^d \in \mathbb{K}[T]^{1\times m}$ is a left relation for $\boldsymbol{\mathcal{F}}$ if and only if $\boldsymbol{p}_0\boldsymbol{F}_s + \cdots + \boldsymbol{p}_d\boldsymbol{F}_{s+d} = \boldsymbol{0}$ holds for $s \in \{0, \ldots, d_r - 1\}$.*

Proof. Since the canonical right generator $\boldsymbol{P} \in \mathbb{K}[T]^{m'\times m'}$ is in column Popov form, we have $\boldsymbol{P} = \boldsymbol{L}\operatorname{Diag}(T^{t_1}, \ldots, T^{t_{m'}}) - \boldsymbol{Q}$, where $\operatorname{cdeg}(\boldsymbol{Q}) < \operatorname{cdeg}(\boldsymbol{P}) = (t_1, \ldots, t_{m'})$ termwise and $\boldsymbol{L} \in \mathbb{K}^{m'\times m'}$ is unit upper triangular. Define the matrix $\boldsymbol{U} = \operatorname{Diag}(T^{d_r - t_1}, \ldots, T^{d_r - t_{m'}})\boldsymbol{L}^{-1}$, which is in $\mathbb{K}[T]^{m'\times m'}$ since $d_r \geq \deg(\boldsymbol{P}) = \max_j t_j$. Then, the columns of the right multiple $\boldsymbol{P}\boldsymbol{U} = T^{d_r}\boldsymbol{I}_{m'} - \boldsymbol{Q}\boldsymbol{U}$ are right relations for $\boldsymbol{\mathcal{F}}$, and we have $\deg(\boldsymbol{Q}\boldsymbol{U}) < d_r$. As a consequence, writing $\boldsymbol{Q}\boldsymbol{U} = \sum_{0\leq k<d_r} \boldsymbol{Q}_k T^k$, we have $\boldsymbol{F}_{s+d_r} = \sum_{0\leq k<d_r} \boldsymbol{F}_{s+k}\boldsymbol{Q}_k$ for all $s \geq 0$.

Assuming that $\boldsymbol{p}_0\boldsymbol{F}_s + \cdots + \boldsymbol{p}_d\boldsymbol{F}_{s+d} = \boldsymbol{0}$ holds for all $s \in \{0, \ldots, d_r - 1\}$, we prove by induction that this holds for all $s \in \mathbb{N}$. Let $s \geq d_r - 1$ and assume that this identity holds for all integers up to s. Then, the identity concluding the previous paragraph implies that

$$\begin{aligned}\sum_{0\leq k\leq d} \boldsymbol{p}_k\boldsymbol{F}_{s+1+k} &= \sum_{0\leq k\leq d} \boldsymbol{p}_k \left(\sum_{0\leq j<d_r} \boldsymbol{F}_{s+1+k-d_r+j}\boldsymbol{Q}_j \right) \\ &= \sum_{0\leq j<d_r} \underbrace{\left(\sum_{0\leq k\leq d} \boldsymbol{p}_k\boldsymbol{F}_{s+1-d_r+j+k} \right)}_{=0 \text{ since } s+1-d_r+j\leq s} \boldsymbol{Q}_j = \boldsymbol{0},\end{aligned}$$

and the proof is complete. □

The fast computation of matrix generators is usually handled via minimal approximant bases algorithms [49, 47, 21]; the next result gives the main idea behind this approach. This result is similar to [47, Theorem 4.6] (see also [47, Theorems 4.7 to 4.10]), using however the reversal of the input sequence rather than that of the output matrix generator.

We recall from [48, 5] that, given a matrix $\boldsymbol{F} \in \mathbb{K}[T]^{m\times m'}$ and an integer $d \in \mathbb{N}$, the set of *approximants for* $\boldsymbol{F}$ *at order* d is defined as

$$\mathcal{A}(\boldsymbol{F}, d) = \{\boldsymbol{p} \in \mathbb{K}[T]^{1\times m} \mid \boldsymbol{p}\boldsymbol{F} = \boldsymbol{0} \bmod T^d\}.$$

Then, the next theorem shows that relations for $\boldsymbol{\mathcal{F}}$ can be retrieved as subvectors of approximants at order about $d_\ell + d_r$ for a matrix involving the first $d_\ell + d_r$ entries of $\boldsymbol{\mathcal{F}}$.

Theorem 2.9. *Let $\boldsymbol{\mathcal{F}} = (\boldsymbol{F}_s)_{s\geq 0} \subset \mathbb{K}^{m\times m'}$ be linearly recurrent and let $(d_\ell, d_r) \in \mathbb{N}^2$ be such that the canonical left (resp. right) matrix generator of $\boldsymbol{\mathcal{F}}$ has degree $\leq d_\ell$ (resp. $\leq d_r$). For $d > 0$, define*

$$\boldsymbol{F} = \begin{bmatrix} \sum_{0\leq s<d} \boldsymbol{F}_s T^{d-s-1} \\ -\boldsymbol{I}_{m'} \end{bmatrix} \in \mathbb{K}[T]^{(m+m')\times m'}.$$

Suppose that $d \geq d_r + 1$ and let $\boldsymbol{B} \in \mathbb{K}[T]^{(m+m')\times(m+m')}$ be a basis of $\mathcal{A}(\boldsymbol{F}, d_\ell + d_r + 1)$. Then,

- *if $\boldsymbol{B}$ is in Popov form then its $m \times m$ leading principal submatrix is the canonical matrix generator for $\boldsymbol{\mathcal{F}}$;*
- *if $\boldsymbol{B}$ is row reduced then it has exactly m rows of degree $\leq d_\ell$, and they form a submatrix $[\boldsymbol{P} \;\; \boldsymbol{R}] \in \mathbb{K}[T]^{m\times(m+m')}$ of $\boldsymbol{B}$ such that $\boldsymbol{P}$ is a minimal matrix generator for $\boldsymbol{\mathcal{F}}$.*

Proof. For any relation $\boldsymbol{p} \in \mathbb{K}[T]^{1\times m}$ for $\boldsymbol{\mathcal{F}}$, there exists $\boldsymbol{r} \in \mathbb{K}[T]^{1\times m'}$ such that $\deg(\boldsymbol{r}) < \deg(\boldsymbol{p})$ and $[\boldsymbol{p} \;\; \boldsymbol{r}] \in \mathcal{A}(\boldsymbol{F}, d)$. Indeed, from Lemma 2.4, if $\boldsymbol{p}$ is a relation for $\boldsymbol{\mathcal{F}}$ then $\boldsymbol{q} = \boldsymbol{p}\boldsymbol{Z}$ has polynomial entries, where $\boldsymbol{Z} = \sum_{s\geq 0} \boldsymbol{F}_s T^{-s-1}$. Then the vector $\boldsymbol{r} = -\boldsymbol{p}(\sum_{s\geq d} \boldsymbol{F}_s T^{d-s-1})$ has polynomial entries, has degree less than $\deg(\boldsymbol{p})$, and is such that $[\boldsymbol{p} \;\; \boldsymbol{r}]\boldsymbol{F} = \boldsymbol{q}T^d$.

Conversely, for any vectors $\boldsymbol{p} \in \mathbb{K}[T]^{1\times m}$ and $\boldsymbol{r} \in \mathbb{K}[T]^{1\times m'}$, if $[\boldsymbol{p} \;\; \boldsymbol{r}] \in \mathcal{A}(\boldsymbol{F}, d)$ and $\deg([\boldsymbol{p} \;\; \boldsymbol{r}]) \leq d - d_r - 1$, then $\boldsymbol{p}$ is a relation for $\boldsymbol{\mathcal{F}}$. Indeed, if $[\boldsymbol{p} \;\; \boldsymbol{r}] \in \mathcal{A}(\boldsymbol{F}, d)$ we have $\boldsymbol{p}(\sum_{0\leq s<d} \boldsymbol{F}_s T^{d-s-1}) = \boldsymbol{r} \bmod T^d$. Since $d \geq d_r + 1$ and $\deg([\boldsymbol{p} \;\; \boldsymbol{r}]) \leq d - d_r - 1$, this

implies that the coefficients of degree $d-d_r$ to $d-1$ of $\boldsymbol{p}(\sum_{0\le s<d}\boldsymbol{F}_s T^{d-s-1})$ are zero. Then, Lemma 2.8 shows that $\boldsymbol{p}$ is a relation for $\boldsymbol{\mathcal{F}}$.

The two items in the theorem follow. □

Using fast approximant basis algorithms, we obtain the main results from this section. They are stated in slightly more generality than needed, since in our main algorithm, we will always use $m' = m$. However, we believe that the more general form stated here may find further applications. The proof is a direct consequence of the previous theorem, using the algorithms of respectively [20], [52], and [24].

Corollary 2.10. *Let $\boldsymbol{\mathcal{F}} \subset \mathbb{K}^{m\times m'}$ be a linearly recurrent sequence and let $d = d_\ell + d_r + 1$, where $(d_\ell, d_r) \in \mathbb{N}^2$ are such that the canonical left (resp. right) matrix generator of $\boldsymbol{\mathcal{F}}$ has degree $\le d_\ell$ (resp. $\le d_r$). Then, given $\boldsymbol{F}_0, \dots, \boldsymbol{F}_{d-1}$, one can compute*

- *a minimal left matrix generator for $\boldsymbol{\mathcal{F}}$ using $O(m'^{\omega}\mathsf{M}(d)\log(d))$ operations in $\mathbb{K}$ if $m' \in \Omega(m)$;*
- *a minimal left matrix generator for $\boldsymbol{\mathcal{F}}$ using $O(m^{\omega}\mathsf{M}(m'd/m)\log(d))$ operations in $\mathbb{K}$ if $m' \in O(m)$. This bound assumes $\mathsf{M}(kd) \in O(k^{\omega-1}\mathsf{M}(d))$; this holds in particular if $\mathsf{M}(\ )$ is quasi-linear and $\omega > 2$.*
- *the canonical left matrix generator for $\boldsymbol{\mathcal{F}}$ using $O((m+m')^{\omega-1}\mathsf{M}(m'd)\log(m'd)^3)$ operations in $\mathbb{K}$.*

In particular, when $m' = m$, we can find a minimal left matrix generator of $\boldsymbol{\mathcal{F}}$ in $O(m^{\omega}\mathsf{M}(d)\log(d))$ operations in $\mathbb{K}$.

2.3 Application to the block Wiedemann algorithm

We next apply the results seen above to a particular class of matrix sequences, namely the Krylov sequences used in Coppersmith's block Wiedemann algorithm [10]. In what follows, the integer m' of the previous subsection is now taken equal to m.

Let $\boldsymbol{M}$ be in $\mathbb{K}^{D\times D}$ and $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{K}^{D\times m}$ be two blocking matrices for some $m \le D$. We can then define the Krylov sequence $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}} = (\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}})_{s\ge 0} \subset \mathbb{K}^{m\times m}$ by

$$\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}} = \boldsymbol{U}^{\perp}\boldsymbol{M}^{s}\boldsymbol{V}, \quad s \ge 0.$$

This sequence is linearly recurrent, since the minimal polynomial of $\boldsymbol{M}$ is a scalar relation for it. The following theorem states some useful properties of any minimal left generator of $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}}$, with in particular a bound on its degree, for generic choices of $\boldsymbol{U}$ and $\boldsymbol{V}$; we also state properties of the invariant factors of such a generator. These results are not new, as all statements can be found between [50] and [28] (see also [26] for an analysis of the block Wiedemann algorithm). We chose to give a close-to-self-contained presentation, that relies on these two references for the key properties.

We let $s_1, \dots, s_r$ be the invariant factors of $T\boldsymbol{I}_D - \boldsymbol{M}$, ordered in such a way that $s_r | s_{r-1} | \cdots | s_1$, and let $d_i = \deg(s_i)$ for all i; for $i > r$, we let $s_i = 1$, with $d_i = 0$. We define $\nu = d_1 + \cdots + d_m \le D$ and $\delta = \lceil \nu/m \rceil \le \lceil D/m \rceil$.

Theorem 2.11. *For a generic choice of $\boldsymbol{U}$ and $\boldsymbol{V}$ in $\mathbb{K}^{D\times m}$, the following holds. Let $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ be a minimal left generator for $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}}$ and denote by $\sigma_1,\dots,\sigma_k$ the invariant factors of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$, for some $k\leq m$, ordered as above, and write $\sigma_{k+1}=\cdots=\sigma_m=1$. Then,*

- *$\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is a minimal left generator for the sequence $\boldsymbol{\mathcal{L}}_{\boldsymbol{U}}=(\boldsymbol{U}^\perp\boldsymbol{M}^s)_{s\geq 0}$;*
- *$\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ has degree δ;*
- *$s_i=\sigma_i$ for $1\leq i\leq m$.*

Proof. Without loss of generality, we can assume that $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is the canonical left generator. Then, define the matrix sequence $\boldsymbol{\mathcal{L}}_{\boldsymbol{U}}=(\boldsymbol{U}^\perp\boldsymbol{M}^s)_{s\geq 0}$; this sequence is linearly generated as well, and we let $\boldsymbol{P}_{\boldsymbol{U}}$ be the canonical left generator for it. We denote by $\langle\boldsymbol{U}\rangle$ the vector space generated by the rows of $\boldsymbol{U}^\perp,\boldsymbol{U}^\perp\boldsymbol{M},\boldsymbol{U}^\perp\boldsymbol{M}^2,\dots$, and we write $D_{\boldsymbol{U}}=\dim(\langle\boldsymbol{U}\rangle)$.

First, we prove that for any $\boldsymbol{U}$ in $\mathbb{K}^{D\times m}$, for a generic $\boldsymbol{V}$ in $\mathbb{K}^{D\times m}$, $\boldsymbol{P}_{\boldsymbol{U}}=\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$. Indeed, by [50, Lemma 4.2] (which considers right-generators), there exist matrices $\boldsymbol{J}_{\boldsymbol{U}}$ in $\mathbb{K}^{D\times D_{\boldsymbol{U}}}$ and $\boldsymbol{N}_{\boldsymbol{U}}\in\mathbb{K}^{D_{\boldsymbol{U}}\times D_{\boldsymbol{U}}}$, with $\boldsymbol{J}_{\boldsymbol{U}}$ of full rank $D_{\boldsymbol{U}}$, and where $\boldsymbol{N}_{\boldsymbol{U}}$ is a matrix of the restriction of $\boldsymbol{M}^\perp$ to $\langle\boldsymbol{U}\rangle$, such that $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}=\boldsymbol{P}_{\boldsymbol{U}}$ if and only if the dimension of the vector space generated by the rows of the span of $\boldsymbol{V}^\perp\boldsymbol{J}_{\boldsymbol{U}},\boldsymbol{V}^\perp\boldsymbol{J}_{\boldsymbol{U}}\boldsymbol{N}_{\boldsymbol{U}},\boldsymbol{V}^\perp\boldsymbol{J}_{\boldsymbol{U}}\boldsymbol{N}_{\boldsymbol{U}}^2,\dots$ is equal to $D_{\boldsymbol{U}}$.

By construction, one can find a basis of $\langle\boldsymbol{U}\rangle$ in which the matrix of $\boldsymbol{N}_{\boldsymbol{U}}$ is block-companion, with $\mu\leq m$ blocks (take the $\boldsymbol{N}_{\boldsymbol{U}}$-span of the first column of $\boldsymbol{U}$, then of the second column, working modulo the previous vector space, etc.). Thus, $\boldsymbol{N}_{\boldsymbol{U}}^\perp$ is similar to a block-companion matrix with μ blocks as well; since $\boldsymbol{J}_{\boldsymbol{U}}^\perp\boldsymbol{V}$ has m columns, the span of the columns of $\boldsymbol{J}_{\boldsymbol{U}}^\perp\boldsymbol{V},\boldsymbol{N}_{\boldsymbol{U}}^\perp\boldsymbol{J}_{\boldsymbol{U}}^\perp\boldsymbol{V},\dots$ has full dimension $D_{\boldsymbol{U}}$ for a generic $\boldsymbol{J}_{\boldsymbol{U}}^\perp\boldsymbol{V}$, and thus for a generic $\boldsymbol{V}$, since $\boldsymbol{J}_{\boldsymbol{U}}$ has rank $D_{\boldsymbol{U}}$. As a result, as claimed, for generic choices of $\boldsymbol{U}$ and $\boldsymbol{V}$, $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}=\boldsymbol{P}_{\boldsymbol{U}}$.

Let us next introduce a matrix $\mathscr{U}$ of indeterminates of size $D\times m$, and let $\boldsymbol{P}_{\mathscr{U}}$ be the canonical generating polynomial of the "generic" sequence $(\mathscr{U}^\perp\boldsymbol{M}^s)_{s\geq 0}$. The notation $\langle\mathscr{U}\rangle$ and $D_{\mathscr{U}}$ are defined as above. In particular, by [50, Proposition 6.1], the canonical generating polynomial $\boldsymbol{P}_{\mathscr{U}}$ has degree δ and determinantal degree ν.

Now, for a generic $\boldsymbol{U}$ in $\mathbb{K}^{D\times m}$, $D_{\boldsymbol{U}}=D_{\mathscr{U}}$. On the other hand, by [50, Lemma 4.3], for any $\boldsymbol{U}$ (including $\mathscr{U}$), the degree of $\boldsymbol{P}_{\boldsymbol{U}}$ is equal to the first index d such that the dimension of the span of the rows of $\boldsymbol{U}^\perp,\boldsymbol{U}^\perp\boldsymbol{M},\dots,\boldsymbol{U}^\perp\boldsymbol{M}^{d-1}$ is $D_{\boldsymbol{U}}$. As a result, for generic $\boldsymbol{U}$, $\boldsymbol{P}_{\boldsymbol{U}}$ and $\boldsymbol{P}_{\mathscr{U}}$ have the same degree, that is, δ. Taking the first item into account, we see that the second item is proved as well.

We conclude by proving that for generic $\boldsymbol{U},\boldsymbol{V}$, the invariant factors $\sigma_1,\dots,\sigma_m$ of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ are $s_1,\dots,s_m$. By [28, Theorem 2.12] (which considers right-generators as well), for any $\boldsymbol{U}$ and $\boldsymbol{V}$ in $\mathbb{K}^{D\times m}$, for $i=1,\dots,m$, the i-th invariant factor σ_i of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ divides s_i, so that $\deg(\det(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}))\leq\nu$, with equality if and only if $\sigma_i=s_i$ for all $i\leq m$.

For $\boldsymbol{U}$ as above and any integers e,d, we let $\mathrm{Hk}_{e,d}(\boldsymbol{U})$ be the block Hankel matrix

$$\mathrm{Hk}_{d,e}(\boldsymbol{U})=\begin{bmatrix}\boldsymbol{U}^\perp\\ \boldsymbol{U}^\perp\boldsymbol{M}\\ \boldsymbol{U}^\perp\boldsymbol{M}^2\\ \vdots\\ \boldsymbol{U}^\perp\boldsymbol{M}^{d-1}\end{bmatrix}\begin{bmatrix}\boldsymbol{I}_D & \boldsymbol{M} & \boldsymbol{M}^2 & \cdots & \boldsymbol{M}^{e-1}\end{bmatrix}.$$

By [28, Eq. (2.6)], $\mathrm{rank}(\mathrm{Hk}_{d,e}(\boldsymbol{U})) = \deg(\det(\boldsymbol{P}_{\boldsymbol{U}}))$ for $d \geq \deg(\boldsymbol{P}_{\boldsymbol{U}})$ and $e \geq D$. We take $e = D$, so that $\mathrm{rank}(\mathrm{Hk}_{d,D}(\boldsymbol{U})) = \deg(\det(\boldsymbol{P}_{\boldsymbol{U}}))$ for $d \geq \deg(\boldsymbol{P}_{\boldsymbol{U}})$. On the other hand, the sequence $\mathrm{rank}(\mathrm{Hk}_{d,D}(\boldsymbol{U}))$ is constant for $d \geq D$; as a result, $\mathrm{rank}(\mathrm{Hk}_{D,D}(\boldsymbol{U})) = \deg(\det(\boldsymbol{P}_{\boldsymbol{U}}))$. For the same reason, we also have $\mathrm{rank}(\mathrm{Hk}_{D,D}(\mathscr{U})) = \deg(\det(\boldsymbol{P}_{\mathscr{U}}))$, so that for a generic $\boldsymbol{U}$, $\boldsymbol{P}_{\boldsymbol{U}}$ and $\boldsymbol{P}_{\mathscr{U}}$ have the same determinantal degree, that is, ν. As a result, for generic $\boldsymbol{U}$ and $\boldsymbol{V}$, we also have $\deg(\det(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})) = \deg(\det(\boldsymbol{P}_{\boldsymbol{U}})) = \nu$, and the conclusion follows. □

In particular, combining Corollary 2.10 and Theorem 2.11, we deduce that for generic $\boldsymbol{U}, \boldsymbol{V}$, given the first $2\lceil D/m \rceil$ terms of the sequence $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}}$, we can recover a minimal matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ of it using $O(m^{\omega-1}\mathsf{M}(D)\log(D)) \subset O^{\sim}(m^{\omega-1}D)$ operations in $\mathbb{K}$.

Besides, the theorem shows that for generic $\boldsymbol{U}$ and $\boldsymbol{V}$, the largest invariant factor σ_1 of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is the minimal polynomial P of $\boldsymbol{M}$. Given $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$, P can thus be computed by solving a linear system $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}\boldsymbol{x} = \boldsymbol{y}$, where $\boldsymbol{y}$ is a vector of m randomly chosen elements in $\mathbb{K}$: for a generic choice of $\boldsymbol{y}$, the least common multiple of the denominators of the entries of $\boldsymbol{x}$ is P. Thus, both P and $\boldsymbol{x}$ can be computed using high-order lifting [46, Algorithm 5] on input $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ and $\boldsymbol{y}$. By [46, Corollary 16], this costs

$$
\begin{aligned}
&O\Bigg(m^{\omega}\mathsf{M}(D/m)\log(m) + \sum_{0\leq i\leq \log_2(m)} 4^i(2^{-i}m)^{\omega}\mathsf{M}(D/m) \\
&\qquad\qquad + m^2\mathsf{M}(D/m)\log(D/m)\log(m) + m\mathsf{M}(D)\log(D)\Bigg) \\
&\subset O(m^{\omega}\mathsf{M}(D/m)\log(m) + m\mathsf{M}(D)\log(D)\log(m)) \\
&\subset O(m^{\omega-1}\mathsf{M}(D)\log(D)\log(m)) \\
&\subset O^{\sim}(m^{\omega-1}D)
\end{aligned}
\tag{5}
$$

operations in $\mathbb{K}$. The latter algorithm is randomized, since it chooses a random point in $\mathbb{K}$ to compute (parts of) the expansion of the inverse of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$.

An alternative solution would be to compute the Smith form of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$, but using an algorithm such as Storjohann's [46, Section 17], the cost is slightly higher (on the level of logarithmic factors).

2.4 Computing a scalar numerator

Let us keep the notation of the previous subsection. The main advantage of using the block Wiedemann algorithm is that it allows one to distribute the bulk of the computation in a straightforward manner: on a platform with m processors (or cores, ...), one would typically compute the $D \times m$ matrices $\boldsymbol{L}_s = \boldsymbol{U}^{\perp}\boldsymbol{M}^s$ for $s = 0, \ldots, 2\lceil D/m \rceil$ by having each processor compute a sequence $\boldsymbol{u}_i\boldsymbol{M}^s$, where $\boldsymbol{u}_i$ is the ith row of $\boldsymbol{U}^{\perp}$. From these values, we then deduce the matrices $\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}} = \boldsymbol{U}^{\perp}\boldsymbol{M}^s\boldsymbol{V} = \boldsymbol{L}_s\boldsymbol{V}$ for $s = 0, \ldots, 2\lceil D/m \rceil$. Note that in our description, we assume for simplicity that memory is not an issue, so that we can store all needed elements from e.g. sequence $\boldsymbol{L}_s$; we discuss this in slightly more detail in Section 4.1.

Our main algorithm will also resort to scalar numerators of the form $\Omega((\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w})_{s\geq 0}, P)$, where $\boldsymbol{w}$ is a given vector in $\mathbb{K}^{D\times 1}$ and P is the minimal polynomial of $\boldsymbol{M}$. Since P may have degree D, and then $\Omega((\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w})_{s\geq 0}, P)$ itself may have degree $D-1$, the definition of Ω suggests that we may need to compute up to D terms of the sequence $\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w}$, which we would of course like to avoid. We now present an alternative solution which involves solving a univariate polynomial linear system and computing a matrix numerator, but only uses the sequence elements $\boldsymbol{L}_s = \boldsymbol{U}^\perp\boldsymbol{M}^s$ for $s = 0, \ldots, \lceil D/m\rceil - 1$, which have already been computed.

Fix i in $1, \ldots, m$ and let $\boldsymbol{a}_i$ be the row vector defined by

$$\boldsymbol{a}_i = [0 \ \cdots \ 0 \ P \ 0 \ \cdots \ 0](\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1},$$

where the minimal polynomial P appears at the ith entry of the left-hand row vector.

Lemma 2.12. *For generic $\boldsymbol{U}, \boldsymbol{V}$, the row vector $\boldsymbol{a}_i$ has polynomial entries of degree at most $\deg(P) \leq D$.*

Proof. For generic $\boldsymbol{U}, \boldsymbol{V}$, we saw that P is the largest invariant factor of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$; thus, the product $P\ (\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1}$ has polynomial entries. Since $\boldsymbol{a}_i$ the ith row of this matrix, $\boldsymbol{a}_i$ has polynomial entries. Now, since $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is reduced, the predictable degree property [25, Theorem 6.3-13] holds; it implies that each entry of $\boldsymbol{a}_i$ has degree at most the maximum of the degrees of the entries of $\boldsymbol{a}_i\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$. This maximum is $\deg(P)$. □

To compute $\boldsymbol{a}_i$, we use again Storjohann's high-order lifting; according to Eq. (5), the cost is $O(m^{\omega-1}\mathsf{M}(D)\log(D)\log(m)) \subset O\tilde{}(m^{\omega-1}D)$ operations in $\mathbb{K}$. Once $\boldsymbol{a}_i$ is known, the following lemma shows that we can recover the scalar numerator $\Omega((\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w})_{s\geq 0}, P)$ as a dot product.

Lemma 2.13. *For a generic choice of $\boldsymbol{U}$ and $\boldsymbol{V}$, and for any $\boldsymbol{w}$ in $\mathbb{K}^{D\times 1}$, $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is a nonsingular matrix of relations for the sequence $\boldsymbol{\mathcal{E}} = (\boldsymbol{e}_s)_{s\geq 0}$, with $\boldsymbol{e}_s = \boldsymbol{U}^\perp\boldsymbol{M}^s\,\boldsymbol{w}$ for all s, and we have*

$$[\ \Omega((\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w})_{s\geq 0}, P)\ ] = \boldsymbol{a}_i \cdot \boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}) \tag{6}$$

with $\boldsymbol{a}_i$ in $\mathbb{K}[T]^{1\times m}$ and $\boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}) \in \mathbb{K}[T]^{m\times 1}$.

Proof. The first item in Theorem 2.11 shows that for a generic choice of $\boldsymbol{U}$ and $\boldsymbol{V}$, $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ cancels the sequence $(\boldsymbol{U}^\perp\boldsymbol{M}^s)_{s\geq 0}$, and thus the sequence $\boldsymbol{\mathcal{E}}$ as well; this proves the first point. Then, the equality in Eq. (6) directly follows from the definitions:

$$\begin{aligned}
[\ \Omega((\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w})_{s\geq 0}, P)\ ] &= [\ P\ ] \sum_{s\geq 0} \frac{\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w}}{T^{s+1}} \\
&= [\ 0 \ \cdots \ 0 \ P \ 0 \ \cdots \ 0\ ] \sum_{s\geq 0} \frac{\boldsymbol{U}^\perp\boldsymbol{M}^s\boldsymbol{w}}{T^{s+1}} \\
&= [\ 0 \ \cdots \ 0 \ P \ 0 \ \cdots \ 0\ ]\ (\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1}\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}} \sum_{s\geq 0} \frac{\boldsymbol{U}^\perp\boldsymbol{M}^s\boldsymbol{w}}{T^{s+1}} \\
&= \boldsymbol{a}_i \cdot \boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}).
\end{aligned}$$

□

The algorithm to compute the scalar numerator $\Omega((\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w})_{s\geq 0}, P)$ follows; in the algorithm, we assume that we know $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$, P, $\boldsymbol{a}_i$, and the matrices $\boldsymbol{L}_s = \boldsymbol{U}^\perp\boldsymbol{M}^s \in \mathbb{K}^{m\times D}$ for $s = 0, \ldots, \lceil D/m\rceil - 1$.

Algorithm 1 ScalarNumerator$(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}, P, \boldsymbol{w}, i, \boldsymbol{a}_i, (\boldsymbol{L}_s)_{0\leq s<\lceil D/m\rceil})$

Input:

- a minimal generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ of $(\boldsymbol{U}^\perp\boldsymbol{M}^s\boldsymbol{V})_{s\geq 0}$
- the minimal polynomial P of $\boldsymbol{M}$
- $\boldsymbol{w}$ in $\mathbb{K}^{D\times 1}$
- i in $\{1, \ldots, m\}$
- $\boldsymbol{a}_i = [0\ \cdots\ 0\ P\ 0\ \cdots\ 0](\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1} \in \mathbb{K}[T]^{1\times m}$
- $\boldsymbol{L}_s = \boldsymbol{U}^\perp\boldsymbol{M}^s \in \mathbb{K}^{m\times D}$, for $s = 0, \ldots, \lceil D/m\rceil - 1$

Output:

- the scalar numerator $\Omega((\boldsymbol{u}_i\boldsymbol{M}^s\boldsymbol{w})_{s\geq 0}, P) \in \mathbb{K}[T]$

1. compute $\boldsymbol{e}_s = \boldsymbol{L}_s\boldsymbol{w} \in \mathbb{K}^{m\times 1}$ for $s = 0, \ldots, \lceil D/m\rceil - 1$
2. use these values to compute the matrix numerator $\boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}) \in \mathbb{K}[T]^{m\times 1}$ by Eq. (4)
3. **return** the entry of the 1×1 matrix $\boldsymbol{a}_i \cdot \boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})$

Computing the first $\lceil D/m\rceil$ values of the sequence $\boldsymbol{\mathcal{E}} = (\boldsymbol{e}_s)_{s\geq 0}$ is done by using the equality $\boldsymbol{e}_s = \boldsymbol{L}_s\boldsymbol{w}$ and takes $O(D^2)$ base field operations. Then, applying the cost estimate given after Eq. (4), we see that we have enough terms to compute $\boldsymbol{\Omega}((\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}) \in \mathbb{K}[T]^{m\times 1}$ and that it takes $O(m^2\mathsf{M}(D/m)) \subset O(m\mathsf{M}(D))$ operations in $\mathbb{K}$. Then, the dot product with $\boldsymbol{a}_i$ takes $O(m\mathsf{M}(D))$ operations, since both vectors have size m and entries of degree at most D. Thus, the runtime is

$$O(D^2 + m\mathsf{M}(D)) \subset O\tilde{}(D^2)$$

operations in $\mathbb{K}$.

3 Sequences associated to a zero-dimensional ideal

We now focus on our main question: computing a zero-dimensional parametrization of an algebraic set of the form $V = V(I)$, for some zero-dimensional ideal I in $\mathbb{K}[X_1, \ldots, X_n]$.

We write $V = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\kappa\}$, with all $\boldsymbol{\alpha}_i$'s in $\overline{\mathbb{K}}^n$, and $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \ldots, \alpha_{i,n})$ for all i. We also let D be the dimension of $\mathscr{Q} = \mathbb{K}[X_1, \ldots, X_n]/I$, so that $\kappa \leq D$, and *we assume that* $\mathrm{char}(\mathbb{K})$ *is greater than* D.

In this section, we recall and slightly expand on results from the appendix of [8], with the objective of computing a zero-dimensional parametrization of V. These results were themselves inspired by those in [40], which were devoted to computations with the trace linear form $\mathscr{Q} \to \mathbb{K}$.

At this stage, we do not discuss data structures or complexity (this is the subject of the next section); the algorithm in this section simply describes what polynomials should be computed in order to obtain a zero-dimensional parametrization.

3.1 The structure of the dual

For i in $\{1, \ldots, \kappa\}$, let $\mathscr{Q}_i$ be the local algebra at $\boldsymbol{\alpha}_i$, that is $\mathscr{Q}_i = \overline{\mathbb{K}}[X_1, \ldots, X_n]/I_i$, with I_i the $\mathfrak{m}_{\boldsymbol{\alpha}_i}$-primary component of I. By the Chinese Remainder Theorem, $\mathscr{Q} \otimes_{\mathbb{K}} \overline{\mathbb{K}} = \overline{\mathbb{K}}[X_1, \ldots, X_n]/I$ is isomorphic to the direct product $\mathscr{Q}_1 \times \cdots \times \mathscr{Q}_\kappa$. We let N_i be the *nil-index* of $\mathscr{Q}_i$, that is, the maximal integer N such that $\mathfrak{m}_{\alpha_i}^N$ is not contained in I_i; for instance, $N_i = 0$ if and only if $\mathscr{Q}_i$ is a field, if and only if $\boldsymbol{\alpha}_i$ is a non-singular root of I. We also let $D_i = \dim_{\overline{\mathbb{K}}}(\mathscr{Q}_i)$, so that we have $D_i \geq N_i$ and $D = D_1 + \cdots + D_\kappa$.

The sequences we consider below are of the form $(\ell(X^s))_{s\geq 0}$, for ℓ a $\mathbb{K}$-linear form $\mathscr{Q} \to \mathbb{K}$ and X in $\mathscr{Q}$ (we will often write $\ell \in \hom_{\mathbb{K}}(\mathscr{Q}, \mathbb{K})$). For such sequences, the following standard result will be useful (see e.g. [8, Propositions 1 & 2] for a proof).

Lemma 3.1. *Let X be in $\mathscr{Q}$ and let $P \in \mathbb{K}[T]$ be its minimal polynomial. For a generic choice of ℓ in $\hom_{\mathbb{K}}(\mathscr{Q}, \mathbb{K})$, P is the minimal polynomial of the sequence $(\ell(X^s))_{s\geq 0}$.*

The following results are classical; they go back to [31], and have been used in computational algebra since the 1990's [32, 35]. Fix i in $1, \ldots, \kappa$. There exists a basis of the dual $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_i, \overline{\mathbb{K}})$ consisting of linear forms $(\lambda_{i,j})_{1\leq j\leq D_i}$ of the form

$$\lambda_{i,j} : f \mapsto (\Lambda_{i,j}(f))(\boldsymbol{\alpha}_i),$$

where $\Lambda_{i,j}$ is the operator

$$f \mapsto \Lambda_{i,j}(f) = \sum_{\mu=(\mu_1,\ldots,\mu_n)\in S_{i,j}} c_{i,j,\mu} \frac{\partial^{\mu_1+\cdots+\mu_n} f}{\partial X_1^{\mu_1} \cdots \partial X_n^{\mu_n}},$$

for some finite subset $S_{i,j}$ of $\mathbb{N}^n$ and non-zero constants $c_{i,j,\mu}$ in $\overline{\mathbb{K}}$. For instance, when $\boldsymbol{\alpha}_i$ is non-singular, we have $D_i = 1$, so there is only one function $\lambda_{i,j}$, namely $\lambda_{i,1}$; we write it $\lambda_{i,1}(f) = f(\boldsymbol{\alpha}_i)$.

More generally, we can always take $\lambda_{i,1}$ of the form $\lambda_{i,1}(f) = f(\boldsymbol{\alpha}_i)$; for $j > 1$, we can then also assume that $S_{i,j}$ does not contain $\mu = (0, \ldots, 0)$ (that is, all terms in $\Lambda_{i,j}$ have order 1 or more). Thus, introducing new variables $(U_{i,j})_{j=1,\ldots,D_i}$, we deduce the existence of non-zero homogeneous linear forms $P_{i,\mu}$ in $(U_{i,j})_{j=1,\ldots,D_i}$ such that for any ℓ in $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_i, \overline{\mathbb{K}})$,

there exists $\boldsymbol{u}_i = (u_{i,j}) \in \overline{\mathbb{K}}^{D_i}$ such that we have

$$\begin{aligned}
\ell : f \mapsto \ell(f) &= \sum_{j=1}^{D_i} u_{i,j} \lambda_{i,j}(f) \\
&= \sum_{j=1}^{D_i} u_{i,j} \big(\Lambda_{i,j}(f)\big)(\boldsymbol{\alpha}_i) \\
&= \sum_{j=1}^{D_i} u_{i,j} \sum_{\mu=(\mu_1,\ldots,\mu_n)\in S_{i,j}} c_{i,j,\mu} \frac{\partial^{\mu_1+\cdots+\mu_n} f}{\partial X_1^{\mu_1} \cdots \partial X_n^{\mu_n}}(\boldsymbol{\alpha}_i) \\
&= \sum_{\mu=(\mu_1,\ldots,\mu_n)\in S_i} P_{i,\mu}(\boldsymbol{u}_i) \frac{\partial^{\mu_1+\cdots+\mu_n} f}{\partial X_1^{\mu_1} \cdots \partial X_n^{\mu_n}}(\boldsymbol{\alpha}_i), \qquad (7)
\end{aligned}$$

where S_i is the union of $S_{i,1}, \ldots, S_{i,D_i}$, with in particular $P_{i,(0,\ldots,0)} = U_{i,1}$ and where $P_{i,\mu}$ depends only on $(U_{i,j})_{j=2,\ldots,D_i}$ for all μ in S_i, $\mu \neq (0,\ldots,0)$. Explicitly, we can write $P_{i,\mu} = \sum_{j\in\{1,\ldots,D_i\} \mid \mu\in S_{i,j}} c_{i,j,\mu} U_{i,j}$.

Fix ℓ non-zero in $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_i, \overline{\mathbb{K}})$, written as in Eq. (7). We can then define its *order* w and *symbol* π. The former is the maximum of all $|\mu| = \mu_1 + \cdots + \mu_n$ for $\mu = (\mu_1, \ldots, \mu_n)$ in S_i such that $P_{i,\mu}(\boldsymbol{u}_i)$ is non-zero; by [35, Lemma 3.3] we have $w \leq N_i - 1$. Then, we let

$$\pi = \sum_{\mu\in S_i,\ |\mu|=w} P_{i,\mu}(\boldsymbol{u}_i) X_1^{\mu_1} \cdots X_n^{\mu_n}$$

be the *symbol* of ℓ; by construction, this is a non-zero polynomial. In the following paragraphs, we will need the next lemma.

Lemma 3.2. *Fix i in $\{1,\ldots,\kappa\}$. For a generic choice of ℓ in $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_i, \overline{\mathbb{K}})$ and of $t_1, \ldots, t_n$ in $\overline{\mathbb{K}}^n$, $\pi_i(t_1, \ldots, t_n)$ is non-zero.*

Proof. Let Ω be the maximum of all $|\mu| = \mu_1 + \cdots + \mu_n$ for $\mu = (\mu_1, \ldots, \mu_n)$ in S_i, and define

$$\Pi = \sum_{\mu\in S_i,\ |\mu|=\Omega} P_{i,\mu} X_1^{\mu_1} \cdots X_n^{\mu_n} \in \overline{\mathbb{K}}[U_{i,1}, \ldots, U_{i,D_i}, X_1, \ldots, X_n];$$

this is by construction a non-zero polynomial. As a result, for a generic choice of $\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,D_i})$, which defines a linear form ℓ in $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_i, \overline{\mathbb{K}})$ as in Eq. (7), and of $t_1, \ldots, t_n$ in $\overline{\mathbb{K}}^n$, the value $\Pi(u_{i,1}, \ldots, u_{i,D_i}, t_1, \ldots, t_n)$ is non-zero. Thus, the symbol of such a linear form ℓ is $\pi = \sum_{\mu\in S_i,\ |\mu|=\Omega} P_{i,\mu}(\boldsymbol{u}_i) X_1^{\mu_1} \cdots X_n^{\mu_n}$, and $\pi(t_1, \ldots, t_n)$ is then non-zero. □

Finally, we say a word about global objects. Fix a linear form $\ell : \mathscr{Q} \to \mathbb{K}$. By the Chinese Remainder Theorem, there exist unique $\ell_1, \ldots, \ell_\kappa$, with ℓ_i in $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_i, \overline{\mathbb{K}})$ for all i, such that the extension $\ell_{\overline{\mathbb{K}}} : \mathscr{Q} \otimes_{\mathbb{K}} \overline{\mathbb{K}} \to \overline{\mathbb{K}}$ decomposes as $\ell_{\overline{\mathbb{K}}} = \ell_1 + \cdots + \ell_\kappa$. Note that formally, we should write $\ell_{\overline{\mathbb{K}}} = \ell_1 \circ \phi_1 + \cdots + \ell_\kappa \circ \phi_\kappa$, where for all i, ϕ_i is the canonical projection $\mathscr{Q} \to \mathscr{Q}_i$; we will however omit these projection operators for simplicity.

We call *support* of ℓ the subset $\mathfrak{S}$ of $\{1,\dots,\kappa\}$ such that ℓ_i is non-zero exactly for i in $\mathfrak{S}$. As a consequence, for all f in $\mathscr{Q}$, we have

$$\ell(f) = \ell_1(f) + \cdots + \ell_\kappa(f) = \sum_{i\in\mathfrak{S}} \ell_i(f). \tag{8}$$

For i in $\mathfrak{S}$, we denote by w_i and π_i respectively the order and the symbol of ℓ_i. For such a subset $\mathfrak{S}$ of $\{1,\dots,\kappa\}$, we also write $\mathscr{Q}_\mathfrak{S} = \prod_{i\in\mathfrak{S}} \mathscr{Q}_i$ and $V_\mathfrak{S} = \{\boldsymbol{\alpha}_i \mid i \in \mathfrak{S}\}$.

3.2 A fundamental formula

Let X be in $\mathscr{Q}$ and ℓ in $\hom_\mathbb{K}(\mathscr{Q},\mathbb{K})$. The sequences $(\ell(X^s))_{s\geq 0}$, and more generally the sequences $(\ell(vX^s))_{s\geq 0}$ for v in $\mathscr{Q}$, are the core ingredients of our algorithm. This is justified by the following lemma, which gives an explicit description of generating series of the form $\sum_{s\geq 0} \ell(vX^s)/T^{s+1}$. A slightly less precise version of it is in [8]; the more explicit expression given here will be needed in the last section of this paper.

Lemma 3.3. *Let ℓ be in $\hom_\mathbb{K}(\mathscr{Q},\mathbb{K})$, with support $\mathfrak{S}$, and let $\{\pi_i \mid i\in\mathfrak{S}\}$ and $\{w_i \mid i\in\mathfrak{S}\}$ be the symbols and orders of $\{\ell_i \mid i\in\mathfrak{S}\}$, for $\{\ell_i \mid i\in\mathfrak{S}\}$ as in Eq. (8).*

Let $X = t_1X_1 + \cdots + t_nX_n$, for some $t_1,\dots,t_n$ in $\mathbb{K}$ and let v be in $\mathbb{K}[X_1,\dots,X_n]$. Then, we have the equality

$$\sum_{s\geq 0} \frac{\ell(vX^s)}{T^{s+1}} = \sum_{i\in\mathfrak{S}} \frac{v(\boldsymbol{\alpha}_i)\, w_i!\, \pi_i(t_1,\dots,t_n) + (T - X(\boldsymbol{\alpha}_i))A_{v,i}}{(T-X(\boldsymbol{\alpha}_i))^{w_i+1}}, \tag{9}$$

for some polynomials $\{A_{v,i} \in \overline{\mathbb{K}}[T] \mid i\in\mathfrak{S}\}$ which depend on the choice of v and are such that $A_{v,i}$ has degree less than w_i for all i in $\mathfrak{S}$.

Proof. Take v and X as above. Consider first an operator of the form $f \mapsto \frac{\partial^{|\mu|} f}{\partial X_1^{\mu_1}\cdots\partial X_n^{\mu_n}}$, where we write $|\mu| = \mu_1 + \cdots + \mu_n$. Then, we have the following generating series identities, with coefficients in $\mathbb{K}(X_1,\dots,X_n)$:

$$\begin{aligned}
\sum_{s\geq 0} \frac{\partial^{|\mu|}(vX^s)}{\partial X_1^{\mu_1}\cdots\partial X_n^{\mu_n}} \frac{1}{T^{s+1}} &= \sum_{s\geq 0} \frac{\partial^{|\mu|}(vX^s/T^{s+1})}{\partial X_1^{\mu_1}\cdots\partial X_n^{\mu_n}} \\
&= \frac{\partial^{|\mu|}}{\partial X_1^{\mu_1}\cdots\partial X_n^{\mu_n}} \left(\sum_{s\geq 0} \frac{vX^s}{T^{s+1}} \right) \\
&= \frac{\partial^{|\mu|}}{\partial X_1^{\mu_1}\cdots\partial X_n^{\mu_n}} \left(\frac{v}{T-X} \right) \\
&= \left(v\,|\mu|!\, \frac{1}{(T-X)^{|\mu|+1}} \prod_{1\leq k\leq n} \left(\frac{\partial X}{\partial X_k}\right)^{\mu_k} \right) + \frac{P_{|\mu|}}{(T-X)^{|\mu|}} + \cdots + \frac{P_1}{(T-X)} \\
&= \left(v\,|\mu|!\, \frac{1}{(T-X)^{|\mu|+1}} \prod_{1\leq k\leq n} t_k^{\mu_k} \right) + \frac{H}{(T-X)^{|\mu|}},
\end{aligned}$$

for some polynomials $P_1, \dots, P_{|\mu|}, H$ in $\mathbb{K}[X_1, \dots, X_n, T]$ that depend on the choices of μ, v and X, with $\deg(P_i, T) < i$ for all i and thus $\deg(H, T) < |\mu|$.

Take now a $\overline{\mathbb{K}}$-linear combination of such operators, such as $f \mapsto \sum_{\mu \in R} c_\mu \frac{\partial^{|\mu|} f}{\partial X_1^{\mu_1} \cdots \partial X_n^{\mu_n}}$ for some finite subset R of $\mathbb{N}^n$. The corresponding generating series becomes

$$\sum_{s \geq 0} \sum_{\mu \in R} c_\mu \frac{\partial^{|\mu|}(vX^s)}{\partial X_1^{\mu_1} \cdots \partial X_n^{\mu_n}} \frac{1}{T^{s+1}} = v \sum_{\mu \in R} \left(c_\mu |\mu|! \frac{1}{(T - X)^{|\mu|+1}} \prod_{1 \leq k \leq n} t_k^{\mu_k} \right) + \sum_{\mu \in R} \frac{H_\mu}{(T - X)^{|\mu|}},$$

where each $H_\mu \in \overline{\mathbb{K}}[X_1, \dots, X_n, T]$ has degree less than $|\mu|$ in T. Let w be the maximum of all $|\mu|$ for μ in R. We can rewrite the above as

$$v\, w! \sum_{\mu \in R, |\mu| = w} \left(c_\mu \frac{1}{(T - X)^{w+1}} \prod_{1 \leq k \leq n} t_k^{\mu_k} \right) + \frac{A}{(T - X)^w},$$

for some polynomial $A \in \overline{\mathbb{K}}[X_1, \dots, X_n, T]$ of degree less than w in T. Then, if we let $\pi = \sum_{\mu \in R,\ |\mu| = w} c_\mu X_1^{\mu_1} \cdots X_n^{\mu_n}$, this becomes

$$\sum_{s \geq 0} \sum_{\mu \in R} c_\mu \frac{\partial^{|\mu|}(vX^s)}{X_1^{\mu_1} \cdots X_n^{\mu_n}} \frac{1}{T^{s+1}} = v\, w!\, \pi(t_1, \dots, t_n) \frac{1}{(T - X)^{w+1}} + \frac{A}{(T - X)^w}.$$

Applying this formula to the sum $\ell = \sum_{i \in \mathfrak{S}} \ell_i$ from Eq. (8), and taking into account the expression in Eq. (7) for each ℓ_i, we obtain the claim in the lemma. □

The most useful consequence of this lemma is the following interpolation formula involving the operator Ω of the previous section, which generalizes a comment we made in Section 2.1.

Fix a subset $\mathfrak{S}$ of $\{1, \dots, \kappa\}$, and let X be as in the previous lemma. The mapping $X : V_{\mathfrak{S}} \to \overline{\mathbb{K}}$ defined by $\boldsymbol{\alpha}_i \mapsto X(\boldsymbol{\alpha}_i)$ plays a special role in the formula in the previous lemma; this leads us to the following definitions.

- We consider ℓ and X as in Lemma 3.3, such that ℓ has support $\mathfrak{S}$.
- $\mathfrak{T}$ is the subset of $\mathfrak{S}$ consisting of all indices i such that
  - $X(\boldsymbol{\alpha}_{i'}) \neq X(\boldsymbol{\alpha}_i)$ for $i' \neq i$ in $\mathfrak{S}$;
  - $\pi_i(t_1, \dots, t_n)$ is non-zero.
- $\{r_1, \dots, r_c\}$ are the pairwise distinct values taken by X on $V_{\mathfrak{S}}$, for some $c \leq |\mathfrak{S}|$.
- $\mathfrak{t}$ is the set of all indices j in $\{1, \dots, c\}$ such that
  - the fiber $X^{-1}(r_j) \subset V_{\mathfrak{S}}$ contains a single point, written $\boldsymbol{\alpha}_{\sigma_j}$;
  - $\pi_{\sigma_j}(t_1, \dots, t_n)$ is non-zero.

  Remark that $j \mapsto \sigma_j$ induces a one-to-one correspondence between $\mathfrak{t}$ and $\mathfrak{T}$, and that $X(\boldsymbol{\alpha}_{\sigma_j}) = r_j$ for all j in $\mathfrak{t}$.

Lemma 3.4. *Let ℓ, X and all other notation be as above. Let further P be the minimal polynomial of X in $\mathscr{Q}_{\mathfrak{S}}$. Suppose that P is also the minimal polynomial of the sequence*

$(\ell(X^s))_{s\geq 0}$. *Then, for v in $\mathbb{K}[X_1,\dots,X_n]$, P cancels the sequence $(\ell(vX^s))_{s\geq 0}$ and there exist non-zero constants $\{c_j \mid j\in\mathfrak{t}\}$ such that for v in $\mathbb{K}[X_1,\dots,X_n]$,*

$$\Omega((\ell(vX^s))_{s\geq 0},P) = c_j v(\boldsymbol{\alpha}_{\sigma_j}) \quad \textit{for all } j \textit{ in } \mathfrak{t}.$$

Proof. For $j=1,\dots,c$, we write T_j for the set of all indices i in $\mathfrak{S}$ such that $X(\boldsymbol{\alpha}_i)=r_j$; the sets $T_1,\dots,T_c$ form a partition of $\mathfrak{S}$. When T_j has cardinality 1, we thus have $T_j=\{\sigma_j\}$.

Take an arbitrary v in $\mathbb{K}[X_1,\dots,X_n]$ and let us collect terms in Eq. (9) as

$$\begin{aligned}\sum_{s\geq 0}\frac{\ell(vX^s)}{T^{s+1}} &= \sum_{j\in\{1,\dots,c\}}\sum_{i\in T_j}\frac{v(\boldsymbol{\alpha}_i)w_i!\,\pi_i(t_1,\dots,t_n)+(T-r_j)A_{v,i}}{(T-r_j)^{w_i+1}}\\ &= \sum_{j\in\mathfrak{t}}\frac{v(\boldsymbol{\alpha}_{\sigma_j})w_{\sigma_j}!\,\pi_{\sigma_j}(t_1,\dots,t_n)+(T-r_j)A_{v,\sigma_j}}{(T-r_j)^{w_{\sigma_j}+1}}\\ &\quad+\sum_{j\in\{1,\dots,c\}-\mathfrak{t}}\frac{\sum_{i\in T_j}\Big(\big[v(\boldsymbol{\alpha}_i)w_i!\,\pi_i(t_1,\dots,t_n)+(T-r_j)A_{v,i}\big](T-r_j)^{y_j-(w_i+1)}\Big)}{(T-r_j)^{y_j}},\end{aligned}$$

where y_j is the maximum of all w_i for i in T_j. Remark that for $v=1$, our condition that $\pi_i(t_1,\dots,t_n)$ is non-zero for i in $\mathfrak{T}$, together with our assumption on the characteristic of $\mathbb{K}$, imply that in the second line, all terms in the first sum are non-zero and in reduced form.

After simplifying terms in the second sum, we can rewrite the expression above as

$$\sum_{s\geq 0}\frac{\ell(vX^s)}{T^{s+1}} = \sum_{j\in\mathfrak{t}}\frac{v(\boldsymbol{\alpha}_{\sigma_j})w_{\sigma_j}!\,\pi_{\sigma_j}(t_1,\dots,t_n)+(T-r_j)A_{v,\sigma_j}}{(T-r_j)^{w_{\sigma_j}+1}}+\sum_{j\in\{1,\dots,c\}-\mathfrak{t}}\frac{D_{v,j}}{(T-r_j)^{z_{v,j}}},$$

for some positive integers $\{z_{v,j}\mid j\in\{1,\dots,c\}-\mathfrak{t}\}$ and polynomials $\{D_{v,j}\mid j\in\{1,\dots,c\}-\mathfrak{t}\}$ such that for all j in $\{1,\dots,c\}-\mathfrak{t}$, we have $\deg(D_{v,j})<z_{v,j}$ and $\gcd(D_{v,j},T-r_j)=1$. Some of the polynomials $D_{v,j}$ may vanish, so we let $\mathfrak{u}_v\subset\{1,\dots,c\}-\mathfrak{t}$ be the set of all j for which this is not the case. We then arrive at our final form for this sum, namely

$$\sum_{s\geq 0}\frac{\ell(vX^s)}{T^{s+1}} = \sum_{j\in\mathfrak{t}}\frac{v(\boldsymbol{\alpha}_{\sigma_j})w_{\sigma_j}!\,\pi_{\sigma_j}(t_1,\dots,t_n)+(T-r_j)A_{v,\sigma_j}}{(T-r_j)^{w_{\sigma_j}+1}}+\sum_{j\in\mathfrak{u}_v}\frac{D_{v,j}}{(T-r_j)^{z_{v,j}}},\tag{10}$$

where all terms in the second sum are non-zero and in reduced form (and similarly for the first sum, for $v=1$). This implies that the minimal polynomial of the sequence $(\ell(vX^s))_{s\geq 0}$ is

$$P_v=\prod_{j\in\mathfrak{t}}(T-r_j)^{\zeta_j}\prod_{j\in\mathfrak{u}_v}(T-r_j)^{z_{v,j}},$$

for some integers $\{\zeta_j\leq w_{\sigma_j}+1\mid j\in\mathfrak{t}\}$; for $v=1$, we actually have $\zeta_j=w_{\sigma_j}+1$ for all such j.

Now, for $v=1$, we assume that the minimal polynomial of the sequence $(\ell(X^s))_{s\geq 0}$ is the minimal polynomial P of t in $\mathscr{Q}_{\mathfrak{S}}$. Writing $\mathfrak{u}=\mathfrak{u}_1$ and $z_j=z_{1,j}$ for all j in $\mathfrak{u}$, we can thus write it as

$$P=\prod_{j\in\mathfrak{t}}(T-r_j)^{w_{\sigma_j}+1}\prod_{j\in\mathfrak{u}}(T-r_j)^{z_j}.$$

Since it is the minimal polynomial of X in $\mathscr{Q}_{\mathfrak{S}}$, it also cancels the sequence $(\ell(vX^s))_{s\geq 0}$ for any v, so that for all v, P_v divides P (which proves the first point in the lemma), and in particular $\mathfrak{u}_v$ is contained in $\mathfrak{u}$. We can then rewrite the sum in Eq. (10) as

$$\sum_{s\geq 0} \frac{\ell(vX^s)}{T^{s+1}} = \frac{\Omega((\ell(vX^s))_{s\geq 0}, P)}{P},$$

with

$$\begin{aligned}
&\Omega((\ell(vX^s))_{s\geq 0}, P) = \\
&\sum_{j\in\mathfrak{t}} \Big(\big[v(\boldsymbol{\alpha}_{\sigma_j}) w_{\sigma_j}! \, \pi_{\sigma_j}(t_1, \ldots, t_n) + (T - r_j) A_{v,\sigma_j} \big] \prod_{\iota\in\mathfrak{t}-\{j\}} (T - r_\iota)^{w_{\sigma_\iota}+1} \Big) \prod_{j\in\mathfrak{u}} (T - r_j)^{z_j} \\
&+ \Big(\prod_{j\in\mathfrak{t}} (T - r_j)^{w_{\sigma_j}+1} \Big) \sum_{j\in\mathfrak{u}_v} \Big((T - r_j)^{z_j - z_{v,j}} D_{v,j} \prod_{\iota\in\mathfrak{u}-\{j\}} (T - r_\iota)^{z_\iota} \Big).
\end{aligned}$$

In particular, for k in $\mathfrak{t}$, the value $\Omega((\ell(vX^s))_{s\geq 0}, P)(r_k)$ is

$$\begin{aligned}
\Omega((\ell(vX^s))_{s\geq 0}, P)(r_k) &= v(\boldsymbol{\alpha}_{\sigma_k}) w_{\sigma_k}! \, \pi_{\sigma_k}(t_1, \ldots, t_n) \prod_{\iota\in\mathfrak{t}-\{k\}} (r_\iota - r_k)^{w_{\sigma_\iota}+1} \prod_{j\in\mathfrak{u}} (r_j - r_k)^{z_k} \\
&= v(\boldsymbol{\alpha}_{\sigma_k}) c_k,
\end{aligned}$$

with

$$c_k = w_{\sigma_k}! \, \pi_{\sigma_k}(t_1, \ldots, t_n) \prod_{\iota\in\mathfrak{t}-\{k\}} (r_\iota - r_k)^{w_{\sigma_\iota}+1} \prod_{j\in\mathfrak{u}} (r_j - r_k)^{z_k}$$

for k in $\mathfrak{t}$. This is a non-zero constant, independent of v, which completes the proof. □

3.3 Computing a zero-dimensional parametrization

As a first application, the following algorithm shows how to compute a zero-dimensional parametrization of $V_{\mathfrak{S}}$. Our main usage of it will be with $\mathfrak{S} = \{1, \ldots, \kappa\}$, in which case $V_{\mathfrak{S}} = V$, but in the last section of the paper, we will also work with strict subsets. At this stage, we do not discuss data structures, complexity, or what terms in the sequences are needed; this is the subject of the next section.

This algorithm is from [8]; we point out that a precursor of this kind of results is in [40], for a particular choice of a linear form $\mathscr{Q} \to \mathbb{K}$, namely the trace.

Algorithm 2 Parametrization(ℓ, X)

Input:
- a linear form ℓ over $\mathscr{Q}_{\mathfrak{S}}$
- $X = t_1X_1 + \cdots + t_nX_n$

Output:
- polynomials $((Q, V_1, \ldots, V_n), X)$, with $Q, V_1, \ldots, V_n$ in $\mathbb{K}[T]$

1. let P be the minimal polynomial of the sequence $(\ell(X^s))_{s\geq 0}$
2. let Q be the squarefree part of P
3. let $C_1 = \Omega((\ell(X^s))_{s\geq 0}, P)$
4. **for** $i = 1, \ldots, n$ **do**
   let $C_{X_i} = \Omega((\ell(X_iX^s))_{s\geq 0}, P)$
5. **return** $((Q, C_{X_1}/C_1 \bmod Q, \ldots, C_{X_n}/C_1 \bmod Q), X)$

Lemma 3.5. *Suppose that ℓ is a generic element of $\hom_{\mathbb{K}}(\mathscr{Q}_{\mathfrak{S}}, \overline{\mathbb{K}})$ and that X is a generic linear form. Then the output $((Q, V_1, \ldots, V_n), X)$ of* Parametrization(ℓ, X) *is a zero-dimensional parametrization of $V_{\mathfrak{S}}$.*

Proof. A generic choice of X separates the points of $V_{\mathfrak{S}}$, and we saw in Lemma 3.2 that for a generic choice of ℓ and X, $\pi_i(t_1, \ldots, t_n)$ vanishes for no index i in $\mathfrak{S}$. As a result, with the notation introduced prior to Lemma 3.4, we have $\mathfrak{T} = \mathfrak{S}$ and $\mathfrak{t}$ consists of $|\mathfrak{S}|$ pairwise distinct values.

Besides, we recall that for a generic ℓ in $\hom_{\mathbb{K}}(\mathscr{Q}_{\mathfrak{S}}, \overline{\mathbb{K}})$, the minimal polynomials of $(\ell(X^s))_{s\geq 0}$ and of X are the same (Lemma 3.1). Thus, the polynomial P we compute at Step 1 is indeed the minimal polynomial of X (with roots $\{r_j \mid j \in \mathfrak{t}\}$), and we can apply Lemma 3.4; then, for any root r_j of P, and $i = 1, \ldots, n$, we have

$$\frac{C_{X_i}(r_j)}{C_1(r_j)} = \frac{\Omega((\ell(X_iX^s))_{s\geq 0}, P)(r_j)}{\Omega((\ell(X^s))_{s\geq 0}, P)(r_j)} = \frac{c_j\alpha_{\sigma_j,i}}{c_j} = \alpha_{\sigma_j,i},$$

so that $C_{X_i}/C_1 \bmod P$ is the ith polynomial in the zero-dimensional parametrization of V corresponding to X. □

We demonstrate how this algorithm works through a small example, in which we already know the coordinates of the solutions. Let

$$I = \langle (X_1 - 1)(X_2 - 2), (X_1 - 3)(X_2 - 4)\rangle \subset \mathbb{F}_{101}[X_1, X_2].$$

Then, $V(I) = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2\}$, with $\boldsymbol{\alpha}_1 = (1, 4)$ and $\boldsymbol{\alpha}_2 = (3, 2)$; we take $X = X_1$, which separates the points of $V(I)$. We choose the linear form

$$\ell : \mathbb{F}_{101}[X_1, X_2]/I \to \mathbb{F}_{101},\ f \mapsto \ell(f) = 17f(\boldsymbol{\alpha}_1) + 33f(\boldsymbol{\alpha}_2);$$

then, $\mathfrak{S} = \{1, 2\}$, and the symbols π_1 and π_2 are respectively the constants 17 and 33. We have

$$\ell(X_1^s) = 17 \cdot 1^s + 33 \cdot 3^s$$
$$\ell(X_2X_1^s) = 17 \cdot 4 \cdot 1^s + 33 \cdot 2 \cdot 3^s.$$

We associate a generating series to each sequence:

$$Z_1 = \sum_{s\geq 0} \frac{\ell(X_1^s)}{T^{s+1}} = \frac{17}{T-1} + \frac{33}{T-3} = \frac{17(T-3)+33(T-1)}{(T-1)(T-3)}$$
$$Z_{X_2} = \sum_{s\geq 0} \frac{\ell(X_2X_1^s)}{T^{s+1}} = \frac{17\cdot 4}{T-1} + \frac{33\cdot 2}{t-3} = \frac{17\cdot 4(T-3)+33\cdot 2(T-1)}{(T-1)(T-3)}.$$

These generating series have for common denominator $P = (T-1)(T-3)$, whose roots are the coordinates of X_1 in $V(I)$; their numerators are respectively

$$C_1 = \Omega((\ell(X_1^s))_{s\geq 0}, P) = 17(T-3)+33(T-1)$$

and

$$C_{X_2} = \Omega((\ell(X_2X_1^s))_{s\geq 0}, P) = 17\cdot 4(T-3)+33\cdot 2(T-1).$$

Now, let

$$\begin{aligned} V_2 &= \frac{C_{X_2}}{C_1} \mod P \\ &= \frac{17\cdot 4(T-3)+33\cdot 2(T-1)}{17(T-3)+33(T-1)} \mod P \\ &= 100T+5. \end{aligned}$$

Then, $V_2(1) = 4$ and $V_2(3) = 2$, as needed.

4 The main algorithm

In this section, we extend the algorithm of [8] to compute a zero-dimensional parametrization of $V(I)$, for some zero-dimensional ideal I of $\mathbb{K}[X_1,\ldots,X_n]$, by using blocking methods. As input, we assume that we know a monomial basis $\mathscr{B} = (b_1,\ldots,b_D)$ of $\mathscr{Q} = \mathbb{K}[X_1,\ldots,X_n]/I$, together with the multiplication matrices $\boldsymbol{M}_1,\ldots,\boldsymbol{M}_n$ of respectively $X_1,\ldots,X_n$ in this basis; for definiteness, we suppose that the first basis element in $\mathscr{B}$ is $b_1 = 1$. As before, we let D denote the dimension of $\mathscr{Q}$.

The first subsection presents the main algorithm. Its main feature is that after we compute the Krylov sequence used to find a minimal matrix generator, we recover all entries of the output for a minor cost, without computing another Krylov sequence. We make no assumption on I (radicality, shape position, ...), except of course that it has dimension zero; however, we assume (as in the previous subsection) that the characteristic of $\mathbb{K}$ is greater than D.

Then, we present a simple example, and experimental results of an implementation based on the C++ libraries LinBox [22], Eigen [23] and NTL [42].

4.1 Description, correctness and cost analysis

We mentioned that Steel's method [45] already uses the block Wiedemann algorithm to compute the minimal polynomial P of $X = t_1X_1 + \cdots + t_nX_n$; given sufficiently many terms of the sequence $(\boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{V})$, this is done by means of polynomial lattice reduction, without computing the matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$. Knowing the roots of P in $\mathbb{K}$, that algorithm uses an "evaluation" method for the rest (several Gröbner basis computations, all with one variable less).

Our algorithm computes the whole zero-dimensional parametrization of $V(I)$ for essentially the same cost as the computation of the minimal polynomial. In what follows, $\boldsymbol{\varepsilon}_1$ denotes the size-D column vector whose only non-zero entry is a 1 in the first row: $\boldsymbol{\varepsilon}_1 = [1\ 0\ \cdots\ 0]^\perp$.

Algorithm 3 BlockParametrization$(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, X)$

Input:

- $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ defined as above
- $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{K}^{D \times m}$, for some block dimension $m \in \{1, \ldots, D\}$
- $X = t_1X_1 + \cdots + t_nX_n$

Output:

- polynomials $((Q, V_1, \ldots, V_n), X)$, with $Q, V_1, \ldots, V_n$ in $\mathbb{K}[T]$

1. let $\boldsymbol{M} = t_1\boldsymbol{M}_1 + \cdots + t_n\boldsymbol{M}_n$
2. compute $\boldsymbol{L}_s = \boldsymbol{U}^\perp \boldsymbol{M}^s$ for $s = 0, \ldots, 2d-1$, with $d = \lceil D/m \rceil$
3. compute $\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}} = \boldsymbol{L}_s\boldsymbol{V}$ for $s = 0, \ldots, 2d-1$
4. compute a minimal matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ of $(\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}})_{0 \le s < 2d}$
5. let P be the largest invariant factor of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$
6. let Q be the squarefree part of P
7. let $\boldsymbol{a}_1 = [P\ 0\ \cdots\ 0](\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1}$
8. let $C_1 = \mathsf{ScalarNumerator}(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}, P, \boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{L}_s)_{0 \le s < d})$
9. **for** $i = 1, \ldots, n$ **do**
   let $C_{X_i} = \mathsf{ScalarNumerator}(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}, P, \boldsymbol{M}_i\boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{L}_s)_{0 \le s < d})$
10. **return** $((Q, C_{X_1}/C_1 \bmod Q, \ldots, C_{X_n}/C_1 \bmod Q), X)$

We first prove correctness of the algorithm, for generic choices of $t_1, \ldots, t_n$, $\boldsymbol{U}$ and $\boldsymbol{V}$. The first step computes the multiplication matrix $\boldsymbol{M} = t_1\boldsymbol{M}_1 + \cdots + t_n\boldsymbol{M}_n$ of $X = t_1X_1 + \cdots + t_nX_n$. Then, we compute the first $2d$ terms of the sequence $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}} = (\boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{V})_{s \ge 0}$. For generic choices of these two matrices, as discussed in Section 2.3, Theorem 2.11 shows that the matrix polynomial $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is indeed a minimal left generator of the sequence $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}}$, that P is the minimal polynomial of X and Q its squarefree part.

We find the rest of the polynomials in the output by following Algorithm 2. Crucially, the scalar numerators $\Omega(\cdots)$ needed in this algorithm are computed using the method of Section 2.4. Indeed, applying Lemma 2.13, we see that calling Algorithm ScalarNumerator

at Steps 8 and 9 computes

$$C_1 = \Omega((\boldsymbol{u}_i \boldsymbol{M}^s \boldsymbol{\varepsilon}_1)_{s\geq 0}, P) \quad \text{and} \quad C_{X_i} = \Omega((\boldsymbol{u}_1 \boldsymbol{M}^s \boldsymbol{M}_i \boldsymbol{\varepsilon}_1)_{s\geq 0}, P), \quad i = 1, \ldots, n.$$

Let $\ell : \mathscr{Q} \to \mathbb{K}$ be the linear form $f = \sum_{i=1}^{D} f_i b_i \mapsto \sum_{1\leq i\leq D} f_i u_{i,1}$, where $u_{i,1}$ is the entry at position $(i, 1)$ in $\boldsymbol{U}$. The two polynomials above can be rewritten as

$$C_1 = \Omega((\ell(X^s))_{s\geq 0}, P) \quad \text{and} \quad C_{X_i} = \Omega((\ell(X_i X^s))_{s\geq 0}, P),$$

so they coincide with the polynomials computed in Algorithm 2. Then, Lemma 3.5 shows that for generic $\boldsymbol{U}$ and X, the output of BlockParametrization is indeed a zero-dimensional parametrization of $V(I)$.

Remark 4.1. As already pointed out in Section 2.4, the algorithm is written assuming that memory usage is not a limiting factor (this makes it slightly easier to write the pseudo-code). As described here, the algorithm stores $\Theta(D^2)$ field elements in the sequence $\boldsymbol{L}_s$ computed at Step 2, since they are re-used at Steps 8 and 9. We may instead discard each matrix $\boldsymbol{L}_s$ after it is used, by computing on the fly the column vectors needed for Steps 8 and 9.

If the multiplication matrices are dense, little is to be gained this way (since in the worst case they use themselves nD^2 field elements), but savings can be substantial if these matrices are sparse. □

For the cost analysis, we will mainly focus on a sparse model, assuming that $\boldsymbol{M}$, as well as all $\boldsymbol{M}_i$'s, have density $\rho \in [0, 1]$, that is, at most ρD^2 non-zero entries. As a result, a matrix-vector product by $\boldsymbol{M}$ can be done in $O(\rho D^2)$ operations in $\mathbb{K}$ (we briefly discuss variants of the algorithm using dense linear algebra at the end of this subsection). In particular, the cost incurred at Step 1 to compute $\boldsymbol{M}$ is $O(\rho n D^2)$.

In this context, the main purpose of Coppersmith's blocking strategy is to allow for easy parallelization. Computing the matrices $\boldsymbol{L}_s = \boldsymbol{U}^\perp \boldsymbol{M}^s$, for $s = 0, \ldots, 2d-1$, is the bottleneck of the algorithm, but this can be parallelized. This is done by working row-wise, computing independently the sequences $(\boldsymbol{\ell}_{i,s})_{0\leq s<2d}$ of the ith rows of $(\boldsymbol{L}_s)_{0\leq s<2d}$ as $\boldsymbol{\ell}_{i,0} = \boldsymbol{u}_i$ and $\boldsymbol{\ell}_{i,s+1} = \boldsymbol{\ell}_{i,s}\boldsymbol{M}$ for all i, s, where $\boldsymbol{u}_i$ is the ith row of $\boldsymbol{U}^\perp$. For a fixed $i \in \{1, \ldots, m\}$, computing $(\boldsymbol{\ell}_{i,s})_{0\leq i<2d}$ costs $O(d\rho D^2) = O(\rho D^3/m)$ field operations. If we are able to compute m vector-matrix products in parallel at once, the *span* of Step 2 is thus $O(\rho D^3/m)$, whereas the total work is $O(\rho D^3)$.

At Step 3, we can then compute $\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}} = \boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{V}$, for $s = 0, \ldots, 2d-1$ by the product

$$\begin{bmatrix} \boldsymbol{U}^\perp \\ \boldsymbol{U}^\perp \boldsymbol{M} \\ \boldsymbol{U}^\perp \boldsymbol{M}^2 \\ \vdots \\ \boldsymbol{U}^\perp \boldsymbol{M}^{2d-1} \end{bmatrix} \boldsymbol{V} = \begin{bmatrix} \boldsymbol{U}^\perp \boldsymbol{V} \\ \boldsymbol{U}^\perp \boldsymbol{M} \boldsymbol{V} \\ \boldsymbol{U}^\perp \boldsymbol{M}^2 \boldsymbol{V} \\ \vdots \\ \boldsymbol{U}^\perp \boldsymbol{M}^{2d} \boldsymbol{V} \end{bmatrix}$$

of size $O(D) \times D$ by $D \times m$; since $m \leq D$, this costs $O(m^{\omega-2} D^2)$ base field operations.

Recall from Section 2.2 that we can compute a minimal generating polynomial $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ in time $O(m^\omega \mathsf{M}(D/m) \log(D/m))$, which is in $O(m^{\omega-1} \mathsf{M}(D) \log(D))$. In Sections 2.3 and 2.4,

we saw that the largest invariant factor P and the vector $\boldsymbol{a}_1$ can be computed in time $O(m^{\omega-1}\mathsf{M}(D)\log(D)\log(m))$. Computing Q takes time $O(\mathsf{M}(D)\log(D))$.

In Section 2.4, we saw that each call to ScalarNumerator takes $O(D^2+m\mathsf{M}(D))$ operations in $\mathbb{K}$, for a total of $O(nD^2+nm\mathsf{M}(D))$; the final modular calculations modulo Q take time $O(n\mathsf{M}(D)+\mathsf{M}(D)\log(D))$. Altogether, assuming perfect parallelization at Step 2, the total span is

$$O\left(\rho\frac{D^3}{m}+m^{\omega-1}\mathsf{M}(D)\log(D)\log(m)+nD^2+nm\mathsf{M}(D)\right),$$

and the total work is

$$O\left(\rho D^3+m^{\omega-1}\mathsf{M}(D)\log(D)\log(m)+nD^2+nm\mathsf{M}(D)\right).$$

Of course, we could parallelize other steps than Step 2, but it is simultaneously the most costly in theory and in practice, and the easiest to parallelize.

We conclude this section by a discussion of "dense" versions of the algorithm (to be used when the density ρ is close to 1). If we use a dense model for our matrices, our algorithms should rely on dense matrix multiplication. We will see two possible approaches, which respectively take $m=1$ and $m=D$; we will not discuss how they parallelize, merely pointing out that one may simply parallelize dense matrix multiplications throughout the algorithms.

Let us first discuss the modifications in the algorithm to apply if we choose $m=1$. In this case, blocking has no effect. We compute the row-vectors $\boldsymbol{L}_s$, for $s=0,\dots,2D-1$, using the square-and-multiply technique used in Keller-Gehrig's algorithm [29], for $O(D^\omega\log(D))$ operations in $\mathbb{K}$. For generic choices of $\boldsymbol{U}$ and $\boldsymbol{V}$, the minimum generating polynomial matrix $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is equal to the minimum polynomial P of $\boldsymbol{M}$, and can be computed by the Berlekamp-Massey algorithm, or a fast version of it; besides, $\boldsymbol{a}_1 = P(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1} = 1$. Computing the scalar numerators is simply a power series multiplication in degree at most D. Altogether, the runtime is $O(D^\omega\log(D)+nD^2)$, where the second term gives the cost of computing $\boldsymbol{M}$.

When $m=D$, $\boldsymbol{U},\boldsymbol{V}\in\mathbb{K}^{D\times D}$ are square matrices and $d=D/m=1$. A minimal generating polynomial of $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}} = (\boldsymbol{U}^\perp\boldsymbol{V}, \boldsymbol{U}^\perp\boldsymbol{M}\boldsymbol{V},\dots)$ is $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}} = T\boldsymbol{I}_D - \boldsymbol{U}^\perp\boldsymbol{M}\boldsymbol{U}^{-\perp}$ and its largest invariant factor P and $\boldsymbol{a}_1$ can be computed in $O(D^\omega\log(D))$ operations in $\mathbb{K}$ using high-order lifting.

The numerator $\boldsymbol{\Omega}(\boldsymbol{U}^\perp\boldsymbol{M}\boldsymbol{\varepsilon}_1, \boldsymbol{P}_{\boldsymbol{U},m\boldsymbol{V}})$ is then seen to be $\boldsymbol{U}^\perp\boldsymbol{\varepsilon}_1$, that is, the first column of $\boldsymbol{U}^\perp$; we recover C_1 from it through a dot product with $\boldsymbol{a}_1$. Similarly, the numerator $\boldsymbol{\Omega}(\boldsymbol{U}^\perp\boldsymbol{M}\boldsymbol{M}_i\boldsymbol{\varepsilon}_1, \boldsymbol{P}_{\boldsymbol{U},m\boldsymbol{V}})$ is $\boldsymbol{U}^\perp\boldsymbol{M}_i\boldsymbol{\varepsilon}_1$, and gives us C_{X_i}. Altogether, the runtime is again $O(D^\omega\log(D)+nD^2)$, where the second term now gives the cost of computing $\boldsymbol{M}$, as well as C_1 and all C_{X_i}.

Remark 4.2. Our algorithm only computes the first invariant factor of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$, that is, of $T\boldsymbol{I}_D-\boldsymbol{M}$. A natural question is whether computing further invariant factors can be of any use in the algorithm (or possibly can help us determine part of the structure of the algebras $\mathscr{Q}_i$). □

4.2 Example

We give an example of our algorithm with a non-radical system as input. Let

$$I = \left\langle \begin{array}{c} X_1^3 + 88X_1^2 + 56X_1 + 21, \\ X_1^2X_2 + 91X_1^2 + 92X_1X_2 + 90X_1 + 20X_2 + 2, \\ X_1X_2^2 + 81X_1X_2 + 100X_1 + 96X_2^2 + 100X_2 + 5, \\ X_1^2X_2 + 81X_1^2 + 93X_1X_2 + 59X_1 + 16X_2 + 84, \\ X_1X_2^2 + 71X_1X_2 + 99X_1 + 97X_2^2 + 19X_2 + 8, \\ X_2^3 + 61X_2^2 + 96X_2 + 20 \end{array} \right\rangle \subset \mathbb{F}_{101}[X_1, X_2];$$

the corresponding residue class ring $\mathscr{Q} = \mathbb{F}_{101}[X_1, X_2]/I$ has dimension $D = 4$. Although it does not appear on the generators, the ideal I is simply $\mathfrak{m}_1^2\mathfrak{m}_2$, where $\mathfrak{m}_1 = \langle X_1 - 4, X_2 - 10\rangle$ and $\mathfrak{m}_2 = \langle X_1 - 5, X_2 - 20\rangle$ (this immediately implies that $D = 3 + 1 = 4$).

We choose $X = 2X_1 + 53X_2$, so that X_1, X_2 and X have respective multiplication matrices in the basis $\mathscr{B} = (1, X_2, X_2, X_2^2)$ of $\mathscr{Q}$ given by

$$\boldsymbol{M}_1 = \begin{bmatrix} 7 & 91 & 100 & 0 \\ 41 & 2 & 20 & 0 \\ 100 & 10 & 8 & 1 \\ 1 & 71 & 86 & 0 \end{bmatrix}, \boldsymbol{M}_2 = \begin{bmatrix} 40 & 1 & 91 & 0 \\ 5 & 0 & 2 & 1 \\ 0 & 0 & 10 & 0 \\ 81 & 0 & 71 & 0 \end{bmatrix}, \boldsymbol{M} = \begin{bmatrix} 13 & 33 & 74 & 0 \\ 44 & 4 & 45 & 53 \\ 99 & 20 & 41 & 2 \\ 53 & 41 & 97 & 0 \end{bmatrix}.$$

We choose $m = 2$ and take $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{F}_{101}^{D\times m}$ with entries

$$\boldsymbol{U} = \begin{bmatrix} 84 & 38 \\ 29 & 58 \\ 80 & 43 \\ 7 & 82 \end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix} 6 & 97 \\ 83 & 58 \\ 0 & 95 \\ 59 & 89 \end{bmatrix}.$$

We compute the first $2d = 2\lceil D/m\rceil = 4$ terms in the matrix sequence $\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}} = (\boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{V})_{s\geq 0}$ and its minimum generating matrix polynomial $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$. This is done by first computing

$$\boldsymbol{U}^\perp = \begin{bmatrix} 84 & 29 & 80 & 7 \\ 38 & 58 & 43 & 82 \end{bmatrix}, \quad \boldsymbol{U}^\perp\boldsymbol{M} = \begin{bmatrix} 54 & 28 & 67 & 81 \\ 34 & 52 & 90 & 29 \end{bmatrix}, \tag{11}$$

$$\boldsymbol{U}^\perp\boldsymbol{M}^2 = \begin{bmatrix} 33 & 91 & 3 & 2 \\ 47 & 77 & 47 & 7 \end{bmatrix}, \quad \boldsymbol{U}^\perp\boldsymbol{M}^3 = \begin{bmatrix} 89 & 80 & 87 & 82 \\ 34 & 56 & 55 & 34 \end{bmatrix},$$

from which we get, by right-multiplication by $\boldsymbol{V}$,

$$\boldsymbol{\mathcal{F}}_{\boldsymbol{U},\boldsymbol{V}} = \begin{bmatrix} 92 & 75 \\ 83 & 51 \end{bmatrix}, \begin{bmatrix} 54 & 34 \\ 70 & 73 \end{bmatrix}, \begin{bmatrix} 92 & 54 \\ 16 & 74 \end{bmatrix}, \begin{bmatrix} 94 & 51 \\ 91 & 51 \end{bmatrix}, \ldots$$

and

$$\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}} = \begin{bmatrix} T^2 + 60T + 62 & 88T + 25 \\ 100T + 33 & T^2 + 84T + 78 \end{bmatrix}.$$

The largest invariant factor of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ is

$$P = T^3 + 76T^2 + 100T + 7,$$

with squarefree part $Q = T^2 + 8T + 61$. Next, we compute the row vector $\boldsymbol{a}_1$ and the numerator $\boldsymbol{\Omega}((\boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{\varepsilon}_1)_{s\geq 0}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})$:

$$\begin{aligned} \boldsymbol{a}_1 &= [T + 16, 13] \\ \boldsymbol{\Omega}((\boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{\varepsilon}_1)_{s\geq 0}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}) &= \begin{bmatrix} 84T + 55 \\ 38T + 11 \end{bmatrix}. \end{aligned}$$

The former is obtained by solving $\boldsymbol{a}_1 = [\ P\ 0\ ](\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1}$. The latter is made from the entries of non-negative degree in the product

$$\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}} \left(\begin{bmatrix} 84 \\ 38 \end{bmatrix} \cdot \frac{1}{T} + \begin{bmatrix} 54 \\ 34 \end{bmatrix} \cdot \frac{1}{T^2} + \cdots \right),$$

where the columns are the first columns of the matrices in Eq. (11) – as per Eq. (4), we only need $d = 2$ terms in the right-hand side. From this, we find the scalar numerator $C_1 = \Omega((\boldsymbol{u}_1 \boldsymbol{M}^s \boldsymbol{\varepsilon}_1)_{s\geq 0}, P)$ by means of the dot product $[\ C_1\ ] = \boldsymbol{a}_1 \cdot \boldsymbol{\Omega}((\boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{\varepsilon}_1)_{s\geq 0}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})$; we get

$$C_1 = 84T^2 + 75T + 13.$$

To find C_{X_1}, we proceed similarly: we compute $\boldsymbol{\Omega}((\boldsymbol{u}_1 \boldsymbol{M}^s \boldsymbol{M}_1 \boldsymbol{\varepsilon}_1)_{s\geq 0}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})$, and we obtain, after dot product with $\boldsymbol{a}_1$,

$$\begin{aligned} \boldsymbol{\Omega}((\boldsymbol{u}_1 \boldsymbol{M}^s \boldsymbol{M}_1 \boldsymbol{\varepsilon}_1)_{s\geq 0}, \boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}) &= \begin{bmatrix} 88T + 19 \\ 57T + 40 \end{bmatrix} \\ C_{X_1} &= 88T^2 + 47T + 16. \end{aligned}$$

Thus, the polynomial $V_1 = C_{X_1}/C_1 \bmod Q$ is given by $V_1 = 15T + 14$. We compute $V_2 = 49T + 9$ in the same way, and our output is

$$((T^2 + 8T + 61, 15T + 14, 49T + 9), 2X_1 + 53X_2).$$

As a sanity check, we recall that $V(I)$ has two points in $\mathbb{F}_{101}^2$, namely $(4, 10)$ and $(5, 20)$; accordingly, Q has two roots in $\mathbb{F}_{101}$, 33 and 60, and we have $(V_1(33) = 4, V_2(33) = 10)$ and $(V_1(60) = 5, V_2(60) = 20)$, as expected.

4.3 Experimental results

In Table 1, we give the timings in seconds for different values of m for Algorithm 3. Our implementation is based on Shoup's NTL [42] polynomials, LinBox [22] for dense polynomial matrices and matrix generator computations, and Eigen [23] for sparse matrix / vector products. It is dedicated to small prime fields; thus, as is done in [17] for dense matrices,

we use machine floats to do the bulk of the calculations. Explicitly, we rely on Eigen's `SparseMatrix<double, RowMajor>` class to store our multiplication matrices and do the matrix / vector products, reducing the results modulo p afterwards.

All timings are measured on an Intel Xeon CPU E5-2667 with 128 GB RAM and 8 cores (16 available through hyperthreading). For each value of m in $\{1, 3, 6\}$, we create and run m threads in parallel.

In all cases, we start from multiplication matrices computed from a degrevlex Gröbner basis in Magma [6]. The timings reported here do not include this precomputation; instead, we refer the reader to [15] for extensive experiments comparing the runtime of two similar stages (degree basis computation and conversion to a lex order) in that paper's algorithm. Rather than optimizing our implementation, our main focus here was to demonstrate the effects of parallelization, and how the Krylov sequence computation dominates the runtime in these examples. In particular, we believe that many improvements are possible for the polynomial matrix computations (computing minimal matrix generator, its largest invariant factor, ...), but this is by no means a bottleneck here.

All our inputs as well as our source code are available at https://git.uwaterloo.ca/sghyun/Block-sparse-FGLM. Some systems are well-known (Katsura or Eco from [34]), while we also consider several families of randomly generated inputs (some are inspired from [15]). Systems rand1-i have 3 variables and randomly generated equations (of degree depending on i), and rand2-i are similar, with 4 variables. These systems are generically radical and in shape position (for the projection on the first coordinate axis; the last column uses that convention). Systems mixed1-i and mixed2-i have similar numbers of variables as the previous ones, but some of their solutions are multiple, so the ideals are not radical (and not in shape position). Systems mixed3-i have only one multiple root (the others are simple), in increasing numbers of variables; they are not in shape position. Systems W1-κ-n-p are determinantal equations that describe the computation of critical points for the projection $(\alpha_1, \ldots, \alpha_n) \mapsto \alpha_1$ on $V(f_1, \ldots, f_p)$, where $f_1, \ldots, f_p$ are p equations of degree κ in n variables.

We used OpenMP `parallel for` pragmas to parallelize the computation of the above sequence, as described in Section 4.1. In the columns $m = 1, m = 3, m = 6$, the numbers in parentheses indicate the part of the total time spent in computing the Krylov sequence $(\boldsymbol{U}^\perp \boldsymbol{M}^s)_{0 \le s < 2d}$, with $d = \lceil D/m \rceil$. This is always by far the dominant factor. In other words, the immense majority of the time is spent doing sparse matrix / vector products for matrices with machine float entries.

Increasing m has two effects. On the plus side, plainly, it decreases the length of the sequence above. On the other side, while the algorithm performs better by a factor often close to 3 for $m = 3$, the gain is not as significant for $m = 6$, so that parallelization is less effective. It is an interesting question to clarify this issue. Besides, increasing m also increases the time to compute the output polynomials, through minimal generator computation, high-order lifting, ... However, from the observed speedup and the ratios in the table, we can conclude that this effect is minor.

Table 1: Timings (in seconds) for polynomials over $\mathbb{F}_{65537}$

name	n	D	**density** ρ	$m = 1$	$m = 3$	$m = 6$	**radical / shape position**
rand1-26	3	17576	0.06	692(0.98)	307(0.969)	168(0.926)	yes/yes
rand1-28	3	21952	0.05	1261(0.983)	471(0.971)	331(0.944)	yes/yes
rand1-30	3	27000	0.05	2191(0.986)	786(0.974)	512(0.946)	yes/yes
rand2-10	4	10000	0.14	301(0.981)	109(0.964)	79(0.934)	yes/yes
rand2-11	4	14641	0.13	851(0.987)	303(0.975)	239(0.961)	yes/yes
rand2-12	4	20736	0.12	2180(0.99)	784(0.982)	648(0.972)	yes/yes
mixed1-22	3	10864	0.07	207(0.973)	75(0.947)	58(0.909)	no/no
mixed1-23	3	12383	0.07	294(0.976)	107(0.95)	92(0.925)	no/no
mixed1-24	3	14040	0.07	413(0.979)	150(0.958)	125(0.934)	no/no
mixed2-10	4	10256	0.16	362(0.984)	130(0.969)	113(0.954)	no/no
mixed2-11	4	14897	0.14	989(0.988)	384(0.98)	278(0.965)	no/no
mixed2-12	4	20992	0.13	2480(0.991)	892(0.984)	807(0.977)	no/no
mixed3-12	12	4109	0.5	75(0.963)	27(0.941)	21(0.929)	no/no
mixed3-13	13	8206	0.48	554(0.982)	198(0.973)	171(0.968)	no/no
eco12	12	1024	0.55	1(0.801)	1(0.641)	1(0.63)	yes/yes
sot1	5	8694	0.01	21(0.745)	9(0.62)	9(0.552)	yes/no
W1-6-5-2	5	18000	0.2	2362(0.992)	859(0.986)	696(0.979)	yes/yes
W1-4-6-2	6	6480	0.32	184(0.981)	66(0.965)	54(0.951)	yes/yes
katsura10	11	1024	0.63	1(0.836)	1(0.679)	1(0.672)	yes/yes

We refer the reader to experiments with systems of comparable (or higher) degrees presented in [15] (the algorithm in that reference does not use a blocking strategy). Recall that the output in [15] is somewhat stronger than here (the authors compute a Gröbner basis of the input ideal, so multiplicities are kept). On the other hand, for ideals not in shape position, Table 2 of that reference reports only the calculation of the last polynomial in the Gröbner basis, whereas our algorithm makes no distinction between ideals in shape position or not.

5 Using the original coordinates

In this last section, we propose a refinement of the algorithm given previously; the main new feature is that we avoid using a generic linear form $X = t_1X_1 + \cdots + t_nX_n$ as much as possible. Indeed, such a linear combination is likely to result in a multiplication matrix $\boldsymbol{M} = t_1\boldsymbol{M}_1 + \cdots + t_n\boldsymbol{M}_n$ significantly more dense than $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$. In all this section, we

will work under the assumption that $\boldsymbol{M}_1$ is the sparsest matrix among $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$, and try to rely on computations involving $\boldsymbol{M}_1$ as much as possible.

All notation, such as I, $\mathscr{Q} = \mathbb{K}[X_1, \ldots, X_n]/I$, its basis $\mathscr{B} = (b_1, \ldots, b_D)$, the local algebras $\mathscr{Q}_i$, ..., are as in the previous two sections. In particular, we write $V = V(I) = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\kappa\}$, with all $\boldsymbol{\alpha}_i$'s in $\overline{\mathbb{K}}^n$, and $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \ldots, \alpha_{i,n})$ for all i.

5.1 Overview

The algorithm will decompose V into two parts: for the first part, we will be able to use X_1 as a linear form in our zero-dimensional parametrization; the remaining points will be dealt with using a random linear form $X = t_1X_1 + \cdots + t_nX_n$ as above. Throughout, we rely on the following operations: evaluations of linear forms on successive powers of a given element in $\mathscr{Q}$ (such as $1, X_1, X_2^2, \ldots$ or $1, X, X^2, \ldots$) and (polynomial) linear algebra, as we did before, as well as some operations on univariate polynomials related to the so-called *power projection* [43, 44].

The main algorithm is as follows. For the moment, we only describe its main structure; the subroutines are described in the next subsections.

Algorithm 4 BlockParametrizationWithSplitting$(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, X, Y)$

Input:

- $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ defined as above
- $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{K}^{D \times m}$, for some block dimension $m \in \{1, \ldots, D\}$
- $X = t_1X_1 + \cdots + t_nX_n$
- $Y = y_2X_2 + \cdots + y_nX_n$

Output:

- polynomials $((Q, V_1, \ldots, V_n), X)$, with $Q, V_1, \ldots, V_n$ in $\mathbb{K}[T]$

1. let $(F, G_1, \ldots, G_n, X_1) =$ BlockParametrizationX$_1(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, Y)$
2. let $(\boldsymbol{\Delta}_s), (\boldsymbol{\Delta}'_s), (\boldsymbol{\Delta}''_s)$ be correction matrices associated to the previous calculation
3. let $(R, W_1, \ldots, W_n, X) =$ BlockParametrizationResidual$(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, (\boldsymbol{\Delta}_s), (\boldsymbol{\Delta}'_s), (\boldsymbol{\Delta}''_s), X)$
4. let $((Q, V_1, \ldots, V_n), X) =$ Union$(((F, G_1, \ldots, G_n), X_1), ((R, W_1, \ldots, W_n), X))$
5. **return** $((Q, V_1, \ldots, V_n), X)$

The call to BlockParametrizationX$_1$ computes a zero-dimensional parametrization of a subset V' of V, such that X_1 separates the points of V' (that is, takes pairwise distinct values on the points of V'); this is done by using sequences of the form $(\boldsymbol{U}^\perp \boldsymbol{M}_1^s \boldsymbol{V})_{s \geq 0}$. The next stage computes three sequences of correction matrices, using objects that were computed in the call to BlockParametrizationX$_1$.

We then apply a modified version of Algorithm BlockParametrization, which we call BlockParametrizationResidual. It computes a zero-dimensional parametrization of $V'' = V - V'$ using sequences such as $(\boldsymbol{U}^\perp \boldsymbol{M}^s \boldsymbol{V})_{s \geq 0} - \boldsymbol{\Delta}_s$, where $\boldsymbol{M} = t_1\boldsymbol{M}_1 + \cdots + t_n\boldsymbol{M}_n$. Subtracting the correction terms $\boldsymbol{\Delta}_s$ has the effect of removing from V the points in V'; in

those "lucky" cases where V' is a large subset of V, $V'' = V - V'$ may only contain a few points, and few values for the latter sequences will be needed.

The last step involves changing coordinates in $((F, G_1, \ldots, G_n), X_1)$ to use X as a linear form instead, and performing the union of the two components V' and V''. These operations can be done in time $O(nD^{(\omega+1)/2})$ [39, Lemmas 2 & 3]; we will not discuss them further.

Remark 5.1. Another possibility is to return the parametrizations $(F, G_1, \ldots, G_n), X_1)$ and $((R, W_1, \ldots, W_n), X)$ without performing the union; this has the obvious advantage of partially decomposing the output. □

5.2 Describing the subset V' of V

In this paragraph, we give the details of Algorithm BlockParametrizationX_1. This is done by specializing the discussion of Section 3.2 to the case $X = X_1$: in the notation of that section, we take $\mathfrak{S} = \{1, \ldots, \kappa\}$, that is, $V_{\mathfrak{S}} = V$, and we let $r_1, \ldots, r_c$ be the pairwise distinct values taken by X_1 on V, for some $c \leq \kappa$. For $j = 1, \ldots, c$, we write T_j for the set of all indices i in $\{1, \ldots, \kappa\}$ such that $\alpha_{i,1} = r_j$; the sets $T_1, \ldots, T_c$ form a partition of $\{1, \ldots, \kappa\}$. When T_j has cardinality 1, we denote it as $T_j = \{\sigma_j\}$, for some index σ_j in $\{1, \ldots, \kappa\}$, so that $\alpha_{\sigma_j,1} = r_j$.

For $i = 1, \ldots, \kappa$, let us write ν_i for the degree of the minimal polynomial of X_1 in $\mathscr{Q}_i$; thus, this polynomial is $(T - \alpha_{i,1})^{\nu_i}$. For j in $\{1, \ldots, c\}$, we define μ_j as the maximum of all ν_i, for i in T_j. As a result, the minimal polynomial of X_1 in $\prod_{j \in T_j} \mathscr{Q}_j$ is $(T - r_j)^{\mu_j}$, and the minimal polynomial of X_1 in $\mathscr{Q}$ is $M = \prod_{j \in \{1,\ldots,c\}} (T - r_j)^{\mu_j}$.

Recall that for any linear form $\ell : \mathscr{Q} \to \mathbb{K}$, the extension $\ell : \mathscr{Q} \otimes_{\mathbb{K}} \overline{\mathbb{K}} \to \overline{\mathbb{K}}$ can be written uniquely as $\ell = \sum_{i \in \{1,\ldots,\kappa\}} \ell_i$, with $\ell_i : \mathscr{Q}_i \to \overline{\mathbb{K}}$; collecting terms, ℓ may also be written as $\ell = \sum_{j \in \{1,\ldots,c\}} \lambda_j$, with $\lambda_j = \sum_{i \in T_j} \ell_i$. Given such an ℓ, we first explain how to compute values of the form $\lambda_j(1)$. We will do this for some values of j only, namely those j for which $\mu_j = 1$. This result is close in spirit to Lemma 3.4, but does not assume that the projection $X_1 : V \to \overline{\mathbb{K}}$ is one-to-one (that lemma makes such an assumption, but works for a more general linear mapping $X : V \to \overline{\mathbb{K}}$).

Lemma 5.2. *Let ℓ be in $\hom_{\mathbb{K}}(\mathscr{Q}, \mathbb{K})$ and let M be the minimal polynomial of X_1 in $\mathscr{Q}$. Then, the polynomial $\Omega((\ell(X_1^s))_{s \geq 0}, M)$ is well-defined and satisfies*

$$\Omega((\ell(X_1^s))_{s \geq 0}, M)(r_j) = \lambda_j(1) M'(r_j) \quad \text{for all } j \text{ in } \{1, \ldots, c\} \text{ such that } \mu_j = 1.$$

Proof. Let $\mathfrak{e}$ be the set of all indices j in $\{1, \ldots, c\}$ such that $\mu_j = 1$, and let $\mathfrak{f} = \{1, \ldots, c\} - \mathfrak{e}$; this definition allows us to split the generating series of sequence $(\ell(X_1^s))_{s \geq 0}$ as

$$\begin{aligned} \sum_{s \geq 0} \frac{\ell(X_1^s)}{T^{s+1}} &= \sum_{j \in \{1,\ldots,c\}} \sum_{i \in T_j} \sum_{s \geq 0} \frac{\ell_i(X_1^s)}{T^{s+1}} \\ &= \sum_{j \in \mathfrak{e}} \sum_{i \in T_j} \sum_{s \geq 0} \frac{\ell_i(X_1^s)}{T^{s+1}} + \sum_{j \in \mathfrak{f}} \sum_{i \in T_j} \sum_{s \geq 0} \frac{\ell_i(X_1^s)}{T^{s+1}}. \end{aligned}$$

Using Lemma 3.3 with $X = X_1$ and $v = 1$, any sum $\sum_{s\geq 0} \ell_i(X_1^s)/T^{s+1}$ in the second summand can be rewritten as

$$\frac{E_i}{(T-r_j)^{v_i}},$$

for some integer v_i, and for some polynomial $E_i \in \overline{\mathbb{K}}[T]$ of degree less than v_j. Next, take j in $\mathfrak{e}$. Since $\mu_j = 1$, $\nu_i = 1$ for all i in T_j, so that each such ℓ_i takes the form

$$\ell_i : f \mapsto (\Lambda_i(f))(\boldsymbol{\alpha}_i),$$

where Λ_i is a differential operator that does not involve $\partial/\partial X_1$. Since all terms of positive order in Λ_i involve one of $\partial/\partial X_2, \ldots, \partial/\partial X_n$, they cancel X_1^s for $s \geq 0$. Thus, $\ell_i(X_1^s)$ can be rewritten as $\ell_{i,1}\alpha_{i,1}^s$, for some constant $\ell_{i,1}$, and the generating series of these terms is

$$\frac{\ell_{i,1}}{T-\alpha_{i,1}} = \frac{\ell_{i,1}}{T-r_j}.$$

Remarking that we can write $\ell_{i,1} = \ell_i(1)$, altogether, the sum in question can be written

$$\begin{aligned}\sum_{s\geq 0} \frac{\ell(X_1^s)}{T^{s+1}} &= \sum_{j\in\mathfrak{e}} \frac{\sum_{i\in T_j} \ell_i(1)}{T-r_j} + \sum_{j\in\mathfrak{f}} \frac{D_j}{(T-r_j)^{x_j}} \\ &= \sum_{j\in\mathfrak{e}} \frac{\lambda_j(1)}{T-r_j} + \sum_{j\in\mathfrak{f}} \frac{D_j}{(T-r_j)^{x_j}}\end{aligned}$$

for some integers $\{x_j \mid j \in \mathfrak{f}\}$ and polynomials $\{D_j \mid j \in \mathfrak{f}\}$ such that $\deg(D_j) < x_j$ holds, and with D_j and $T - r_j$ coprime. In particular, the minimal polynomial of $(\ell(X_1^s))_{s\geq 0}$ is $N = \prod_{j\in\mathfrak{e}}(T-r_j)\prod_{j\in\mathfrak{f}}(T-r_j)^{x_j}$.

The minimal polynomial of the sequence $(\ell(X_1^s))_{s\geq 0}$ divides M, so that $x_j \leq \mu_j$ holds for all j in $\mathfrak{f}$. As a result, $\Omega((\ell(X_1^s))_{s\geq 0}, M)$ is well-defined and is given by

$$\begin{aligned}\Omega((\ell(X_1^s))_{s\geq 0}, M) = &\sum_{j\in\mathfrak{e}} \Big(\lambda_j(1) \prod_{\iota\in\mathfrak{e}-\{j\}} (T-r_\iota)\Big)\Big(\prod_{j\in\mathfrak{f}} (T-r_j)^{\mu_j}\Big) \\ &+ \sum_{j\in\mathfrak{f}} \Big((T-r_j)^{\mu_j - x_j} D_j \prod_{\iota\in\mathfrak{f}-\{j\}} (T-r_j)^{\mu_\iota}\Big)\Big(\prod_{j\in\mathfrak{e}} (T-r_j)\Big).\end{aligned}$$

This implies that

$$\Omega((\ell(X_1^s))_{s\geq 0}, M)(r_k) = \lambda_k(1) \prod_{\iota\in\mathfrak{e}-\{k\}} (r_k - r_\iota) \prod_{j\in\mathfrak{f}} (r_k - r_j)^{\mu_j} = \lambda_k(1) M'(r_k)$$

holds for all k in $\mathfrak{e}$. □

We now show how this result allows us to use sequences of the form $(\ell(X_1^s))_{s\geq 0}$ to compute a zero-dimensional parametrization of a subset V' of V. Precisely, we characterize the set V' as follows: for i in $\{1, \ldots, \kappa\}$, $\boldsymbol{\alpha}_i$ is in V' if and only if:

- for i' in $\{1,\dots,\kappa\}$, with $i' \neq i$, $\alpha_{i',1} \neq \alpha_{i,1}$;
- $\mathscr{Q}_i$ is a reduced algebra, or equivalently, I_i is radical (see Section 3.1 for the notation used here).

We denote by $\mathfrak{A} \subset \{1,\dots,\kappa\}$ the set of corresponding indices i, and we let $\mathfrak{B} = \{1,\dots,\kappa\} - \mathfrak{A}$, so that we have $V' = V_{\mathfrak{A}}$ and $V'' = V_{\mathfrak{B}}$. Remark that X_1 separates the points of V'.

Correspondingly, we define $\mathfrak{a}$ as the set of all indices j in $\{1,\dots,c\}$ such that σ_j is in $\mathfrak{A}$. In other words, j is in $\mathfrak{a}$ if and only if T_j has cardinality 1, so that $T_j = \{\sigma_j\}$, and $\mathscr{Q}_{\sigma_j}$ is reduced. The algorithm in this paragraph will compute a zero-dimensional parametrization of $V_{\mathfrak{A}}$; we will use the following lemma to perform the decomposition.

Lemma 5.3. *Let j be in $\{1,\dots,c\}$ such that $\mu_j = 1$, let λ be a linear form over $\prod_{i\in T_j} \mathscr{Q}_i$ and let $Y = y_2X_2 + \cdots + y_nX_n$, for some $y_2,\dots,y_n$ in $\mathbb{K}$. Define constants a, b, c in $\overline{\mathbb{K}}$ by*

$$a = \lambda(1), \quad b = \lambda(Y), \quad c = \lambda(Y^2).$$

Then, j is in $\mathfrak{a}$ if and only if, for a generic choice of λ and Y, $ac = b^2$.

Proof. The assumption that $\mu_j = 1$ means that for all i in T_j, $\nu_i = 1$. The linear form λ can be uniquely written as a sum $\lambda = \sum_{i\in T_j} \ell_i$, where each ℓ_i is in $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_i, \overline{\mathbb{K}})$. The fact that all ν_i are equal to 1 then implies that each ℓ_i takes the form

$$\ell_i : f \mapsto (\Lambda_i(f))(\boldsymbol{\alpha}_i),$$

where Λ_i is a differential operator that does not involve $\partial/\partial X_1$. Thus, as in Eq. (7), we can write a general Λ_i of this form as

$$\Lambda_i : f \mapsto u_{i,1}f + \sum_{2\le r\le n} P_{i,r}(u_{i,2},\dots,u_{i,D_i})\frac{\partial}{\partial X_j}f + \sum_{2\le r\le s\le n} P_{i,r,s}(u_{i,2},\dots,u_{i,D_i})\frac{\partial^2}{\partial X_j \partial X_k}f + \tilde{\Lambda}_i(f),$$

where all terms in $\tilde{\Lambda}_i$ have order at least 3, $\boldsymbol{u}_i = (u_{i,1},\dots,u_{i,D_i})$ are parameters and $(P_{i,r})_{2\le r\le n}$ and $(P_{i,r,s})_{2\le r\le s\le n}$ are linear forms in $u_{i,2},\dots,u_{i,D_i}$. We obtain

$$\begin{aligned}
\Lambda_i(1) &= u_{i,1}\\
\Lambda_i(Y) &= u_{i,1}Y + \sum_{2\le r\le n} P_{i,r}(u_{i,2},\dots,u_{i,D_i})y_r\\
\Lambda_i(Y^2) &= u_{i,1}Y^2 + 2Y\sum_{2\le r\le n} P_{i,r}(u_{i,2},\dots,u_{i,D_i})y_r + 2\sum_{2\le r\le s\le n} P_{i,r,s}(u_{i,2},\dots,u_{i,D_i})y_ry_s,
\end{aligned}$$

which gives

$$\begin{aligned} a &= \sum_{i\in T_j} u_{i,1} \\ b &= \sum_{i\in T_j} u_{i,1}Y(\boldsymbol{\alpha}_i) + \sum_{i\in T_j, 2\le r\le n} P_{i,r}(u_{i,2},\ldots,u_{i,D_i})y_r \\ c &= \sum_{i\in T_j} u_{i,1}Y(\boldsymbol{\alpha}_i)^2 + 2\sum_{i\in T_j, 2\le r\le n} Y(\boldsymbol{\alpha}_i)P_{i,r}(u_{i,2},\ldots,u_{i,D_i})y_r \\ &\qquad + 2\sum_{i\in T_j, 2\le r\le s\le n} P_{i,r,s}(u_{i,2},\ldots,u_{i,D_i})y_r y_s. \end{aligned}$$

Suppose first that j is in $\mathfrak{a}$. Then, $T_j = \{\sigma_j\}$, so we have only one term Λ_{σ_j} to consider, and $\mathscr{Q}_{\sigma_j}$ is reduced, so that all coefficients $P_{\sigma_j,r}$ and $P_{\sigma_j,r,s}$ vanish. Thus, we are left in this case with

$$a = u_{\sigma_j,1}, \quad b = u_{\sigma_j,1}Y(\boldsymbol{\alpha}_{\sigma_j}), \quad c = u_{\sigma_j,1}Y(\boldsymbol{\alpha}_{\sigma_j})^2,$$

so that we have $ac = b^2$, for *any* choice of λ and Y. Now, we suppose that j is not in $\mathfrak{a}$, and we prove that for a generic choice of λ and Y, $ac - b^2$ is non-zero. The quantity $ac - b^2$ is a polynomial in the parameters $(\boldsymbol{u}_i)_{i\in T_j}$, and $(y_i)_{i\in\{2,\ldots,n\}}$, and we have to show that it is not identically zero. We discuss two cases; in both of them, we prove that a suitable specialization of $ac - b^2$ is non-zero.

Suppose first that for at least one index σ in T_j, $\mathscr{Q}_\sigma$ is not reduced. In this case, there exists as least one index ρ in $\{2,\ldots,n\}$ such that $P_{\sigma,\rho}(u_{\sigma,2},\ldots,u_{\sigma,D_\sigma})$ is not identically zero. Let us set all $\boldsymbol{u}_{\sigma'}$ to 0, for σ' in $T_j - \{\sigma\}$, as well as $u_{\sigma,1}$, and all y_r for $r \neq \rho$. Then, under this specialization, $ac - b^2$ becomes $-(P_{\sigma,\rho}(u_{\sigma,2},\ldots,u_{\sigma,D_\sigma})y_\rho)^2$, which is non-zero, so that $ac - b^2$ itself must be non-zero.

Else, since j is not in $\mathfrak{a}$, we can assume that T_j has cardinality at least 2, with $\mathscr{Q}_\sigma$ reduced for all σ in T_j (so that $P_{\sigma,r}$ and $P_{\sigma,r,s}$ vanish for all such σ and all r, s). Suppose that σ and σ' are two indices in T_j; we set all indices $u_{\sigma'',1}$ to zero, for σ'' in $T_j - \{\sigma,\sigma'\}$. We are left with

$$a = u_{\sigma,1} + u_{\sigma',1}, \quad b = u_{\sigma,1}Y(\boldsymbol{\alpha}_\sigma) + u_{\sigma',1}Y(\boldsymbol{\alpha}_{\sigma'}), \quad c = u_{\sigma,1}Y(\boldsymbol{\alpha}_\sigma)^2 + u_{\sigma',1}Y(\boldsymbol{\alpha}_{\sigma'})^2.$$

Then, $ac - b^2$ is equal to $2u_{\sigma,1}u_{\sigma',1}(Y(\boldsymbol{\alpha}_\sigma) - Y(\boldsymbol{\alpha}_{\sigma'}))^2$, which is non-zero, since $\boldsymbol{\alpha}_\sigma \neq \boldsymbol{\alpha}_{\sigma'}$. □

The previous lemmas allow us to write Algorithm BlockParametrizationX$_1$. After computing M, we determine its factor $F = \prod_{j\in\{1,\ldots,c\},\mu_j=1}(T - r_j)$, which is obtained by taking the squarefree part of M and dividing it by its gcd with $\gcd(M, M')$. We split this polynomial further using the previous lemma in order to find $\prod_{j\in\mathfrak{a}}(T - r_j)$, and we conclude using the same kind of calculations as in Algorithm BlockParametrization.

Algorithm 5 BlockParametrizationX$_1(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, Y)$

Input:

- multiplication matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ in $\mathbb{K}^{D\times D}$
- $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{K}^{D\times m}$, for some block dimension $m \in \{1, \ldots, D\}$
- $Y = y_2X_2 + \cdots + y_nX_n$

Output:

- polynomials $((F, G_1, \ldots, G_n), X_1)$, with $F, G_1, \ldots, G_n$ in $\mathbb{K}[T]$

1. compute $\boldsymbol{L}_s = \boldsymbol{U}^\perp \boldsymbol{M}_1^s$ for $s = 0, \ldots, 2d-1$, with $d = \lceil D/m \rceil$
2. compute $\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}} = \boldsymbol{L}_s\boldsymbol{V}$ for $s = 0, \ldots, 2d-1$
3. compute a minimal matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ of $(\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}})_{0\le s<2d}$
4. let M be the largest invariant factor of $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$
5. let F be the squarefree part of M
6. let $F = F/\gcd(F, \gcd(M, M'))$
7. let $\boldsymbol{a}_1 = [M\ 0\ \cdots\ 0](\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}})^{-1}$
8. let $\boldsymbol{N} = y_2\boldsymbol{M}_2 + \cdots + y_n\boldsymbol{M}_n$
9. **for** $i = 0, 1, 2$ **do**
   let $A_i = \mathsf{ScalarNumerator}(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}, M, \boldsymbol{N}^i\boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{L}_s)_{0\le s<d})$
10. let $F = \gcd(F, A_0A_2 - A_1^2)$
11. **for** $i = 2, \ldots, n$ **do**
    let $A_{X_i} = \mathsf{ScalarNumerator}(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}, M, \boldsymbol{M}_i\boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{L}_s)_{0\le s<d})$
12. **return** $((F, T, A_{X_2}/A_0 \bmod F, \ldots, A_{X_n}/A_0 \bmod F), X_1)$

Lemma 5.4. *For generic choices of $\boldsymbol{U}$, $\boldsymbol{V}$ and Y, the output $((F, G_1, \ldots, G_n), X_1)$ of* BlockParametrizationX$_1$ *is a zero-dimensional parametrization of $V_{\mathfrak{A}}$.*

Proof. As in the case of BlockParametrization, for generic choices of $\boldsymbol{U}$ and $\boldsymbol{V}$, the degree bound $\deg(\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}) \le d$ holds and M is the minimal polynomial of X_1 in $\mathscr{Q}$; hence, after Step 6, we have $F = \prod_{j\in\{1,\ldots,c\},\mu_j=1}(T - r_j)$.

The calculation of A_0, A_1, A_2 and $A_{X_2}, \ldots, A_{X_n}$ is justified as in BlockParametrization, by means of Lemma 2.13. For $i = 1, \ldots, D$, let $u_{i,1}$ is the entry at position $(i, 1)$ in $\boldsymbol{U}$, and define the linear form $\ell : \mathscr{Q} \to \mathbb{K}$ by

$$\ell(f) = \sum_{i=1}^{D} f_i u_{i,1}, \quad \text{for} \quad f = \sum_{i=1}^{D} f_i b_i.$$

Then, the above lemma proves that $A_i = \Omega((\ell(Y^iX_1^s))_{s\ge0}, M)$ for $i = 0, 1, 2$ and $A_{X_i} = \Omega((\ell(X_iX_1^s))_{s\ge0}, M)$ for $i = 2, \ldots, n$.

Take then j in $\{1, \ldots, c\}$ such that $\mu_j = 1$, that is, a root of F as computed at Step 6. By Lemma 5.2, for $i = 0, 1, 2$, we have $A_i(r_j) = M'(r_j)(Y^i \cdot \lambda_j)(1)$, where $\lambda_j = \sum_{i\in T_j} \ell_i$, where the ℓ_i's are the components of ℓ, and where $Y^i \cdot \lambda_j$ is the linear form $f \mapsto \lambda_j(Y^i f)$.

As a result, the value of $A_0A_2 - A_1^2$ at r_j is (up to the non-zero factor $M'(r_j)^2$) equal to the quantity $ac - b^2$ defined in Lemma 5.3, so for a generic choice of ℓ (that is, of $\boldsymbol{U}$) and Y, it vanishes if and only if j is in $\mathfrak{a}$. Thus, after Step 10, F is equal to $\prod_{j\in\mathfrak{a}}(T - r_j)$.

The last step is to compute the zero-dimensional parametrization of $V_{\mathfrak{A}}$. This is done using again Lemma 5.2. Indeed, for j in $\mathfrak{a}$, T_j is simply equal to $\{\sigma_j\}$, so that we have, for $i = 2, \ldots, n$,

$$A_0(r_j) = M'(r_j)\lambda_j(1) \quad \text{and} \quad A_{X_i}(r_j) = M'(r_j)(X_i \cdot \lambda_j)(1) = M'(r_j)\lambda_j(X_i),$$

where as above, $X_i \cdot \lambda_j$ is the linear form $f \mapsto \lambda_j(X_i f)$. Now, since j is in $\mathfrak{a}$, $\mathscr{Q}_{\sigma_j}$ is reduced, so that there exists a constant $\lambda_{j,1}$ such that for all f in $\overline{\mathbb{K}}[X_1, \ldots, X_n]$, $\lambda_j(f)$ takes the form $\lambda_{j,1} f(\boldsymbol{\alpha}_{\sigma_j})$. This shows that, as claimed,

$$\frac{A_{X_j}(r_j)}{A_0(r_j)} = \frac{M'(r_j)\lambda_{j,1}\alpha_{\sigma_j,i}}{M'(r_j)\lambda_{j,1}} = \alpha_{\sigma_j,i},$$

since $M'(r_j)$ is non-zero. For $i = 1$, since we use X_1 as a separating variable for $V_{\mathfrak{A}}$, we simply insert the polynomial T into our list. □

The cost analysis is the same as that of Algorithm BlockParametrization, the crucial difference being that the density ρ_1 of $\boldsymbol{M}_1$ plays the role of the density ρ of $\boldsymbol{M}$ used in that algorithm.

5.3 Computing correction matrices

Next, we describe an operation of decomposition for linear forms $\mathscr{Q} \to \mathbb{K}$; this is essentially akin to the Chinese Remainder Theorem. We then use it to compute the sequences of correction matrices $(\boldsymbol{\Delta}_s), (\boldsymbol{\Delta}'_s), (\boldsymbol{\Delta}''_s)$ defined in Algorithm BlockParametrizationWithSplitting.

As a preamble, we introduce the notation $\mathsf{P}(r, t)$ for the cost of a *power projection* operation, as defined in [43, 44]: given a polynomial F in $\mathbb{K}[T]$ of degree r, a linear form $\ell : \mathbb{K}[T]/F \to \mathbb{K}$, and H in $\mathbb{K}[T]/F$, the goal is to compute $(\ell(H^s))_{0\le s<t}$, for some upper bound t. We denote this operation by PowerProjection(F, H, ℓ, t); this is essentially the analogue for univariate polynomials of the Krylov computations that we heavily rely on in this paper. Here, ℓ is represented by the vector $(\ell(1), \ell(T), \ldots, \ell(T^{r-1}))$.

In [43, Theorem 4], Shoup showed that this can be done in $\mathsf{P}(r, t) = O(r^{(\omega-1)/2}t)$ operations in $\mathbb{K}$ for $t \le r$. For $t \ge r$, we first solve the problem up to index r in time $O(r^{(\omega+1)/2})$; then we use the fact that the sequence $(\ell(H^s))_{s\ge 0}$ is linearly recurrent to compute all further values in time $O(t\mathsf{M}(r)/r)$, as for instance in [7, Proposition 1]. Thus, for $t \ge r$, we take $\mathsf{P}(r, t) = O(r^{(\omega+1)/2} + t\mathsf{M}(r)/r)$.

Let $\mathfrak{A}$ and $\mathfrak{B}$ be defined as in the previous subsection, and let $D_{\mathfrak{A}}$ be the number of points in $V_{\mathfrak{A}}$. Since $\mathscr{Q}_i$ is a reduced algebra for all i in $\mathfrak{A}$, $D_{\mathfrak{A}}$ is also the dimension of $\mathscr{Q}_{\mathfrak{A}} = \prod_{i\in\mathfrak{A}} \mathscr{Q}_i$ and $\mathscr{Q}_{\mathfrak{B}} = \prod_{i\in\mathfrak{B}} \mathscr{Q}_i$ has dimension $D_{\mathfrak{B}} = D - D_{\mathfrak{A}}$.

Consider a linear form $\ell : \mathscr{Q} \to \mathbb{K}$; we still denote ℓ for its extension $\mathscr{Q} \otimes_{\mathbb{K}} \overline{\mathbb{K}} \to \overline{\mathbb{K}}$. It can then be decomposed as $\ell = \ell_{\mathfrak{A}} + \ell_{\mathfrak{B}}$, with $\ell_{\mathfrak{A}} : \mathscr{Q}_{\mathfrak{A}} \to \overline{\mathbb{K}}$ and $\ell_{\mathfrak{B}} : \mathscr{Q}_{\mathfrak{B}} \to \overline{\mathbb{K}}$. Remark that the support of $\ell_{\mathfrak{B}}$ is contained in $\mathfrak{B}$, and actually equal to $\mathfrak{B}$ for a generic ℓ.

Suppose that we are given the minimal polynomial M of X_1 in $\mathscr{Q}$, the numerator $C = \Omega((\ell(X_1^s))_{s\geq 0}, M)$, as well as the zero-dimensional parametrization $((F, G_1, \ldots, G_n), X_1)$ of $V_{\mathfrak{A}}$ computed in the previous paragraph. Given $X = t_1X_1 + \cdots + t_nX_n$, and an upper bound τ, we show how to compute the values $\ell_{\mathfrak{A}}(X^s)$, for $s = 0, \ldots, \tau - 1$.

Let $E = M/F$; the division is exact and E and F are coprime, by construction. The equality $\ell = \ell_{\mathfrak{A}} + \ell_{\mathfrak{B}}$ implies an equality between generating series

$$\begin{aligned}\sum_{s\geq 0} \frac{\ell(X_1^s)}{T^{s+1}} &= \sum_{s\geq 0} \frac{\ell_{\mathfrak{A}}(X_1^s)}{T^{s+1}} + \sum_{s\geq 0} \frac{\ell_{\mathfrak{B}}(X_1^s)}{T^{s+1}} \\ &= \frac{A}{F} + \frac{B}{E},\end{aligned}$$

for some polynomials A, B in $\mathbb{K}[T]$, with $\deg(A) < \deg(F)$ and $\deg(B) < \deg(E)$. With $C = \Omega((\ell(X_1^s))_{s\geq 0}, M)$, we deduce the equality

$$\frac{C}{M} = \frac{A}{F} + \frac{B}{E},$$

from which we find $A = C/E \bmod F$. Knowing A and F allows us to compute the values $\ell_{\mathfrak{A}}(X_1^s)$, for $s = 0, \ldots, D_{\mathfrak{A}} - 1$, by Laurent series expansion. Since $\mathscr{Q}_{\mathfrak{A}}$ is reduced, we have $X = t_1G_1 + \cdots t_nG_n$ in $\mathscr{Q}_{\mathfrak{A}}$, where $G_1, \ldots, G_n$ are polynomials in the zero-dimensional parametrization of $V_{\mathfrak{A}}$. As a result, we can finally compute $\ell_{\mathfrak{A}}(X^s)$, for $s = 0, \ldots, \tau - 1$ by applying our algorithm for univariate power projection to $G = t_1G_1 + \cdots t_nG_n$.

The following pseudo-code summarizes this discussion; the cost of the algorithm is $O(\mathsf{M}(D_{\mathfrak{A}}) \log(D_{\mathfrak{A}}) + \mathsf{P}(D_{\mathfrak{A}}, \tau) + nD_{\mathfrak{A}})$ operations in $\mathbb{K}$, where the first term accounts for the cost of the first three steps, $\mathsf{P}(D_{\mathfrak{A}}, \tau)$ is the cost of power projection and the term $O(nD_{\mathfrak{A}})$ describes the cost of computing G as defined above.

Algorithm 6 $\mathsf{Decompose}(M, C, ((F, G_1, \ldots, G_n), X_1), X, \tau)$

Input:

- minimal polynomial M of X_1 in $\mathscr{Q}$
- numerator $C = \Omega((\ell(X_1^s))_{s\geq 0}, M)$
- zero-dimensional parametrization $((F, G_1, \ldots, G_n), X_1)$ of $V_{\mathfrak{A}}$
- $X = t_1X_1 + \cdots + t_nX_n$
- a bound τ

Output:

- $\ell_{\mathfrak{A}}(X^s)$, for $s = 0, \ldots, \tau - 1$

1. let $E = M/F$
2. let $A = C/E \bmod F$
3. compute the first $D_{\mathfrak{A}}$ terms $(v_0, \ldots, v_{D_{\mathfrak{A}}-1})$ of the Laurent series A/F
4. **return** $\mathsf{PowerProjection}(F, t_1G_1 + \cdots + t_nG_n, (v_0, \ldots, v_{D_{\mathfrak{A}}-1}), \tau)$

We will use this algorithm to compute the correction matrices to be used in Algorithm BlockParametrizationResidual. We assume that we have stored various quantities computed in Algorithm BlockParametrizationX_1: the sequence of matrices $(\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}})_{0\le s<2d}$, the matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$, the minimal polynomial M of X_1, and of course the zero-dimensional parametrization $((F, G_1, \ldots, G_n), X_1)$.

Let $\boldsymbol{U}$ and $\boldsymbol{V}$ be the blocking matrices used in BlockParametrizationX_1. For $i = 1, \ldots, m$, we let $\ell_i : \mathscr{Q} \to \mathbb{K}$ be the linear form whose values on the basis $\mathscr{B} = (b_1, \ldots, b_D)$ are given by the i-th column of $\boldsymbol{U}$. In other words, $\ell_i(f) = \sum_{j=1}^{D} f_j u_{j,i}$, for $f = \sum_{j=1}^{D} f_j b_j$. Similarly, for $j = 1, \ldots, m$, we let γ_j be the element of $\mathscr{Q}$ whose coefficient vector on the basis $\mathscr{B}$ is the j-th column of $\boldsymbol{V}$.

Hereafter, we also write $d_{\mathfrak{B}} = \lceil D_{\mathfrak{B}}/m \rceil$, in analogy with the definition of d used so far.

- To each (i, j) in $\{1, \ldots, m\} \times \{1, \ldots, m\}$ is associated a linear form $\ell_{i,j} : \mathscr{Q} \to \mathbb{K}$ defined by $\ell_{i,j}(f) = \ell_i(\gamma_j f)$ for all f in $\mathscr{Q}$. Then, the entry (i, j) of the matrix sequence $(\boldsymbol{U}^\perp \boldsymbol{M}_1^s \boldsymbol{V})_{s\ge 0}$ is the scalar sequence $(\ell_{i,j}(X_1^s))_{s\ge 0}$.

  For all such (i, j), since we know the minimal polynomial M of X_1, we can compute the scalar numerator $C_{i,j} \in \mathbb{K}[T]$ associated to $\ell_{i,j}$ and M. This is done by applying Algorithm ScalarNumerator of Section 2.4, using the row vector $\boldsymbol{a}_i$ defined in that section, together with the sequence of matrices $\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}}$ and the matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ computed in Algorithm BlockParametrizationX_1.

  Once $C_{i,j}$ is known, we can call Decompose, which allows us to compute $\ell_{i,j,\mathfrak{A}}(X^s)$, for $s = 0, \ldots, 2d_{\mathfrak{B}} - 1$. We can then construct the sequence $(\boldsymbol{\Delta}_s)_{0\le s<2d_{\mathfrak{B}}}$ of matrices in $\mathbb{K}^{m\times m}$ by setting the (i, j)-th entry of $\boldsymbol{\Delta}_s$ to be $\ell_{i,j,\mathfrak{A}}(X^s)$.

- To each (i, k) in $\{1, \ldots, m\} \times \{1, \ldots, n\}$ is associated a linear form $\ell'_{i,k} : \mathscr{Q} \to \mathbb{K}$ defined by $\ell'_{i,k}(f) = \ell_i(X_k f)$ for all f in $\mathscr{Q}$. Then, the ith entry of the sequence of column vectors $(\boldsymbol{U}^\perp \boldsymbol{M}_1^s \boldsymbol{M}_k \boldsymbol{\varepsilon}_1)_{s\ge 0}$ is the scalar sequence $(\ell'_{i,k}(X_1^s))_{s\ge 0}$, where $\boldsymbol{\varepsilon}_1$ is the column vector $[1\ 0 \cdots\ 0]^\perp$ we already used several times.

  Proceeding as before, we construct the sequence of $m \times n$ matrices $(\boldsymbol{\Delta}'_s)_{0\le s<d_{\mathfrak{B}}}$ by setting the (i, k)-th entry of $\boldsymbol{\Delta}'_s$ to be $\ell'_{i,k,\mathfrak{A}}(X^s)$. Note that we will only need $d_{\mathfrak{B}}$ entries in this sequence.

- Finally, we apply this process to the linear forms ℓ_i themselves; they are such that the ith entry of the sequence of column vectors $(\boldsymbol{U}^\perp \boldsymbol{M}_1^s \boldsymbol{\varepsilon}_1)_{s\ge 0}$ is the scalar sequence $(\ell_i(X_1^s))_{s\ge 0}$. Using again ScalarNumerator and Decompose, we construct the sequence of column vectors $(\boldsymbol{\Delta}''_s)_{0\le s<d_{\mathfrak{B}}}$ by setting the i-th entry of $\boldsymbol{\Delta}''_s$ to $\ell_{i,\mathfrak{A}}(X^s)$.

In terms of cost, we need to compute all vectors $\boldsymbol{a}_i$, for $i = 1, \ldots, m$; using the analysis of Eq. (5), this takes $O(m^\omega \mathsf{M}(D) \log(D) \log(m))$ operations in $\mathbb{K}$. Then, the total time spent in ScalarNumerator is $m(m + n + 1)$ times the cost reported in Section 2.4, which was $O(D^2 + m\mathsf{M}(D))$; similarly, the total cost incurred by Decompose is $m(m + n + 1)$ times the cost of a single call, which was reported above.

5.4 Describing the residual set

We finally describe Algorithm BlockParametrizationResidual. Let $\mathfrak{A}$ and $\mathfrak{B}$ be as in the previous section. This part of the main algorithm computes a zero-dimensional parametrization of the residual set $V_{\mathfrak{B}} = V - V_{\mathfrak{A}}$. For this, we are going to call a modified version Algorithm BlockParametrization, where we update the values of our matrix sequences before computing the minimal matrix generator, using the correction matrices defined just above.

The resulting algorithm is as follows. A superficial difference with BlockParametrization is that names of the main variables have been changed (so as not to create any confusion with those used in BlockParametrizationX_1). More importantly, using the correction matrices makes it possible for us to compute fewer terms in the sequences, namely only $2\lceil D_{\mathfrak{B}}/m\rceil$ and $\lceil D_{\mathfrak{B}}/m\rceil$, respectively. Hence, if $D_{\mathfrak{B}} \ll D$ (that is, $V_{\mathfrak{B}}$ contains few points, with small multiplicities), this last stage of the algorithm will be fast.

The algorithm uses a subroutine called ScalarNumeratorCorrected at Steps 8 and 9. It is similar to Algorithm ScalarNumerator of Section 2.4, with a minor difference: instead of computing the vectors $\boldsymbol{D}_s\boldsymbol{\varepsilon}_1$, resp. $\boldsymbol{D}_s\boldsymbol{M}_i\boldsymbol{\varepsilon}_1$, at the first step of ScalarNumerator, it computes $\boldsymbol{D}_s\boldsymbol{\varepsilon}_1 - \boldsymbol{\Delta}''_s$, resp. $\boldsymbol{D}_s\boldsymbol{M}_i\boldsymbol{\varepsilon}_1 - \boldsymbol{\Delta}'_{s,i}$, where $\boldsymbol{\Delta}'_{s,i}$ is the ith column of $\boldsymbol{\Delta}'_s$.

Algorithm 7 BlockParametrizationResidual$(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, (\boldsymbol{\Delta}_s), (\boldsymbol{\Delta}'_s), (\boldsymbol{\Delta}''_s), X)$

Input:

- $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ defined as above
- $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{K}^{D\times m}$, for some block dimension $m \in \{1, \ldots, D\}$
- sequences of correction matrices $(\boldsymbol{\Delta}_s), (\boldsymbol{\Delta}'_s), (\boldsymbol{\Delta}''_s)$
- $X = t_1X_1 + \cdots + t_nX_n$

Output:

- polynomials $((R, W_1, \ldots, W_n), X)$, with $R, W_1, \ldots, W_n$ in $\mathbb{K}[T]$

1. let $\boldsymbol{M} = t_1\boldsymbol{M}_1 + \cdots + t_n\boldsymbol{M}_n$
2. compute $\boldsymbol{D}_s = \boldsymbol{U}^\perp\boldsymbol{M}^s$ for $s = 0, \ldots, 2d_{\mathfrak{B}} - 1$, with $d_{\mathfrak{B}} = \lceil D_{\mathfrak{B}}/m\rceil$
3. compute $\boldsymbol{E}_{s,\boldsymbol{U},\boldsymbol{V}} = \boldsymbol{D}_s\boldsymbol{V} - \boldsymbol{\Delta}_s$ for $s = 0, \ldots, 2d_{\mathfrak{B}} - 1$
4. compute a minimal matrix generator $\boldsymbol{S}_{\boldsymbol{U},\boldsymbol{V}}$ of $(\boldsymbol{E}_{s,\boldsymbol{U},\boldsymbol{V}})_{0\le s<2d_{\mathfrak{B}}}$
5. let S be the largest invariant factor of $\boldsymbol{S}_{\boldsymbol{U},\boldsymbol{V}}$
6. let R be the squarefree part of S
7. let $\boldsymbol{a}_1 = [S\ 0\ \cdots\ 0](\boldsymbol{S}_{\boldsymbol{U},\boldsymbol{V}})^{-1}$
8. let $C_1 =$ ScalarNumeratorCorrected$(\boldsymbol{S}_{\boldsymbol{U},\boldsymbol{V}}, S, \boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{D}_s)_{0\le s<d_{\mathfrak{B}}}, (\boldsymbol{\Delta}''_s)_{0\le s<d_{\mathfrak{B}}})$
9. **for** $i = 1, \ldots, n$ **do**
   let $C_{X_i} =$ ScalarNumeratorCorrected$(\boldsymbol{S}_{\boldsymbol{U},\boldsymbol{V}}, S, \boldsymbol{M}_i\boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{D}_s)_{0\le s<d_{\mathfrak{B}}}, (\boldsymbol{\Delta}'_s)_{0\le s<d_{\mathfrak{B}}})$
10. **return** $((R, C_{X_1}/C_1 \bmod R, \ldots, C_{X_n}/C_1 \bmod R), X)$

Let us prove correctness. Since $\mathscr{Q} = \mathscr{Q}_{\mathfrak{A}} \times \mathscr{Q}_{\mathfrak{B}}$, we may assume without loss of generality that our multiplication matrices are block diagonal, with two blocks corresponding respectively to bases of $\mathscr{Q}_{\mathfrak{A}}$ and $\mathscr{Q}_{\mathfrak{B}}$; if not, apply a change of basis to reduce to this situation, updating $\boldsymbol{U}$ and $\boldsymbol{V}$ accordingly.

We denote by $\boldsymbol{M}_{\mathfrak{A}}$ and $\boldsymbol{M}_{\mathfrak{B}}$ the two blocks on the diagonal of matrix $\boldsymbol{M}$. The projection matrices can also be divided into blocks, namely as

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U}_{\mathfrak{A}} \\ \boldsymbol{U}_{\mathfrak{B}} \end{bmatrix} \quad \text{and} \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_{\mathfrak{A}} \\ \boldsymbol{V}_{\mathfrak{B}} \end{bmatrix},$$

and we have $\boldsymbol{U}^{\perp}\boldsymbol{M}^{s}\boldsymbol{V} = \boldsymbol{U}_{\mathfrak{A}}^{\perp}\boldsymbol{M}_{\mathfrak{A}}^{s}\boldsymbol{V}_{\mathfrak{A}} + \boldsymbol{U}_{\mathfrak{B}}^{\perp}\boldsymbol{M}_{\mathfrak{B}}^{s}\boldsymbol{V}_{\mathfrak{B}}$ for $s \geq 0$. The first summand is none other than the matrix $\boldsymbol{\Delta}_s$, so that $\boldsymbol{E}_{s,\boldsymbol{U},\boldsymbol{V}}$ is equal to $\boldsymbol{U}_{\mathfrak{B}}^{\perp}\boldsymbol{M}_{\mathfrak{B}}^{s}\boldsymbol{V}_{\mathfrak{B}}$. These are thus the kind of Krylov matrices we would obtain if we were working with a basis of $\mathscr{Q}_{\mathfrak{B}}$, and shows that we have enough terms to compute a minimal matrix generator $\boldsymbol{S}_{\boldsymbol{U},\boldsymbol{V}}$, at least for generic $\boldsymbol{U}$ and $\boldsymbol{V}$. Similarly, S is generically the minimal polynomial of X_1 in $Q_{\mathfrak{B}}$, and R its squarefree part.

The same considerations justify the computation of C_1 and $C_{X_1}, \ldots, C_{X_n}$. Indeed, subtracting the correction matrices implies that

$$C_1 = \Omega((\ell_1(X^s) - \ell_{1,\mathfrak{A}}(X^s))_{s\geq 0}, S) = \Omega((\ell_{1,\mathfrak{B}}(X^s))_{s\geq 0}, S)$$

and

$$C_{X_i} = \Omega((\ell_1(X_iX^s) - \ell_{1,\mathfrak{A}}(X_iX^s))_{s\geq 0}, S) = \Omega((\ell_{1,\mathfrak{B}}(X_iX^s))_{s\geq 0}, S),$$

for $i = 1, \ldots, n$. As shown in Section 3.3, these are precisely the polynomials we need to compute a zero-dimensional parametrization of $V_{\mathfrak{B}}$.

The cost analysis is similar to that of Algorithm BlockParametrization, with the important exception that the sequence length $d = \lceil D/m \rceil$ can then be replaced by the (hopefully much smaller) $d_{\mathfrak{B}} = \lceil D_{\mathfrak{B}}/m \rceil$.

5.5 Experimental results

The algorithms in this section were implemented using the same platform as the one in the previous section, using in particular NTL's built-in implementation of power projection.

In Table 2, we give the ratio of the runtime of BlockParametrizationWithSplitting to that of our first algorithm, BlockParametrization, for each input: numbers less than 1 indicate a speed-up. The last column shows the number of points in $V_{\mathfrak{A}}$, that is, $D_{\mathfrak{A}}$, compared to the total degree of I, which is $D = D_{\mathfrak{A}} + D_{\mathfrak{B}}$. The inputs, the machine used for timings and the prime field are the same as in Section 4.3.

The performance of BlockParametrizationWithSplitting depends on the density of $\boldsymbol{M}_1$ and the number of points $D_{\mathfrak{A}}$. For generic inputs, with no multiplicity, $D_{\mathfrak{A}} = D$, so we will actually not need to spend any time computing correction matrices, or running BlockParametrizationResidual. On the other hand, in the worst case, if $D_{\mathfrak{A}} = 0$, so that $D_{\mathfrak{B}} = D$, then Algorithm BlockParametrizationWithSplitting may take more than twice as long as BlockParametrization (due to the two calls to respectively BlockParametrizationX_1 and BlockParametrizationResidual, together with the overhead induced by power projection).

This unlucky case was seldomly seen in our experiments, since the systems with multiplicities generated randomly had few multiple points, and thus were favorable to us. An

unfavourable case is system sot1, where $V_{\mathfrak{A}}$ only accounts for 1012 points out of 7682 points in the variety.

Table 2: Comparison of BlockParametrizationWithSplitting and BlockParametrization

name	n	D	$m=1$	$m=3$	$m=6$	$D_{\mathfrak{A}}/D$
rand1-26	3	17576	0.453	0.384	0.65	17576/17576
rand1-28	3	21952	0.438	0.435	0.562	21952/21952
rand1-30	3	27000	0.429	0.577	0.608	27000/27000
rand2-10	4	10000	0.437	0.462	0.49	10000/10000
rand2-11	4	14641	0.423	0.566	0.435	14641/14641
rand2-12	4	20736	0.431	0.437	0.399	20736/20736
mixed1-22	3	10864	0.49	0.568	0.791	10648/10675
mixed1-23	3	12383	0.477	0.546	0.655	12167/12194
mixed1-24	3	14040	0.463	0.514	0.613	13824/13851
mixed2-10	4	10256	0.43	0.482	0.626	10000/10016
mixed2-11	4	14897	0.414	0.408	0.521	14641/14657
mixed2-12	4	20992	0.416	0.438	0.416	20736/20752
mixed3-12	12	4109	0.453	0.513	0.664	4096/4097
mixed3-13	13	8206	0.435	0.454	0.471	8192/8193
eco12	12	1024	0.446	0.572	0.602	1024/1024
sot1	5	8694	1.31	1.84	2.37	1012/8694
W1-6-5-2	5	18000	0.462	0.471	0.472	18000/18000
W1-4-6-2	6	6480	0.452	0.474	0.57	6480/6480
katsura10	11	1024	0.557	0.661	0.652	1024/1024

References

[1] M. E. Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Zeroes, multiplicities and idempotents for zerodimensional systems. In *MEGA'94*, volume 142 of *Progress in Mathematics*, pages 1–15. Birkhäuser, 1996.

[2] M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of the F5 Gröbner basis algorithm. *J. Symbolic Comput.*, 70:49–70, 2015.

[3] E. Becker, T. Mora, M. Marinari, and C. Traverso. The shape of the Shape Lemma. In *ISSAC'94*, pages 129–133. ACM, 1994.

[4] E. Becker and T. Wörmann. Radical computations of zero-dimensional ideals and real root counting. *Mathematics and Computers in Simulation*, 42(4-6):561–569, 1996.

[5] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM J. Matrix Anal. Appl.*, 15(3):804–823, 1994.

[6] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.

[7] A. Bostan, P. Flajolet, B. Salvy, and É. Schost. Fast computation of special resultants. *J. Symbolic Comput.*, 41(1):1–29, 2006.

[8] A. Bostan, B. Salvy, and É. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. *Appl. Algebra Engrg. Comm. Comput.*, 14:239–272, 2003.

[9] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, 1(3):259–295, 1980.

[10] D. Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Math. Comp.*, 62(205):333–350, 1994.

[11] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reductions to zero (F5). In *ISSAC'02*, pages 75–83. ACM, 2002.

[12] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Polynomial systems solving by fast linear algebra. https://hal.archives-ouvertes.fr/hal-00816724, 2013.

[13] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Sub-cubic change of ordering for Gröbner basis: a probabilistic approach. In *ISSAC'14*, pages 170–177. ACM, 2014.

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

[15] J.-C. Faugère and C. Mou. Sparse FGLM algorithms. *J. Symbolic Comput.*, 80(8):538–569, 2017.

[16] J.-C. Faugère, M. Safey El Din, and P.-J. Spaenlehauer. On the complexity of the generalized MinRank problem. *J. Symbolic Comput.*, pages 30–58, 2013.

[17] FFLAS-FFPACK-Team. *FFLAS-FFPACK: Finite Field Linear Algebra Subroutines / Package*, v2.2.2 edition, 2016. http://github.com/linbox-team/fflas-ffpack.

[18] J. Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, third edition, 2013.

[19] P. Gianni and T. Mora. Algebraic solution of systems of polynomial equations using Gröbner bases. In *AAECC'5*, number 356 in Lecture Notes in Comput. Sci., pages 247–257. Springer, 1989.

[20] P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *ISSAC'03*, pages 135–142. ACM, 2003.

[21] P. Giorgi and R. Lebreton. Online order basis algorithm and its impact on the block Wiedemann algorithm. In *ISSAC'14*, pages 202–209. ACM, 2014.

[22] The LinBox Group. Linbox: Linear algebra over black-box matrices, version 1.5.0. `http://linalg.org/`, 2017.

[23] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[24] C.-P. Jeannerod, V. Neiger, É. Schost, and G. Villard. Fast computation of minimal interpolation bases in Popov form for arbitrary shifts. In *ISSAC'16*, pages 295–302. ACM, 2016.

[25] T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

[26] E. Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Mathematics of Computation*, 64(210):777–806, 1995.

[27] E. Kaltofen and G. Villard. On the complexity of computing determinants. In *ISSAC'01*, pages 13–27. ACM, 2001.

[28] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Comput. Complexity*, 13(3-4):91–130, 2004.

[29] W. Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theoret. Comput. Sci.*, 36(2-3):309–317, 1985.

[30] B. A. LaMacchia and A. M. Odlyzko. Solving large sparse linear systems over finite fields. In *Adv. in Cryptography, Crypto '90*, number 537 in Lectures Notes in Computer Science, pages 109–133. Springer, 1990.

[31] F. S. Macaulay. *The Algebraic Theory of Modular Systems*. Cambridge University Press, 1916.

[32] M. G. Marinari, H. M. Möller, and T. Mora. On multiplicities in polynomial system solving. *Trans. Amer. Math. Soc.*, 348(8):3283–3321, 1996.

[33] G. Moreno-Socías. *Autour de la fonction de Hilbert-Samuel (escaliers d'idéaux polynomiaux)*. PhD thesis, École polytechnique, 1991.

[34] A. Morgan. *Solving Polynominal Systems Using Continuation for Engineering and Scientific Problems*. Prentice-Hall, 1988.

[35] B. Mourrain. Isolated points, duality and residues. *Journal of Pure and Applied Algebra*, 117/118:469–493, 1997.

[36] V. Neiger. *Bases of relations in one or several variables: fast algorithms and applications*. PhD thesis, École Normale Supérieure de Lyon, November 2016.

[37] V. Neiger, H. Rahkooy, and É. Schost. Algorithms for zero-dimensional ideals using linear recurrent sequences. In *CASC'17*, pages 313–328. Springer, 2017.

[38] V. M. Popov. Invariant description of linear, time-invariant controllable systems. *SIAM Journal on Control*, 10(2):252–264, 1972.

[39] A. Poteaux and É. Schost. On the complexity of computing with zero-dimensional triangular sets. *J. Symbolic Comput.*, 50:110–138, 2013.

[40] F. Rouillier. Solving zero-dimensional systems through the Rational Univariate Representation. *Appl. Algebra Engrg. Comm. Comput.*, 9(5):433–461, 1999.

[41] S. Sakata. Extension of the Berlekamp-Massey algorithm to N dimensions. *Information and Computation*, 84(2):207–239, 1990.

[42] V. Shoup. NTL: A library for doing number theory. `http://www.shoup.net`.

[43] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symbolic Comput.*, 17(5):371–391, 1994.

[44] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *ISSAC'99*, pages 53–58. ACM, 1999.

[45] A Steel. Direct solution of the (11,9,8)-MinRank problem by the block Wiedemann algorithm in Magma with a Tesla GPU. In *PASCO'15*, pages 2–6. ACM, 2015.

[46] A. Storjohann. High-order lifting and integrality certification. *J. Symbolic Comput.*, 36:613–648, 2003.

[47] W. J. Turner. *Black box linear algebra with the LINBOX library*. PhD thesis, North Carolina State University, 2002.

[48] M. Van Barel and A. Bultheel. A general module theoretic framework for vector M-Padé and matrix rational interpolation. *Numer. Algorithms*, 3:451–462, 1992.

[49] G. Villard. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In *ISSAC'97*, pages 32–39. ACM, 1997.

[50] G. Villard. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials. Technical report, LMC-IMAG, Report 975 IM, 1997.

[51] W. A. Wolovich. *Linear Multivariable Systems*, volume 11 of *Applied Mathematical Sciences*. Springer-Verlag New-York, 1974.

[52] W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *J. Symbolic Comput.*, 47(7):793–819, 2012.