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

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

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

Seung Gyu Hyun

Cheriton School of Computer Science, University of Waterloo

Vincent Neiger

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

Hamid Rahkooy

Cheriton School of Computer Science, University of Waterloo

Éric Schost

Cheriton School of Computer Science, University of Waterloo

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 Eigen, LinBox and NTL.

Keywords: Polynomial systems; Block-Krylov algorithms; Sparse FGLM.

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 lexicographic 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 (Faugère, 2002; Faugère et al., 2013; Bardet et al., 2015). On the other hand, other orderings, such as the lexicographic 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 ordering 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* (Gianni and Mora, 1989). 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 exploitable output, such as a lexicographic basis. As pointed out in (Faugère and Mou, 2017), 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., 1993) showed how to do it in time $O(nD^3)$; more recently, algorithms have been given with cost bound $\tilde{O}(nD^\omega)$ (Faugère et al., 2013; Faugère et al., 2014; Neiger, 2016), at least for some favorable families of inputs. Here, the notation $\tilde{O}$ 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 (Faugère et al., 1993) 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 $\tilde{O}(nD^\omega)$ was only recently achieved (Faugère et al., 2013; Faugère et al., 2014; Neiger, 2016).

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 (Becker et al., 1994). As an alternative, one may then use the Rational Univariate Representation algorithm of Rouillier (1999) (see also (Alonso et al., 1996; Becker and Wörmann, 1996) 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 Alonso et al. (1996); Becker and Wörmann (1996); Rouillier (1999) 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 $\mathrm{trace} : \mathscr{Q} \to \mathbb{K}$ is the trace form and $X = t_1 X_1 + \cdots + t_n X_n$ is a generic $\mathbb{K}$-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. Bostan et al. (2003) 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 track of the multiplicities of the solutions (the algorithm in (Bostan et al., 2003) 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 (Marinari et al., 1996), (Mourrain, 1997) or (Neiger et al., 2017).

The most costly part of the algorithm of (Bostan et al., 2003) 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. Faugère and Mou (2017) 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 (1991). On this basis, they designed several forms of *sparse FGLM* algorithms. For instance, if I is in shape position, the algorithms in (Faugère and Mou, 2017) 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, these algorithms fall back either on the algorithm of Sakata (1990) or on plain FGLM.

The ideas at play in these algorithms are essentially based on Krylov subspace methods, using projections as well as Berlekamp-Massey techniques, along the lines of the algorithm of Wiedemann (1986) to solve sparse linear systems. These techniques have also been widely used in integer factorization or discrete logarithm calculations, going back to (LaMacchia and Odlyzko, 1990). It has become customary to design block versions of such algorithms to parallelize their bottleneck, as pioneered by Coppersmith (1994) in the context of integer factorization: it is then natural to adapt this strategy to our situation. This was already discussed by Steel (2015), 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 (Bostan et al., 2003) that extends the approach introduced in (Steel, 2015) to compute all polynomials in Eq. (3) for essentially the same cost as the computation of Q. More precisely, the bottleneck of the algorithm of (Steel, 2015) 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 (Faugère and Mou, 2017), a notable difference is that our algorithm deals with any zero-dimensional ideal I (for instance, we do not require shape position), but the base field must have sufficiently large 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 computations 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 is often denser than those of the variables X_i). The algorithm first computes a 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 general algorithm.

For experimental purposes, our algorithms have been implemented in C++ using the libraries Eigen (Guennebaud et al., 2018), LinBox (The LinBox Group, 2018) and NTL (Shoup, 2018).

The paper is organized as follows. The next section mainly reviews known results on scalar and matrix recurrent sequences, and introduces a simple 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 refined versions of results from (Bostan et al., 2003) that will be used throughout the paper. The main algorithm is given in Section 4, and the refinement mentioned above is in Section 5. Finally, in appendix, we prove a few technical statements on linearly recurrent matrix sequences.

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 the input ideal I is generated by polynomials $F_1, \ldots, F_t$. Given a zero-dimensional parametrization $((Q, V_1, \ldots, V_n), X)$ found 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 (von zur Gathen and Gerhard, 2013, 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, Dave Saunders, and Gilles Villard for several discussions, and a reviewer of the first version of this paper for their useful remarks. This research was partially supported by NSERC (Schost's NSERC Discovery Grant) and by the CNRS-INS2I Institute through its program for young researchers (Neiger's project ARCADIE).

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 Theorem 2.6 and Theorem 2.7, 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 (Villard, 1997a,b; Kaltofen and Villard, 2001; Turner, 2002; Kaltofen and Villard, 2004)), but the cost analysis we give uses results not available when those references were 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 nonzero 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.

In terms of generating series, it is easier to work here with generating series in the variable $1/T$. Let thus $Z = \sum_{s\geq 0} \ell_s/T^{s+1}$ 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 $\deg(P)$. In particular, given a scalar sequence 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 particular, $\Omega(\mathcal{L}, P)$ is a polynomial of degree less than $\deg(P)$.

The numerators thus defined are related to the Lagrange interpolation polynomials; this will explain why they play an important role in our main algorithm. Assuming that $\mathcal{L}$ is known to have order at most d and that we are given $\ell_0, \ldots, \ell_{2d-1}$, we can recover its minimal polynomial P by means of the Berlekamp-Massey algorithm, or of Euclid's algorithm (Brent et al., 1980). Given P and $\ell_0, \ldots, \ell_{d-1}$, we can deduce $\Omega(\mathcal{L}, P)$ by a simple multiplication.

As an example, consider the Fibonacci sequence $\mathcal{L} = (1, 1, 2, 3, 5, 8, \ldots)$, which is linearly recurrent with minimal polynomial $P = T^2 - T - 1$. Defining $Z = \sum_{s\geq 0} \ell_s/T^{s+1}$, we obtain

$$\Omega(\mathcal{L}, P) = (T^2 - T - 1)\left(\frac{1}{T} + \frac{1}{T^2} + \frac{2}{T^3} + \frac{3}{T^4} + \cdots\right) = T.$$

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, obtained by means of recent algorithms for approximant bases. A similar discussion, relying on the approximant basis algorithm in (Beckermann and Labahn, 1994), can be found in (Turner, 2002, Chapter 4); as a result, in the body of the text, we indicate only the key definitions and results (and in particular, an improved complexity analysis). Proofs of some statements are in appendix.

We first define linear recurrences for matrix sequences over a field $\mathbb{K}$ as in (Kaltofen and Villard, 2001, Section 3) or (Turner, 2002, Definition 4.2), hereby extending the above notions for scalar sequences. We only discuss the square case, since this is what we need below; the whole discussion can be generalized to rectangular matrices.

Definition 2.2. *Let $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ be a sequence of matrices in $\mathbb{K}^{m\times m}$. Then,*

- *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 $\mathcal{F}$ *if* $\boldsymbol{p}_0\boldsymbol{F}_s + \cdots + \boldsymbol{p}_d\boldsymbol{F}_{s+d} = \boldsymbol{0}$ *holds for all* $s \geq 0$*;*
- *the sequence* $\mathcal{F}$ *is* linearly recurrent *if the set of left vector relations for* $\mathcal{F}$ *is a* $\mathbb{K}[T]$*-submodule of* $\mathbb{K}[T]^{1\times m}$ *of rank* m*.*

Note that the set of left vector relations is always a free $\mathbb{K}[T]$-module, but its rank may be less than m. The following equivalent characterization of linearly recurrent sequences can be found for example in (Villard, 1997a; Kaltofen and Villard, 2001; Turner, 2002).

Lemma 2.3. *Let* $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ *be a sequence of matrices in* $\mathbb{K}^{m\times m}$*. Then* $\mathcal{F}$ *is linearly recurrent if and only if it admits a nonzero scalar relation, that is, if there exists a non-zero polynomial* $P = p_0 + \cdots + p_d T^d \in \mathbb{K}[T]$ *such that the identity* $p_0\boldsymbol{F}_s + \cdots + p_d\boldsymbol{F}_{s+d} = \boldsymbol{0}$ *holds for all* $s \geq 0$*.*

Definition 2.4. *Let* $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ *be a linearly recurrent sequence of matrices in* $\mathbb{K}^{m\times m}$*. A matrix* $\boldsymbol{P}$ *in* $\mathbb{K}[T]^{m\times m}$ *is*

- *a* left matrix relation *if its rows are all left vector relations for* $\mathcal{F}$*;*
- *a* left matrix generator *if its rows form a basis of the module of left vector relations for* $\mathcal{F}$*.*

In the latter case, $\boldsymbol{P}$ *is a* minimal generator *if it is in row reduced form.*

We refer to (Wolovich, 1974; Kailath, 1980) for a definition of reduced forms. The following proposition, which can be proved by a direct examination of the terms in the product $\boldsymbol{PZ}$, shows that matrix relations are denominators in a fraction description of the generating series of the sequence. As suggested in the previous subsection, working with generating series in $1/T$ turns out to be the most convenient choice here.

Proposition 2.5. *Let* $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ *be a sequence of matrices in* $\mathbb{K}^{m\times m}$ *and let* $\boldsymbol{P}$ *be a nonsingular matrix in* $\mathbb{K}[T]^{m\times m}$*. Then,* $\boldsymbol{P}$ *is a left matrix relation for* $\mathcal{F}$ *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, the* i*th row of* $\boldsymbol{Q}$ *has degree less than the* i*th row of* $\boldsymbol{P}$*, for* $1 \leq i \leq m$*.*

Given a nonsingular matrix relation $\boldsymbol{P}$ for $\mathcal{F}$, we will thus write $\boldsymbol{\Omega}(\mathcal{F}, \boldsymbol{P}) = \boldsymbol{PZ} \in \mathbb{K}[T]^{m\times m}$, generalizing Definition 2.1. By the previous proposition, 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 $\boldsymbol{\Omega}(\mathcal{F}, \boldsymbol{P})$ as

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

where "div T^d" means we keep the quotient of each entry by T^d. Given $\boldsymbol{P}$ and $\boldsymbol{F}_0, \ldots, \boldsymbol{F}_{d-1}$, the cost of computing $\boldsymbol{\Omega}(\mathcal{F}, \boldsymbol{P})$ is then $O(m^\omega \mathsf{M}(d))$ operations in $\mathbb{K}$.

Given a linearly recurrent sequence, our goal will be to find a minimal left matrix generator of it (from which we will easily deduce the corresponding numerator). In order to do this, we will assume that we know bounds on the degree of this left generator, but also on that of a *right* generator. Indeed, in all the previous discussion, we may also consider vector relations operating on the right. In particular, Lemma 2.3 shows that if a sequence is linearly recurrent, these right relations form a submodule of $\mathbb{K}[T]^{m\times 1}$ of rank m, so that a linearly recurrent sequence also admits right generators.

As it turns out, all minimal left generators have the same degree (by property of reduced forms); the same remark holds for minimal right generators. Knowing bounds (d_ℓ, d_r) on these

degrees allows us to control the number of terms of the sequence we will access during the algorithm (the bounds (d_ℓ, d_r) correspond to (γ_1, γ_2) in (Turner, 2002, Definitions 4.6 and 4.7) and (δ_l, δ_r) in (Villard, 1997b, Section 4.2)). In concrete terms, the fast computation of minimal matrix generators is usually handled via minimal approximant basis algorithms (see for example Villard, 1997a; Turner, 2002; Giorgi and Lebreton, 2014). The runtime below is obtained by using the divide and conquer approximant basis algorithm in (Giorgi et al., 2003).

Theorem 2.6. *Let $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ be a linearly recurrent sequence of matrices in $\mathbb{K}^{m\times m}$ and let $d = d_\ell + d_r + 1$, where $(d_\ell, d_r) \in \mathbb{N}^2$ are such that the minimal left (resp. right) matrix generators of $\mathcal{F}$ have degree at most d_ℓ (resp. at most d_r). Then, given $\boldsymbol{F}_0, \ldots, \boldsymbol{F}_{d-1}$, one can compute a minimal left matrix generator of $\mathcal{F}$ 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 (Coppersmith, 1994). 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 $m \leq D$. We can then define the Krylov sequence $\mathcal{F}_{\boldsymbol{U},\boldsymbol{V}} = (\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}})_{s\geq 0} \subset \mathbb{K}^{m\times m}$ by

$$\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}} = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s\boldsymbol{V}, \quad s \geq 0.$$

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 $\mathcal{F}_{\boldsymbol{U},\boldsymbol{V}}$, 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 in (Villard, 1997b) and (Kaltofen and Villard, 2004) (see also (Kaltofen, 1995) for an analysis of the block Wiedemann algorithm).

We let $s_1, \ldots, s_r$ be the nontrivial invariant factors of $T\boldsymbol{I}_D - \boldsymbol{M}$, ordered such that s_i divides s_{i-1} for $2 \leq i \leq r$, and let $d_i = \deg(s_i)$ for all i; for $i > r$, we let $s_i = 1$, with $d_i = 0$. We define $\nu = d_1 + \cdots + d_m \leq D$ and $\delta = \lceil \nu/m \rceil \leq \lceil D/m \rceil$.

Theorem 2.7. *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 $\mathcal{F}_{\boldsymbol{U},\boldsymbol{V}}$ and denote by $\sigma_1, \ldots, \sigma_k$ 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 matrix sequence $\mathcal{L}_{\boldsymbol{U}} = (\boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s)_{s\geq 0}$;*
- *$\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ has degree δ;*
- *$s_i = \sigma_i$ for $1 \leq i \leq m$.*

In particular, combining Theorem 2.6 and Theorem 2.7, we deduce that for generic $\boldsymbol{U}, \boldsymbol{V}$, given the first $2\lceil D/m \rceil + 1$ terms of the sequence $\mathcal{F}_{\boldsymbol{U},\boldsymbol{V}}$, we can recover a minimal matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ of it using $O(m^{\omega-1}\mathsf{M}(D)\log(D)) \subset \mathcal{O}(m^{\omega-1}D)$ 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 $P = s_1$ 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 (Storjohann, 2003, Algorithm 5) on input $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$ and $\boldsymbol{y}$; by (Storjohann, 2003, Corollary 16), this costs

$$O(m^{\omega-1}\mathsf{M}(D)\log(D)\log(m)) \subset \mathcal{O}(m^{\omega-1}D) \tag{5}$$

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 $\boldsymbol{P}_{U,V}$. An alternative solution would be to compute the Smith form of $\boldsymbol{P}_{U,V}$ using an algorithm such as that in (Storjohann, 2003, Section 17), yet the cost would be 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 $\boldsymbol{L}_s = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s$ 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 $\boldsymbol{U}^{\mathsf{T}}$. From these values, we then deduce the matrices $\boldsymbol{F}_{s,U,V} = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s\boldsymbol{V} = \boldsymbol{L}_s\boldsymbol{V}$ 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 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 $\boldsymbol{L}_s = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s$ 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

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

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

Lemma 2.8. *For generic $\boldsymbol{U}, \boldsymbol{V}$, the row vector $\boldsymbol{a}_i$ has entries which are polynomials 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 $\boldsymbol{P}_{U,V}$; thus, the product $P\,(\boldsymbol{P}_{U,V})^{-1}$ has polynomial entries. Since $\boldsymbol{a}_i$ the ith row of this matrix, $\boldsymbol{a}_i$ has polynomial entries. Now, since $\boldsymbol{P}_{U,V}$ is reduced, the predictable degree property (Kailath, 1980, 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 $\boldsymbol{a}_i\boldsymbol{P}_{U,V}$. 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 $O(m^{\omega-1}\mathsf{M}(D)\log(D)\log(m)) \subset O^{\sim}(m^{\omega-1}D)$ 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 2.9. *For a generic choice of $\boldsymbol{U}$ and $\boldsymbol{V}$, and for any $\boldsymbol{w}$ in $\mathbb{K}^{D\times 1}$, $\boldsymbol{P}_{U,V}$ is a nonsingular matrix relation for the sequence $\boldsymbol{\mathcal{E}} = (\boldsymbol{e}_s)_{s\geq 0}$, with $\boldsymbol{e}_s = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s\,\boldsymbol{w}$ for all s, and we have*

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

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

Proof. The first item in Theorem 2.7 shows that for a generic choice of $\boldsymbol{U}$ and $\boldsymbol{V}$, $\boldsymbol{P}_{U,V}$ cancels the sequence $(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s)_{s\geq 0}$, 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:

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

□

The algorithm to obtain 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 $\boldsymbol{P}_{U,V}$, P, $\boldsymbol{a}_i$ and the matrices $\boldsymbol{L}_s = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s \in \mathbb{K}^{m\times D}$ for $s = 0, \ldots, \lceil D/m\rceil - 1$.

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

Input:

- a minimal generator $\boldsymbol{P}_{U,V}$ of $(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s\boldsymbol{V})_{s\geq 0}$
- the minimal polynomial P of $\boldsymbol{M}$
- $\boldsymbol{w}$ in $\mathbb{K}^{D\times 1}$
- i in $\{1, \ldots, m\}$
- $\boldsymbol{a}_i = [0\ \cdots\ 0\ P\ 0\ \cdots\ 0](\boldsymbol{P}_{U,V})^{-1} \in \mathbb{K}[T]^{1\times m}$ where P appears at the ith entry
- $\boldsymbol{L}_s = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s \in \mathbb{K}^{m\times D}$, 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 $\boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{U,V}) \in \mathbb{K}[T]^{m\times 1}$ by Eq. (4)
3. **return** the entry of the 1×1 matrix $\boldsymbol{a}_i \cdot \boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{U,V})$

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 $O(D^2)$ field operations. Then, applying the cost estimate given after Eq. (4), we see that we have enough terms to compute $\boldsymbol{\Omega}(\boldsymbol{\mathcal{E}}, \boldsymbol{P}_{U,V}) \in \mathbb{K}[T]^{m\times 1}$ 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 $O(D^2 + m\mathsf{M}(D)) \subset \tilde{O}(D^2)$ 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 $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \ldots, \alpha_{i,n}) \in \overline{\mathbb{K}}^n$ 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 expand on results from the appendix of (Bostan et al., 2003), with the objective of computing a zero-dimensional parametrization of V. These results were

themselves inspired by those in (Rouillier, 1999), the latter being devoted to computations with the trace linear form $\mathrm{tr} : \mathscr{Q} \to \mathbb{K}$. 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 $\mathfrak{m}_{\boldsymbol{\alpha}_i}^N$ 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 nonsingular 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. (Bostan et al., 2003, 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 (Macaulay, 1916), and have been used in computational algebra since the 1990's (Marinari et al., 1996; Mourrain, 1997). 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 nonzero constants $c_{i,j,\mu}$ in $\overline{\mathbb{K}}$. For instance, when $\boldsymbol{\alpha}_i$ is nonsingular, 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 nonzero 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

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

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

Fix ℓ nonzero 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 $\mu = (\mu_1, \dots, \mu_n)$ in S_i such that $P_{i,\mu}(\boldsymbol{u}_i)$ is nonzero; by (Mourrain, 1997, 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 nonzero polynomial.

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 $\ell_1, \dots, \ell_\kappa$, 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 nonzero 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 a 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 (Bostan et al., 2003); the more explicit expression given here will be needed in the last section of this paper.

Lemma 3.2. *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 Section 3.1.*

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 $\mathbb{K}(X_1, \ldots, X_n)$:

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

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

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 $H_\mu \in \overline{\mathbb{K}}[X_1, \ldots, X_n, T]$ has degree in T less than $|\mu|$. 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 $A \in \overline{\mathbb{K}}[X_1, \ldots, X_n, T]$ 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

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

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. □

3.3. Computing a zero-dimensional parametrization

As a corollary, we recover the following result, that shows how to compute a zero-dimensional parametrization of $V_\mathfrak{S}$ (see Algorithm 2). 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.

Lemma 3.3 ((Bostan et al., 2003, Proposition 3)). *Let ℓ be a generic element of $\hom_{\overline{\mathbb{K}}}(\mathscr{Q}_\mathfrak{S}, \overline{\mathbb{K}})$ and $X = t_1X_1 + \cdots + t_nX_n$ be a generic linear combination of $X_1, \ldots, X_n$. Then the output $((Q, V_1, \ldots, V_n), X)$ of* Parametrization(ℓ, X) *is a zero-dimensional parametrization of $V_\mathfrak{S}$.*

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** $((Q, C_{X_1}/C_1 \bmod Q, \ldots, C_{X_n}/C_1 \bmod Q), X)$

The proof in Bostan et al. (2003) is written for $\mathfrak{S} = \{1, \ldots, \kappa\}$, but works equally as well for the more general case; it uses a weaker form of the previous lemma, that is sufficient in this case. 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

$$C_1 = \Omega((\ell(X_1^s))_{s\geq 0}, P) = 17(T-3) + 33(T-1),$$
$$C_{X_2} = \Omega((\ell(X_2X_1^s))_{s\geq 0}, P) = 17 \cdot 4(T-3) + 33 \cdot 2(T-1).$$

Now, let

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

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

4. The main algorithm

In this section, we extend the algorithm of (Bostan et al., 2003) 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. Our input is 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 above, D denotes 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, in the second subsection we present a simple example, and in the third we show experimental results of an implementation based on the C++ libraries Eigen, LinBox and NTL.

4.1. Description, correctness and cost analysis

We mentioned that the method of Steel (2015) 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 $(\boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s\boldsymbol{V})$, this is done by means of polynomial lattice reduction. 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 (Algorithm 3) computes the whole zero-dimensional parametrization of $V(I)$ for essentially the same cost as the computation of the minimal polynomial. Hereafter, $\boldsymbol{\varepsilon}_1$ denotes the size-D column vector whose only nonzero entry is a 1 at the first index: $\boldsymbol{\varepsilon}_1 = [1\ 0\ \cdots\ 0]^\mathsf{T}$.

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

Input:
- $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n$ multiplication matrices 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 $\boldsymbol{L}_s = \boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s$ 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 $\mathcal{F}_{U,V} = (\boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s\boldsymbol{V})_{s\geq 0}$. As discussed in Section 2.3, Theorem 2.7 shows that the matrix polynomial $\boldsymbol{P}_{U,V}$ is indeed a minimal left generator of the sequence $\mathcal{F}_{U,V}$, 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. In particular, the scalar numerators needed in this algorithm are computed using Algorithm 1 (ScalarNumerator); indeed, applying Lemma 2.9, we see that calling this algorithm 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

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

so they coincide with the polynomials in Algorithm 2. Thus, by Lemma 3.3, 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 gained this way (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 focus on a sparse model: we let $\rho \in [0, 1]$ denote the density of $\boldsymbol{M}$ and the $\boldsymbol{M}_i$'s, that is, all these matrices have at most ρD^2 nonzero entries. As a result, a matrix-vector product by $\boldsymbol{M}$ can be done in $O(\rho D^2)$ operations in $\mathbb{K}$. 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 $\boldsymbol{L}_s = \boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s$, for $s = 0, \ldots, 2d-1$, is the bottleneck of the algorithm but 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 $\boldsymbol{U}^\mathsf{T}$. 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 in parallel m vector-matrix products 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 $\boldsymbol{F}_{s,U,V} = \boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s\boldsymbol{V}$, for $s = 0, \ldots, 2d-1$ by the product

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

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 matrix generator $\boldsymbol{P}_{U,V}$ in time

$$O(m^\omega \mathsf{M}(D/m)\log(D/m)) \subseteq O(m^{\omega-1}\mathsf{M}(D)\log(D)),$$

and from Sections 2.3 and 2.4 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, for a total of $O(nD^2 + nm\mathsf{M}(D))$; the final computations modulo Q at Step 10 take time $O(n\mathsf{M}(D) + \mathsf{M}(D)\log(D))$. Thus, 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

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

Although one may work on parallelizing other steps than Step 2, we note that this step is simultaneously the most costly in theory and in practice, and the easiest to parallelize.

Remark 4.2. Our algorithm only computes the first invariant factor of $\boldsymbol{P}_{U,V}$, 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 $\mathscr{Q}_i$). □

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 $m = 1$. We compute the row-vectors $\boldsymbol{L}_s$, for $s = 0, \dots, 2D-1$, using the square-and-multiply technique from (Keller-Gehrig, 1985), for $O(D^\omega \log(D))$ operations in $\mathbb{K}$. For generic choices of $\boldsymbol{U}$ and $\boldsymbol{V}$, a minimal matrix generator $\boldsymbol{P}_{U,V}$ is equal to the minimum polynomial P of $\boldsymbol{M}$, and can be computed efficiently by the Berlekamp-Massey algorithm; besides, $\boldsymbol{a}_1 = P(\boldsymbol{P}_{U,V})^{-1} = 1$. 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 $d = D/m = 1$. The canonical matrix generator of $\boldsymbol{\mathcal{F}}_{U,V} = (\boldsymbol{U}^\mathsf{T}\boldsymbol{V}, \boldsymbol{U}^\mathsf{T}\boldsymbol{M}\boldsymbol{V}, \dots)$ is $\boldsymbol{P}_{U,V} = T\boldsymbol{I}_D - \boldsymbol{U}^\mathsf{T}\boldsymbol{M}\boldsymbol{U}^{-\mathsf{T}}$ 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 $\boldsymbol{\Omega}(\boldsymbol{U}^\mathsf{T}\boldsymbol{M}\boldsymbol{\varepsilon}_1, \boldsymbol{P}_{U,V})$ is then seen to be $\boldsymbol{U}^\mathsf{T}\boldsymbol{\varepsilon}_1$, that is, the first column of $\boldsymbol{U}^\mathsf{T}$; we recover C_1 from it through a dot product with $\boldsymbol{a}_1$. Similarly, the numerator $\boldsymbol{\Omega}(\boldsymbol{U}^\mathsf{T}\boldsymbol{M}\boldsymbol{M}_i\boldsymbol{\varepsilon}_1, \boldsymbol{P}_{U,mV})$ is $\boldsymbol{U}^\mathsf{T}\boldsymbol{M}_i\boldsymbol{\varepsilon}_1$, 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}.

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 is not obvious from 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 the multiplication matrices of X_1, X_2 and X in the basis $\mathscr{B} = (1, X_1, X_2, X_2^2)$ of $\mathscr{Q}$ are respectively

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

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 $\mathcal{F}_{U,V} = (\boldsymbol{U}^\mathsf{T} \boldsymbol{M}^s \boldsymbol{V})_{s \geq 0}$ and its minimum matrix generator $\boldsymbol{P}_{U,V}$. This is done by first computing

$$\boldsymbol{U}^\mathsf{T} = \begin{bmatrix} 84 & 29 & 80 & 7 \\ 38 & 58 & 43 & 82 \end{bmatrix}, \quad \boldsymbol{U}^\mathsf{T}\boldsymbol{M} = \begin{bmatrix} 54 & 28 & 67 & 81 \\ 34 & 52 & 90 & 29 \end{bmatrix}, \tag{10}$$

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

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

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

and then we obtain the minimal matrix generator

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

The largest invariant factor of $\boldsymbol{P}_{U,V}$ is $P = T^3 + 76T^2 + 100T + 7$, with squarefree part $Q = T^2 + 8T + 61$. Next, we compute the row vector $\boldsymbol{a}_1 = [T + 16, 13]$ by solving $\boldsymbol{a}_1 = [P \ 0](\boldsymbol{P}_{U,V})^{-1}$, and the matrix numerator

$$\boldsymbol{\Omega}((\boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s\boldsymbol{\varepsilon}_1)_{s \geq 0}, \boldsymbol{P}_{U,V}) = \begin{bmatrix} 84T + 55 \\ 38T + 11 \end{bmatrix}$$

which is made from the entries of nonnegative degree in the product

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

where the columns are the first columns of the matrices in Eq. (10) (as per Eq. (4), we only need $d = 2$ terms in the right-hand side). From this, we find the scalar numerator

$$C_1 = \Omega((\boldsymbol{u}_1 \boldsymbol{M}^s \boldsymbol{\varepsilon}_1)_{s \geq 0}, P) = 84T^2 + 75T + 13$$

by means of the dot product $[\ C_1\ ] = \boldsymbol{a}_1 \cdot \boldsymbol{\Omega}((\boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s\boldsymbol{\varepsilon}_1)_{s\geq 0}, \boldsymbol{P}_{U,V})$.

We then find $C_{X_1} = 88T^2 + 47T + 16$, by proceeding similarly: we find the matrix numerator

$$\boldsymbol{\Omega}((\boldsymbol{u}_1\boldsymbol{M}^s\boldsymbol{M}_1\boldsymbol{\varepsilon}_1)_{s\geq 0}, \boldsymbol{P}_{U,V}) = \begin{bmatrix} 88T + 19 \\ 57T + 40 \end{bmatrix}$$

and we take the dot product with $\boldsymbol{a}_1$. Thus, we obtain the polynomial $V_1 = C_{X_1}/C_1 \bmod Q = 15T + 14$. 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 for univariate polynomials, LinBox for dense polynomial matrices and matrix generator computations, and Eigen for sparse matrix-vector products. It is dedicated to small prime fields; thus, as is done in (The FFLAS-FFPACK Group, 2019) 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 128GB RAM and 8 cores (or 16 via 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 (Bosma et al., 1997). The timings reported here do not include this precomputation; instead, we refer the reader to (Faugère and Mou, 2017) for extensive experiments comparing the runtime of two similar algorithmic stages (degree basis computation and conversion to a lex ordering). 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, improvements are possible for polynomial matrix computations (minimal matrix generator, 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://github.com/vneiger/block-sparse-fglm`. Some systems are well-known (Katsura or Eco from (Morgan, 1988)), while we also consider families of randomly generated inputs (some are inspired from (Faugère and Mou, 2017)). 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 Krylov sequence, as described in Section 4.1. In the columns $m = 1$, $m = 3$, $m = 6$, the numbers in brackets

indicate the fraction of the total time spent in computing the Krylov sequence $(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s)_{0\le s<2d}$, with $d = \lceil D/m \rceil$. This is always by far the dominant factor. In other words, almost all the time is spent doing sparse matrix-vector products for matrices with machine float entries.

Increasing m has two effects. On the plus side, it decreases the length of the Krylov sequence. 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**
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 (Faugère and Mou, 2017) (the algorithm in that reference does not use a blocking strategy). Recall that the output in (Faugère and Mou, 2017) is somewhat stronger than here (the authors compute a Gröbner basis of the input ideal, so multiplicities are preserved). 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 focus on avoiding or reducing the use of a generic linear form $X = t_1X_1 + \cdots + t_nX_n$. 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 less sparse 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 $\mathscr{Q}_i, \ldots$, are as in the previous two sections. In particular, we write $V = V(I) = \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\kappa\}$, with $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \ldots, \alpha_{i,n}) \in \overline{\mathbb{K}}^n$ for all i.

5.1. Overview

The algorithm decomposes 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; for the remaining points, we will use 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 $1, X, X^2, \ldots$), linear algebra over univariate polynomials, and some operations on univariate polynomials related to the so-called *power projection* (Shoup, 1994, 1999).

The main algorithm is as follows. For the moment, we only describe its main structure; the subroutines are detailed 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 $(F, G_1, \ldots, G_n, X_1) = \mathsf{BlockParametrizationX}_1(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, Y)$
2. let $(\boldsymbol{\Delta}_s), (\boldsymbol{\Delta}'_s), (\boldsymbol{\Delta}''_s)$ be correction matrices associated to the previous calculation
3. let $(R, W_1, \ldots, W_n, X) = \mathsf{BlockParametrizationResidual}(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_n, \boldsymbol{U}, \boldsymbol{V}, (\boldsymbol{\Delta}_s), (\boldsymbol{\Delta}'_s), (\boldsymbol{\Delta}''_s), X)$
4. let $((Q, V_1, \ldots, V_n), X) = \mathsf{Union}(((F, G_1, \ldots, G_n), X_1), ((R, W_1, \ldots, W_n), X))$
5. **return** $((Q, V_1, \ldots, V_n), X)$

The call to $\mathsf{BlockParametrizationX}_1$ 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 $(\boldsymbol{U}^\mathsf{T}\boldsymbol{M}_1^s\boldsymbol{V})_{s\geq 0}$. The next step finds three sequences of correction matrices, using objects computed in the call to $\mathsf{BlockParametrizationX}_1$.

We then apply a modified version of Algorithm BlockParametrization, which we call Block ParametrizationResidual. It computes a zero-dimensional parametrization of $V'' = V \setminus V'$ using sequences such as $(\boldsymbol{U}^\mathsf{T}\boldsymbol{M}^s\boldsymbol{V})_{s\geq 0} - \boldsymbol{\Delta}_s$, 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 cases where V' is a large subset of V, then $V'' = V \setminus V'$ contains 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 $O(nD^{(\omega+1)/2})$ (Poteaux and Schost, 2013, Lemmas 2 & 3); we will not discuss them further.

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 $\mu_j = 1$.

Lemma 5.1. *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

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

Using Lemma 3.2 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 $E_i/(T - r_j)^{\nu_i}$, for some integer ν_i, and for some polynomial $E_i \in \overline{\mathbb{K}}[T]$ of degree less than ν_j. Next, take j in $\mathfrak{e}$. Since $\mu_j = 1$, $\nu_i = 1$ for all i in T_j, so that each ℓ_i takes the form $\ell_i : f \mapsto (\Lambda_i(f))(\boldsymbol{\alpha}_i)$, for a differential operator Λ_i 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

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

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 polynomial N divides M, so that $x_j \le \mu_j$ holds for all j in $\mathfrak{f}$. As a result, $\Omega((\ell(X_1^s))_{s\ge 0}, M)$ is well-defined and is given by

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

This implies that

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

holds for all k in $\mathfrak{e}$. □

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

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

We denote by $\mathfrak{A} \subset \{1,\dots,\kappa\}$ the set of corresponding indices i, and we let $\mathfrak{B} = \{1,\dots,\kappa\} \setminus \mathfrak{A}$, 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 5.2. *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 $a = \lambda(1)$, $b = \lambda(Y)$, $c = \lambda(Y^2)$ in $\overline{\mathbb{K}}$. 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

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

which gives

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

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 nonzero. The quantity $ac-b^2$ is a polynomial in the parameters $(\boldsymbol{u}_i)_{i\in T_j}$, and $(y_i)_{i\in\{2,\dots,n\}}$, 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 nonzero.

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 $\{2,\dots,n\}$ such that $P_{\sigma,\rho}(u_{\sigma,2},\dots,u_{\sigma,D_\sigma})$ 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 $-(P_{\sigma,\rho}(u_{\sigma,2},\dots,u_{\sigma,D_\sigma})y_\rho)^2$ which is nonzero, hence $ac - b^2$ itself is nonzero.

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 nonzero, 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 $F = \prod_{j\in\{1,\dots,c\},\mu_j=1}(T - r_j)$, 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.

Lemma 5.3. *For generic choices of $\boldsymbol{U}$, $\boldsymbol{V}$ and Y, the output $((F, G_1,\dots,G_n), X_1)$ of the algorithm* 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 $F = \prod_{j\in\{1,\dots,c\},\mu_j=1}(T - r_j)$.

The calculation of A_0, A_1, A_2 and $A_{X_2},\dots,A_{X_n}$ is justified as in BlockParametrization, by means of Lemma 2.9. For $i = 1,\dots,D$, 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

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

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 $\boldsymbol{L}_s = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}_1^s$ 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** $((F, T, A_{X_2}/A_0 \bmod F, \ldots, A_{X_n}/A_0 \bmod F), X_1)$

Then, the above lemma proves that we have $A_i = \Omega((\ell(Y^iX_1^s))_{s\ge0}, M)$ for $i = 0, 1, 2$ as well as $A_{X_i} = \Omega((\ell(X_iX_1^s))_{s\ge0}, M)$ 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 5.1, 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 $f \mapsto \lambda_j(Y^if)$.

As a result, the value of $A_0A_2 - A_1^2$ at r_j is (up to the nonzero factor $M'(r_j)^2$) equal to the quantity $ac - b^2$ defined in Lemma 5.2, 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 $V_{\mathfrak{N}}$. This is done using again Lemma 5.1. 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 $f \mapsto \lambda_j(X_if)$. 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 nonzero. For $i = 1$, since we use X_1 as a separating variable for $V_{\mathfrak{N}}$, 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 $(\mathbf{\Delta}_s), (\mathbf{\Delta}'_s), (\mathbf{\Delta}''_s)$ 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 (Shoup, 1994, 1999): 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 $(\ell(1), \ell(T), \dots, \ell(T^{r-1}))$.

Shoup (1994, Theorem 4) 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 (Bostan et al., 2006, 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 $C = \Omega((\ell(X_1^s))_{s\ge 0}, M)$, as well as the zero-dimensional parametrization $((F, G_1, \dots, G_n), X_1)$ 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 $s = 0, \dots, \tau - 1$.

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

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

for some polynomials A, B in $\mathbb{K}[T]$, with $\deg(A) < \deg(F)$ and $\deg(B) < \deg(E)$. With $C = \Omega((\ell(X_1^s))_{s\ge 0}, M)$, 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 $s = 0, \dots, D_{\mathfrak{A}} - 1$, 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 $G_1, \dots, G_n$ are polynomials in the zero-dimensional parametrization of $V_{\mathfrak{A}}$. As a result, we can finally compute $\ell_{\mathfrak{A}}(X^s)$, for $s = 0, \dots, \tau - 1$ by applying our algorithm for univariate power projection to $G = t_1G_1 + \cdots t_nG_n$.

Algorithm 6 (Decompose) summarizes this discussion. Its cost bound 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 $O(nD_{\mathfrak{A}})$ is the cost of computing G as defined above.

Algorithm 6 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 $V_{\mathfrak{P}}$
- $X = t_1 X_1 + \cdots + t_n X_n$
- a bound τ

Output:

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

1. let $E = M/F$
2. let $A = C/E \bmod F$
3. compute the first $D_{\mathfrak{P}}$ terms $(v_0, \ldots, v_{D_{\mathfrak{P}}-1})$ of the Laurent series A/F
4. **return** PowerProjection$(F, t_1 G_1 + \cdots + t_n G_n, (v_0, \ldots, v_{D_{\mathfrak{P}}-1}), \tau)$

Algorithm Decompose will be used for obtaining correction matrices given as input of Algorithm BlockParametrizationResidual. We assume that we have stored various quantities computed in Algorithm BlockParametrizationX$_1$: the sequence of matrices $(\boldsymbol{F}_{s,\boldsymbol{U},\boldsymbol{V}})_{0\leq s<2d}$, the matrix generator $\boldsymbol{P}_{\boldsymbol{U},\boldsymbol{V}}$, the minimal polynomial M of X_1, and the 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 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 $(\boldsymbol{U}^{\mathsf{T}} \boldsymbol{M}_1^s \boldsymbol{V})_{s\geq 0}$ is the scalar sequence $(\ell_{i,j}(X_1^s))_{s\geq 0}$.

  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 $\ell_{i,j,\mathfrak{P}}(X^s)$, for $s = 0, \ldots, 2d_{\mathfrak{B}} - 1$. We can then construct the sequence $(\boldsymbol{\Delta}_s)_{0\leq s<2d_{\mathfrak{B}}}$ of matrices in $\mathbb{K}^{m\times m}$ by setting the (i, j)-th entry of $\boldsymbol{\Delta}_s$ to be $\ell_{i,j,\mathfrak{P}}(X^s)$.

- 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 $(\boldsymbol{U}^{\mathsf{T}} \boldsymbol{M}_1^s \boldsymbol{M}_k \boldsymbol{\varepsilon}_1)_{s\geq 0}$ is the scalar sequence $(\ell'_{i,k}(X_1^s))_{s\geq 0}$, where $\boldsymbol{\varepsilon}_1$ is the column vector $[1\, 0 \cdots\, 0]^{\mathsf{T}}$ we already used several times.

  Proceeding as before, we construct the sequence of $m \times n$ matrices $(\boldsymbol{\Delta}'_s)_{0\leq s<d_{\mathfrak{B}}}$ by setting the (i, k)-th entry of $\boldsymbol{\Delta}'_s$ to be $\ell'_{i,k,\mathfrak{P}}(X^s)$. 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 $(\boldsymbol{U}^{\mathsf{T}} \boldsymbol{M}_1^s \boldsymbol{\varepsilon}_1)_{s\geq 0}$ is the scalar sequence $(\ell_i(X_1^s))_{s\geq 0}$. 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 $\ell_{i,\mathfrak{U}}(X^s)$.

In terms of cost, computing the vectors $\boldsymbol{a}_i$, for $i = 1, \ldots, m$, uses $O(m^\omega \mathsf{M}(D)\log(D)\log(m))$ operations in $\mathbb{K}$ (see Eq. (5)). 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 $\mathfrak{U}$ and $\mathfrak{B}$ be as in the previous section. This part of the main algorithm computes a zero-dimensional parametrization of the residual set $V_{\mathfrak{B}} = V \setminus V_{\mathfrak{U}}$. 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 $\boldsymbol{D}_s = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s$ for $s = 0, \ldots, 2d_{\mathfrak{B}} - 1$, with $d_{\mathfrak{B}} = \lceil D_{\mathfrak{B}}/m\rceil$
3. compute $\boldsymbol{E}_{s,U,V} = \boldsymbol{D}_s\boldsymbol{V} - \boldsymbol{\Delta}_s$ for $s = 0, \ldots, 2d_{\mathfrak{B}} - 1$
4. compute a minimal matrix generator $\boldsymbol{S}_{U,V}$ of $(\boldsymbol{E}_{s,U,V})_{0\le s<2d_{\mathfrak{B}}}$
5. let S be the largest invariant factor of $\boldsymbol{S}_{U,V}$
6. let R be the squarefree part of S
7. let $\boldsymbol{a}_1 = [S\ 0\ \cdots\ 0](\boldsymbol{S}_{U,V})^{-1}$
8. let $C_1 =$ ScalarNumeratorCorrected$(\boldsymbol{S}_{U,V}, S, \boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{D}_s)_{0\le s<d_{\mathfrak{B}}}, (\boldsymbol{\Delta}''_s)_{0\le s<d_{\mathfrak{B}}})$
9. **for** $i = 1, \ldots, n$ **do**
   let $C_{X_i} =$ ScalarNumeratorCorrected$(\boldsymbol{S}_{U,V}, S, \boldsymbol{M}_i\boldsymbol{\varepsilon}_1, 1, \boldsymbol{a}_1, (\boldsymbol{D}_s)_{0\le s<d_{\mathfrak{B}}}, (\boldsymbol{\Delta}'_s)_{0\le s<d_{\mathfrak{B}}})$
10. **return** $((R, C_{X_1}/C_1 \bmod R, \ldots, C_{X_n}/C_1 \bmod R), X)$

Let us prove correctness. Since $\mathscr{Q} = \mathscr{Q}_{\mathfrak{U}} \times \mathscr{Q}_{\mathfrak{B}}$, 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 $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{M}^s\boldsymbol{V} = \boldsymbol{U}_{\mathfrak{A}}^{\mathsf{T}}\boldsymbol{M}_{\mathfrak{A}}^s\boldsymbol{V}_{\mathfrak{A}} + \boldsymbol{U}_{\mathfrak{B}}^{\mathsf{T}}\boldsymbol{M}_{\mathfrak{B}}^s\boldsymbol{V}_{\mathfrak{B}}$ for $s \geq 0$. The first summand is none other than the matrix $\boldsymbol{\Delta}_s$, so that $\boldsymbol{E}_{s,U,V}$ is equal to $\boldsymbol{U}_{\mathfrak{B}}^{\mathsf{T}}\boldsymbol{M}_{\mathfrak{B}}^s\boldsymbol{V}_{\mathfrak{B}}$. 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 $\boldsymbol{S}_{U,V}$, 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

$$\begin{aligned} C_1 &= \Omega((\ell_1(X^s) - \ell_{1,\mathfrak{A}}(X^s))_{s\geq 0}, S) = \Omega((\ell_{1,\mathfrak{B}}(X^s))_{s\geq 0}, S), \\ C_{X_i} &= \Omega((\ell_1(X_iX^s) - \ell_{1,\mathfrak{A}}(X_iX^s))_{s\geq 0}, S) = \Omega((\ell_{1,\mathfrak{B}}(X_iX^s))_{s\geq 0}, S) \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

As shown in Section 3.3, these are precisely the polynomials we need in order 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 $d_{\mathfrak{B}} = \lceil D_{\mathfrak{B}}/m \rceil$ (which is hopefully much smaller).

5.5. Experimental results

The algorithms in this section were implemented using the same framework as 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 it actually will not 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 seldom seen in our experiments, since the systems with multiplicities generated randomly had few multiple points, and thus were favorable to us. An unfavorable case is system "sot1", where $V_{\mathfrak{A}}$ only accounts for 1012 points out of 7682 points in the variety.

Appendix

In this appendix, we prove Theorem 2.6 from Section 2.2: *Let $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ be a linearly recurrent sequence of matrices in $\mathbb{K}^{m\times m}$ and let $d = d_\ell + d_r + 1$, where $(d_\ell, d_r) \in \mathbb{N}^2$ are such that the*

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

minimal left (resp. right) matrix generators of $\mathcal{F}$ have degree at most d_ℓ (resp. at most d_r). Then, given $\boldsymbol{F}_0, \ldots, \boldsymbol{F}_{d-1}$, one can compute a minimal left matrix generator of $\mathcal{F}$ in $O(m^\omega \mathsf{M}(d)\log(d))$ operations in $\mathbb{K}$. The first lemma we need is similar to (Turner, 2002, Theorem 4.5).

Lemma 5.4. *Let $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ be a linearly recurrent sequence of matrices in $\mathbb{K}^{m\times m}$ and let $d_r \in \mathbb{N}$ be such that minimal right matrix generators of $\mathcal{F}$ have 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 $\mathcal{F}$ 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. In this proof, for a $u \times v$ polynomial matrix $\boldsymbol{Q}$, we denote by $\mathrm{cdeg}(\boldsymbol{Q})$ the size-v vector of the degrees of the columns of $\boldsymbol{Q}$. Consider a minimal right generator $\boldsymbol{P} \in \mathbb{K}[T]^{m\times m}$ in Popov form (as defined e.g. in (Kailath, 1980)). Then, we have $\boldsymbol{P} = \boldsymbol{L}\,\mathrm{Diag}(T^{t_1}, \ldots, T^{t_m}) - \boldsymbol{Q}$, where $\mathrm{cdeg}(\boldsymbol{Q}) < \mathrm{cdeg}(\boldsymbol{P}) = (t_1, \ldots, t_m)$ termwise and $\boldsymbol{L} \in \mathbb{K}^{m\times m}$ is unit upper triangular. Define the matrix $\boldsymbol{U} = \mathrm{Diag}(T^{d_r - t_1}, \ldots, T^{d_r - t_m})\boldsymbol{L}^{-1}$, 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 $\boldsymbol{PU} = T^{d_r}\boldsymbol{I}_m - \boldsymbol{QU}$ are right relations for $\mathcal{F}$, and we have $\deg(\boldsymbol{QU}) < d_r$. Thus, writing $\boldsymbol{QU} = \sum_{0\leq k<d_r} \boldsymbol{Q}_k T^k$, 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

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

and the proof is complete. □

The next result is similar to (Turner, 2002, Theorem 4.6) (see also Theorems 4.7 to 4.10 in that reference). We recall from (Van Barel and Bultheel, 1992; Beckermann and Labahn, 1994) 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 $\mathcal{F}$ 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 $\mathcal{F}$.

Theorem 5.5. *Let $\mathcal{F} = (\boldsymbol{F}_s)_{s\geq 0}$ be a linearly recurrent sequence of matrices in $\mathbb{K}^{m\times m}$ and let $(d_\ell, d_r) \in \mathbb{N}^2$ be such that the minimal left (resp. right) matrix generator of $\mathcal{F}$ have degree at most d_ℓ (resp. at most 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 row reduced, it has exactly m rows of degree $\leq d_\ell$, and they form a submatrix $\begin{bmatrix}\boldsymbol{P} & \boldsymbol{R}\end{bmatrix} \in \mathbb{K}[T]^{m\times(m+m)}$ of $\boldsymbol{B}$ such that $\boldsymbol{P}$ is a minimal matrix generator for $\mathcal{F}$.

Proof. We first observe that for any relation $\boldsymbol{p} \in \mathbb{K}[T]^{1\times m}$ for $\mathcal{F}$, 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, if $\boldsymbol{p}$ is a relation for $\mathcal{F}$ 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, we show that 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 $\mathcal{F}$. 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 $\boldsymbol{p}(\sum_{0\leq s<d} \boldsymbol{F}_s T^{d-s-1})$ are zero. Then, Lemma 5.4 shows that $\boldsymbol{p}$ is a relation for $\mathcal{F}$. The theorem follows. □

Using the fast approximant basis algorithm of (Giorgi et al., 2003), this implies Theorem 2.6.

References

Alonso, M.E., Becker, E., Roy, M.F., Wörmann, T., 1996. Zeroes, multiplicities and idempotents for zero-dimensional systems, in: MEGA'94, Birkhäuser. pp. 1–15.

Bardet, M., Faugère, J.C., Salvy, B., 2015. On the complexity of the F5 Gröbner basis algorithm. J. Symbolic Comput. 70, 49–70.

Becker, E., Mora, T., Marinari, M., Traverso, C., 1994. The shape of the Shape Lemma, in: ISSAC'94, ACM. pp. 129–133.

Becker, E., Wörmann, T., 1996. Radical computations of zero-dimensional ideals and real root counting. Mathematics and Computers in Simulation 42, 561–569.

Beckermann, B., Labahn, G., 1994. A uniform approach for the fast computation of matrix-type Padé approximants. SIAM J. Matrix Anal. Appl. 15, 804–823.

Bosma, W., Cannon, J., Playoust, C., 1997. The Magma algebra system. I. The user language. J. Symbolic Comput. 24, 235–265.

Bostan, A., Flajolet, P., Salvy, B., Schost, É., 2006. Fast computation of special resultants. J. Symbolic Comput. 41, 1–29.

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

Brent, R.P., Gustavson, F.G., Yun, D.Y.Y., 1980. Fast solution of Toeplitz systems of equations and computation of Padé approximants. Journal of Algorithms 1, 259–295.

Coppersmith, D., 1994. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. Math. Comp. 62, 333–350.
Faugère, J.C., 2002. A new efficient algorithm for computing Gröbner bases without reductions to zero (F5), in: ISSAC'02, ACM. pp. 75–83.
Faugère, J.C., Gaudry, P., Huot, L., Renault, G., 2013. Polynomial systems solving by fast linear algebra. https://hal.archives-ouvertes.fr/hal-00816724.
Faugère, J.C., Gaudry, P., Huot, L., Renault, G., 2014. Sub-cubic change of ordering for Gröbner basis: a probabilistic approach, in: ISSAC'14, ACM. pp. 170–177.
Faugère, J.C., Gianni, P., Lazard, D., Mora, T., 1993. Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symbolic Comput. 16, 329–344.
Faugère, J.C., Mou, C., 2017. Sparse FGLM algorithms. J. Symbolic Comput. 80, 538–569.
Faugère, J.C., Safey El Din, M., Spaenlehauer, P.J., 2013. On the complexity of the generalized MinRank problem. J. Symbolic Comput. , 30–58.
von zur Gathen, J., Gerhard, J., 2013. Modern Computer Algebra. Third ed., Cambridge University Press, Cambridge.
Gianni, P., Mora, T., 1989. Algebraic solution of systems of polynomial equations using Gröbner bases, in: AAECC'5, Springer. pp. 247–257.
Giorgi, P., Jeannerod, C.P., Villard, G., 2003. On the complexity of polynomial matrix computations, in: ISSAC'03, ACM. pp. 135–142.
Giorgi, P., Lebreton, R., 2014. Online order basis algorithm and its impact on the block Wiedemann algorithm, in: ISSAC'14, ACM. pp. 202–209.
Guennebaud, G., Jacob, B., et al., 2018. Eigen, version 3.3.7. http://eigen.tuxfamily.org.
Kailath, T., 1980. Linear Systems. Prentice-Hall.
Kaltofen, E., 1995. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. Mathematics of Computation 64, 777–806.
Kaltofen, E., Villard, G., 2001. On the complexity of computing determinants, in: ISSAC'01, ACM. pp. 13–27.
Kaltofen, E., Villard, G., 2004. On the complexity of computing determinants. Comput. Complexity 13, 91–130.
Keller-Gehrig, W., 1985. Fast algorithms for the characteristic polynomial. Theoret. Comput. Sci. 36, 309–317.
LaMacchia, B.A., Odlyzko, A.M., 1990. Solving large sparse linear systems over finite fields, in: Adv. in Cryptography, Crypto '90, Springer. pp. 109–133.
Macaulay, F.S., 1916. The Algebraic Theory of Modular Systems. Cambridge University Press.
Marinari, M.G., Möller, H.M., Mora, T., 1996. On multiplicities in polynomial system solving. Trans. Amer. Math. Soc. 348, 3283–3321.
Moreno-Socías, G., 1991. Autour de la fonction de Hilbert-Samuel (escaliers d'idéaux polynomiaux). Ph.D. thesis. École polytechnique.
Morgan, A., 1988. Solving Polynominal Systems Using Continuation for Engineering and Scientific Problems. Prentice-Hall.
Mourrain, B., 1997. Isolated points, duality and residues. Journal of Pure and Applied Algebra 117/118, 469–493.
Neiger, V., 2016. Bases of relations in one or several variables: fast algorithms and applications. Ph.D. thesis. École Normale Supérieure de Lyon.
Neiger, V., Rahkooy, H., Schost, É., 2017. Algorithms for zero-dimensional ideals using linear recurrent sequences, in: CASC'17, Springer. pp. 313–328.
Poteaux, A., Schost, E., 2013. On the complexity of computing with zero-dimensional triangular sets. J. Symbolic Comput. 50, 110–138.
Rouillier, F., 1999. Solving zero-dimensional systems through the Rational Univariate Representation. Appl. Algebra Engrg. Comm. Comput. 9, 433–461.
Sakata, S., 1990. Extension of the Berlekamp-Massey algorithm to N dimensions. Information and Computation 84, 207–239.
Shoup, V., 1994. Fast construction of irreducible polynomials over finite fields. J. Symbolic Comput. 17, 371–391.
Shoup, V., 1999. Efficient computation of minimal polynomials in algebraic extensions of finite fields, in: ISSAC'99, ACM. pp. 53–58.
Shoup, V., 2018. NTL: A library for doing number theory, version 11.3.2. http://www.shoup.net.
Steel, A., 2015. Direct solution of the (11,9,8)-MinRank problem by the block Wiedemann algorithm in Magma with a Tesla GPU, in: PASCO'15, ACM. pp. 2–6.
Storjohann, A., 2003. High-order lifting and integrality certification. J. Symbolic Comput. 36, 613–648.
The FFLAS-FFPACK Group, 2019. FFLAS-FFPACK: Finite Field Linear Algebra Subroutines / Package, version 2.3.2. http://github.com/linbox-team/fflas-ffpack.
The LinBox Group, 2018. Linbox: Linear algebra over black-box matrices, version 1.5.3. https://github.com/linbox-team/linbox/.
Turner, W.J., 2002. Black box linear algebra with the LINBOX library. Ph.D. thesis. North Carolina State University.

Van Barel, M., Bultheel, A., 1992. A general module theoretic framework for vector M-Padé and matrix rational interpolation. Numer. Algorithms 3, 451–462.
Villard, G., 1997a. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems, in: ISSAC'97, ACM. pp. 32–39.
Villard, G., 1997b. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials. Technical Report. LMC-IMAG, Report 975 IM.
Wiedemann, D., 1986. Solving sparse linear equations over finite fields. IEEE Trans. Inf. Theory IT-32, 54–62.
Wolovich, W.A., 1974. Linear Multivariable Systems. volume 11 of *Applied Mathematical Sciences*. Springer-Verlag New-York.