Linear algebra for skew-polynomial matrices

Sergei Abramov — Manuel Bronstein

N° 4420

March 2002

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4420--FR+ENG

Linear algebra for skew-polynomial matrices

Sergei Abramov[*] , Manuel Bronstein[†]

Thème 2 — Génie logiciel
et calcul symbolique
Projet Café

Rapport de recherche n° 4420 — March 2002 — 18 pages

Abstract: We describe an algorithm for transforming skew-polynomial matrices over an Ore domain in row-reduced form, and show that this algorithm can be used to perform the standard calculations of linear algebra on such matrices (ranks, kernels, linear dependences, inhomogeneous solving). The main application of our algorithm is to desingularize recurrences and to compute the rational solutions of a large class of linear functional systems. It also turns out to be efficient when applied to ordinary commutative matrix polynomials.

Key-words: computer algebra, linear algebra, skew polynomials, differential systems, recurrence systems

[*] Computer Center of the Russian Academy of Science, Vavilova 40, Moscow 117967, abramov@ccas.ru
[†] INRIA – Projet Café, Manuel.Bronstein@inria.fr

Algèbre linéaire pour matrices de polynômes de Ore

Résumé : Nous décrivons un algorithme qui transforme une matrice de polynômes de Ore à coefficients dans un domaine de Ore en une forme réduite par lignes. Cet algorithme est utilisable pour faire des calculs standards d'algèbre linéaire (rang, noyau, dépendance linéaire, résolution d'équations) sur ce type de matrices. L'application principale de notre algorithme est la désingularisation de systèmes de récurrences linéaires, ainsi que le calcul des solutions rationnelles d'une grande classe de systèmes linéaires fonctionnels. Il s'avère aussi être efficace sur des matrices de polynômes commutatifs usuels.

Mots-clés : calcul formel, algèbre linéaire, polynômes de Ore, systèmes différentiels, systèmes aux différences finies

Introduction

In order to bound the degree of polynomial solutions of certain linear systems of functional equations, we used in our ISSAC'2001 paper [3] a desingularizing transformation for linear recurrences systems. Given such a system $\sum_{k=l}^{h} M_k(n)Z_{n+k} = G(n)$, where the $M_k(n)$ are matrices with entries in a suitable integral domain, our transformation was used to ensure that $M_l(n)$ had full rank, which then allowed the bounding procedure to proceed. We study in this paper that transformation in greater depth, in particular extending it to coefficients in an arbitrary left Ore domain, which makes it applicable to multivariate skew-polynomial matrices, for example matrices over Weyl algebras. In addition, we study its additional properties and extend its range of applications to the standard calculations of linear algebra for skew-polynomial matrices: computing their ranks and kernels, as well as finding linear dependences and solving inhomogeneous linear systems. It turns out that our transformation is a generalisation to matrices of skew-polynomials of known row–reduction methods [6], so we study its arithmetic complexity in the commutative case, where it is comparable to (but not better than) some of the best methods known. Experimental benchmarks confirm this to be the case in practice for commutative matrix polynomials over finite fields. Finally, our ISSAC'2001 use is recalled as one of the applications of this transformation.

By convention, all rings and fields in this paper have characteristic 0 but are not necessarily commutative. Given a nonzero Laurent polynomial $p = \sum_i p_i X^i$ (with finite support), its *degree* is $\deg(p) = \max\{i \text{ s.t. } p_i \neq 0\}$, while its *valuation* is $\nu(p) = \min\{i \text{ s.t. } p_i \neq 0\}$. We use $[M]_i$ to denote the i^{th} row of the matrix M.

1 Ore domains and skew-polynomials

We recall in this section the basic definitions and properties of Ore domains and skew-polynomials, which are a common abstraction of differential and difference operators.

Definition 1 *A* left Ore domain *is a ring R without zero-divisors and such that every two nonzero elements of R have a nonzero common left multiple.*

Any left Ore domain R can be embedded in a field K of left-fractions of R [9]. The *rank* of a left R-module $\mathcal{M}$ is then defined to be the dimension of $K \otimes \mathcal{M}$ as a vector space over K, which corresponds to the maximal cardinality of R-linearly independent subsets of $\mathcal{M}$. Given a matrix M with entries in R, we use $\mathrm{rk}_R(M)$ to denote the rank of the left R-module generated by the rows of M, and omit the subscript when it is clear from the context.

Any commutative integral domain is obviously a left Ore domain. A classical noncommutative left Ore domain is the ring of skew-polynomials, which we proceed to describe. Let R be a ring and σ an injective endomorphism of R. A *σ-derivation* is a map $\delta : R \to R$ satisfying

$$\delta(a+b) = \delta a + \delta b \quad \text{and} \quad \delta(ab) = (\sigma a)(\delta b) + (\delta a)b \quad \text{for any } a, b \in R\,.$$

Note that the map 0_R that sends every $a \in R$ to 0 is a σ-derivation. Let X be an indeterminate over R. The *skew-polynomial ring over* R, denoted $R[X;\sigma,\delta]$ is the ring of usual polynomials in X over R, with the usual polynomial addition and the multiplication given by $Xa = \sigma(a)X + \delta(a)$ for any $a \in R$. When $\delta = 0_R$, that ring is denoted $R[X;\sigma]$ and is called a *difference operator ring over* R. When $\delta = 0_R$ and σ is an automorphism of R, localizing $R[X;\sigma]$ at the powers of X, we obtain the *skew Laurent polynomial ring* $R[X,X^{-1};\sigma]$, which is the ring of Laurent polynomials in X over R, with the usual addition and the multiplication given by $Xa = \sigma(a)X$ and $X^{-1}a = \sigma^{-1}(a)X^{-1}$ for any $a \in R$.

Lemma 1 *If R is a left Ore domain and σ is an automorphism of R, then $R[X,X^{-1};\sigma]$ is a left Ore domain.*

Proof. By Corollary 1.1 of [8], the difference operator ring $R[X;\sigma]$ is a left Ore domain. Since every $p \in R[X,X^{-1};\sigma]$ can be written as $p = X^{-s}p'$ where $s \geq 0$ and $p' \in R[X;\sigma]$, it follows that $R[X,X^{-1};\sigma]$ has no zero divisors. Let $p,q \in R[X,X^{-1};\sigma]$, write $p = X^{-s}p'$ and $q = X^{-t}q'$ where $s,t \geq 0$ and $p',q' \in R[X;\sigma]$, and let $m \in R[X;\sigma]$ be a nonzero left common multiple of p' and q'. Then, $m = p''p' = q''q'$ for some $p'',q'' \in R[X;\sigma]$, which implies that $m = p''X^s p = q''X^t q$ is a nonzero left common multiple of p and q. □ The above result allows us to construct multivariate skew Laurent polynomial rings over a left Ore domain by iterating the univariate construction with several automorphisms. Our algorithm, described below, can be applied to matrices with entries in such rings by considering them univariate in their topmost variable. A more general construction of multivariate skew-polynomials, but with nonnegative exponents only, is described by [8], and sect. 4.1 describes how our algorithm can be applied to those.

2 Rank-revealing transformations

Let M be a matrix with entries in a ring S. The *elementary row operations* applicable to M are (i) applying a permutation to the rows, (ii) multiplying a row by an element of S that is not a zero-divisor, and (iii) adding a multiple of a row to another one. A *row transformation* is a finite sequence of elementary row operations. Each elementary row operation can be performed by multiplying M on the left by an *elementary matrix*, namely (i) a permutation of the identity matrix, (ii) the identity matrix with one entry on its diagonal replaced by an element of S that is not a zero-divisor, and (iii) the identity matrix with at most one nonzero entry outside its diagonal. Therefore, any row transformation can be seen as multiplying M on the left by a matrix T, which is a finite product of elementary matrices.

Definition 2 *Given a matrix M with entries in a left Ore domain R, a row transformation T is a* rank-revealing-transformation (RRT) for M *if rk(M) is exactly the number of nonzero rows of TM.*

Note that an RRT T for M immediately yields $\mathrm{rk}(M)$. In addition, the rows of T corresponding to the zero rows of TM form a basis for the left-kernel of M. Therefore, applying an RRT to the transpose of M yields a basis of its kernel.

The use of various RRTs is classic in commutative linear algebra: Gaussian elimination over commutative fields and its fraction–free variants over commutative integral domains [5] obviously satisfy the above definition since they produce a row-echelon form of M. For matrix polynomials over a commutative field, there are weaker RRTs that originate from linear control theory: some methods used to compute the row-reduced and Popov forms [11] are RRTs, as is the algorithm of [14] for computing weak Popov forms. We describe now a skew-variant of row-reduction that provides a fraction–free RRT for skew-polynomial matrices.

Let R be a left Ore domain, σ an automorphism of R, $R[X, X^{-1}; \sigma]$ a skew Laurent polynomial ring over R, and suppose that we can compute ranks and kernels of matrices with entries in R. Our RRT is described as algorithm 1 below, using the following notations: 1_n denotes the $n \times n$ identity matrix, B^t denotes the transpose of the matrix or vector B, and if B has entries in $R[X, X^{-1}; \sigma]$, $\deg_i(B)$ denotes the maximum degree in X of all the elements of the i^{th} row of B. Finally, for V any left R-submodule of R^k and J any subset of $\{1, \ldots, k\}$, $V_J = \{v \in V$ such that $v_j = 0$ for all $j \in J\}$ is the intersection of kernels of projections and is therefore a left submodule of V.

```
Algorithm 1: One–step trailing skew–RRT.
Input: An $n \times m$ matrix $M$ with entries in $R[X, X^{-1}; \sigma]$.
Output: An RRT $T$ for $M$ and the transformed matrix $TM$.
(1)   Write $M$ as $M = \sum_{k=l}^{h} M_k X^k$ where $l \le h$ and $M_l \neq 0$
(2)   $T \leftarrow 1_n$
(3)   $Z \leftarrow \{i \in \{1, \ldots, n\}$ s.t. the $i^{\text{th}}$ row of $M$ is $0\}$
(4)   while $\mathrm{Ker}(M_l^t)_Z \neq \{0\}$
(5)       $v \leftarrow$ a nonzero element of $\mathrm{Ker}(M_l^t)_Z$
(6)       $I \leftarrow \{i \in \{1, \ldots, n\}$ such that $v_i \neq 0\}$
(7)       Choose $i_0 \in I$ such that $\forall i \in I, \deg_{i_0}(M) \ge \deg_i(M)$
(8)       $A \leftarrow 1_n$ with ${i_0}^{\text{th}}$ row replaced by $X^{-1} v^t$
(9)       $T \leftarrow AT$, $M \leftarrow AM$, update $M_l, \ldots, M_h$
(10)      if the ${i_0}^{\text{th}}$ row of $M$ is $0$ then $Z \leftarrow Z \cup \{i_0\}$
(11)  return $(T, M)$
```

Note the following remarks about the steps of algorithm 1:

- In step (1), the M_k are matrices with entries in R.
- Computing $\mathrm{Ker}(M_l^t)_Z$ in step (5) is done by removing from M_l^t the columns indexed by Z and computing the kernel of the resulting matrix.
- In step (5), a full basis of the kernel is not required, a single nonzero vector is sufficient. In addition, if R is itself a skew Laurent polynomial ring, then algorithm 1 can be used recursively to compute $\mathrm{Ker}(M_l^t)_Z$.

- Even though the matrix A of step (8) is more complex than a single elementary matrix, it is a row transformation because $v_{i_0} \neq 0$.
- The products AT and AM in step (9) are not actually computed as matrix products, they simply correspond to replacing the ${i_0}^{\text{th}}$ rows of T and M by the linear combination of rows given by $X^{-1}v$. Furthermore, if M is represented by the matrices $M_l, \dots, M_h$, then the product AM can be performed directly inside those matrices.
- The matrix T does not need to be updated in step (9) if only $\mathrm{rk}(M)$ is needed. Even if elements of $\mathrm{Ker}(M^t)$ are desired, one can store the sequence of pairs (v, i_0) used at each loop rather than update T, and use them to compute elements of the kernel after the algorithm has terminated.

We now proceed to prove termination and correctness of algorithm 1.

Lemma 2 *Algorithm 1 terminates after at most $n(h - \min(0,l) + 1)$ loops.*

Proof. Write $M = \sum_{k=l}^{h} M_k X^k$ before step (9), and let M' be M after that step. As noted above, M' is M with the ${i_0}^{\text{th}}$ row replaced by

$$X^{-1}v^t M = \sum_{k=l}^{h} X^{-1} v^t M_k X^k \,. \tag{1}$$

Since $v^t M_l = 0$, the starting point of the above sum is $k = l+1$. In addition, $v_i = 0$ whenever $\deg_i(M) > \deg_{i_0}(M)$, so $v^t M_k = 0$ for $k > \deg_{i_0}(M)$. Therefore, using the commutation rule $X^{-1}a = \sigma^{-1}(a)X^{-1}$ we get

$$X^{-1}v^t M = \sum_{k=l}^{h} X^{-1}v^t M_k X^k = \sum_{k=l+1}^{\deg_{i_0}(M)} X^{-1} v^t M_k X^k = \sum_{k=l+1}^{\deg_{i_0}(M)} \sigma^{-1}(v^t M_k) X^{k-1} \,.$$

It follows that either $X^{-1}v^t M = 0$ or $\deg_{i_0}(M') < \deg_{i_0}(M)$. In the first case, the cardinality of Z is increased in step (10), so that case can occur at most n times. In the second case, $\sum_i \deg_i(M') < \sum_i \deg_i(M)$, where the sums are taken over the nonzero rows. Since that sum is at most nh and at least 0 if $l \geq 0$, or nl if $l < 0$, that case can occur at most $n(h - \min(0,l))$ times. $\square$

Lemma 3 *rk(M) remains unchanged throughout algorithm 1.*

Proof. Write $M = \sum_{k=l}^{h} M_k X^k$ before step (9), and let M' be M after that step, and $\mathcal{M}'$ and $\mathcal{M}$ denote the left $R[X, X^{-1}; \sigma]$-modules generated by the rows of M' and M respectively. As noted above, M' is M with the ${i_0}^{\text{th}}$ row replaced by

$$[M']_{i_0} = X^{-1}v^t M = \sum_{i=1}^{n} X^{-1} v_i [M]_i \quad \in \mathcal{M},$$

so $\mathcal{M}'$ is a left $R[X,X^{-1};\sigma]$-submodule of $\mathcal{M}$. Proposition 9.3 of [9] then implies that

$$\mathrm{rk}(M) = \mathrm{rk}(M') + \mathrm{rk}(\mathcal{M}/\mathcal{M}')\,. \tag{2}$$

Let $w = \sum_{i=1}^{n} w_i [M]_i$ be an arbitrary element of $\mathcal{M}$. If $w_{i_0} = 0$, then $w \in \mathcal{M}'$. Otherwise, $w_{i_0} \neq 0$, so w_{i_0} and $X^{-1} v_{i_0}$ have a nonzero common left multiple $\alpha w_{i0} = \beta X^{-1} v_{i_0}$ where $\alpha, \beta \in R[X,X^{-1};\sigma]$ are nonzero. We then have

$$\begin{aligned} \alpha w &= \alpha w_{i_0}[M]_{i_0} + \sum_{i \neq i_0} \alpha w_i [M]_i = \beta X^{-1} v_{i_0} [M]_{i_0} + \sum_{i \neq i_0} \alpha w_i [M']_i \\ &= \beta \left([M']_{i_0} - \sum_{i \neq i_0} X^{-1} v_i [M']_i \right) + \sum_{i \neq i_0} \alpha w_i [M']_i \quad \in \mathcal{M}', \end{aligned}$$

which implies that $\mathcal{M}/\mathcal{M}'$ is a torsion module, hence of rank 0, and the lemma follows from (2). ☐

We finally conclude that algorithm 1 yields a rank-revealing transformation.

Theorem 1 *Let (T, M') be the result produced by algorithm 1 on the input matrix M. Then, rk(M) is the number of nonzero rows of M'. Furthermore, rk$_R(M'_l)$ = rk$_{R[X,X^{-1};\sigma]}(M)$ and the rows of T corresponding to the zero rows of M' form a basis of Ker(M^t).*

Proof. Let $[M']_{i_1}, \dots, [M']_{i_r}$ be the nonzero rows of M'. Since the algorithm terminated, we must have had $Z = \{1, \dots, n\} \setminus \{i_1, \dots, i_r\}$ and $\mathrm{Ker}(M_l'^t)_Z = \{0\}$ in step (4), which implies that $[M'_l]_{i_1}, \dots, [M'_l]_{i_r}$ are linearly independent over R, hence that $\mathrm{rk}_R(M'_l) \geq r$. However, the remaining rows of M'_l must be zero since they are zero in M', so $\mathrm{rk}_R(M'_l) = r$. Let now $\alpha_1, \dots, \alpha_r \in R[X,X^{-1};\sigma]$ be not all 0 and such that $\sum_{j=1}^{r} \alpha_j [M']_{i_j} = 0$, and $\beta_j = X^{-\nu} \alpha_j$ for all j, where $\nu = \min_{j \mid \alpha_j \neq 0}(\nu(\alpha_j))$. Then, $\beta_j \in R[X;\sigma]$ for all j and there is at least one j such that $\beta_j(0) \neq 0$. Multiplying the linear dependence on the left by $X^{-\nu}$ and on the right by X^{-l} we get

$$0 = \sum_{j=1}^{r} X^{-\nu} \alpha_j [M']_{i_j} X^{-l} = \sum_{j=1}^{r} \beta_j [M' X^{-l}]_{i_j}\,.$$

Since the β_j's are in $R[X;\sigma]$ as well as the entries of $M'X^{-l}$, evaluating the above at $X = 0$ yields

$$0 = \sum_{j=1}^{r} \beta_j(0) [M'_l]_{i_j}$$

in contradiction with $[M'_l]_{i_1}, \dots, [M'_l]_{i_r}$ linearly independent over R. Therefore, the r nonzero rows of M' are linearly independent over $R[X,X^{-1};\sigma]$, whence $\mathrm{rk}_{R[X,X^{-1};\sigma]}(M') = r$. Since M and M' have the same rank by lemma 3, we get $r = \mathrm{rk}_{R[X,X^{-1};\sigma]}(M) = \mathrm{rk}_R(M'_l)$. Since $M' = TM$, the rows of T corresponding to the zero rows of M' are elements of $\mathrm{Ker}(M^t)$. There are $n - \mathrm{rk}(M)$ such rows, which is exactly the dimension of $\mathrm{Ker}(M^t)$. Finally, the rows of a row transformation are linearly independent, so we obtain a basis of $\mathrm{Ker}(M^t)$. ☐

Corollary 1 *Let R be a left Ore domain, σ an automorphism of R and $R[X, X^{-1}; \sigma]$ a skew Laurent polynomial ring over R. If we can compute ranks and kernels of matrices with entries in R, then we can compute ranks and kernels of matrices with entries in $R[X, X^{-1}; \sigma]$.*

We only require in algorithm 1 the computation of one nonzero vector in a kernel. It is frequently the case that algorithms for computing such vectors return a full basis of the kernel, or several vectors. If fraction–free elimination algorithms exist for matrices with entries in R, then we can use several linearly independent kernel vectors in order to decrease the number of kernel computations performed by algorithm 1. Our modified RRT is described as algorithm 2 below.

```
Algorithm 2: Multi–step trailing skew–RRT.
Input: An $n \times m$ matrix $M$ with entries in $R[X, X^{-1}; \sigma]$.
Output: An RRT $T$ for $M$ and the transformed matrix $TM$.
(1)   Write $M$ as $M = \sum_{k=l}^{h} M_k X^k$ where $l \leq h$ and $M_l \neq 0$
(2)   $T \leftarrow 1_n$
(3)   $Z \leftarrow \{i \in \{1, \dots, n\}$ s.t. the $i^{\text{th}}$ row of $M$ is $0\}$
(4)   while $\text{Ker}(M_l^t)_Z \neq \{0\}$
(5)       $U \leftarrow s \times n$ matrix whose rows are $R$-linearly independent ele-
          ments of $\text{Ker}(M_l^t)_Z$
(6)       for $j \leftarrow 1$ to $s$
(7)           $v \leftarrow [U]_j$
(8)           $I \leftarrow \{i \in \{1, \dots, n\}$ such that $v_i \neq 0\}$
(9)           Choose $i_0 \in I$ such that $\forall i \in I, \deg_{i_0}(M) \geq \deg_i(M)$
(10)          $A \leftarrow 1_n$ with ${i_0}^{\text{th}}$ row replaced by $X^{-1} v^t$
(11)          $T \leftarrow AT$, $M \leftarrow AM$, update $M_l, \dots, M_h$
(12)          if the ${i_0}^{\text{th}}$ row of $M$ is 0 then $Z \leftarrow Z \cup \{i_0\}$
(13)          Compute a row transformation $E$ over $R$ such that the ${i_0}^{\text{th}}$
              column of $EU$ has zeroes in rows $j+1$ to $s$
(14)          $U \leftarrow EU$
(15)  return $(T, M)$
```

Fraction–free elimination over R, as used in step (13), certainly exists when R is a commutative integral domain [5], but also in the noncommutative case, provided that R is an *effective* left Ore domain in the sense of [8], *i.e.* that nonzero common left multiples can actually be computed in R with their cofactors. In that case, given a matrix B with entries in R an entry $b_{ij} \neq 0$ can be used as a pivot to eliminate the j^{th} column of B: for each $k \neq i$ such that $b_{kj} \neq 0$, computing $\alpha, \beta \in R$ such that $\alpha b_{ij} = \beta b_{kj} \neq 0$, then multiplying the k^{th} row of B by $-\beta$ and adding to it α times its i^{th} row is a pair of elementary row operations that brings a 0 at row k and column j. Termination and correctness of algorithm 2 follow from the following lemma.

Lemma 4 *$rk(U) = s$ throughout the inner loop of algorithm 2. In addition, the rows j to s of U are in $Ker(M_l^t)_Z$ throughout that loop.*

Proof. $\mathrm{rk}(U) = s$ at step (5) and it does not change when U is multiplied on the left by the row transformation E at step (14), so it remains s throughout the inner loop. When $j = 1$, the rows of U are in $\mathrm{Ker}(M_l^t)_Z$ by definition, so suppose now that the rows j to s of U are in $\mathrm{Ker}(M_l^t)_Z$ for a given $j < s$ at step (7). Let M' be M after step (11), Z' be Z after step (12) and $U' = EU$ be U after step (14). Since M_l and M_l' differ only at row i_0, and the ${i_0}^{\text{th}}$ entries of $[U']_{j+1}, \dots, [U']_s$ are 0, those rows are in $\mathrm{Ker}(M_l')$. In addition, the entries of $[U]_j, \dots, [U]_s$ whose indices are in Z are zero and $Z' \subseteq Z \cup \{i_0\}$, so $[U']_{j+1}, \dots, [U']_s$ are in $\mathrm{Ker}(M_l')_{Z'}$ and the lemma follows by induction. □

As a consequence of lemma 4, the element v of step (7) in algorithm 2 is always in $\mathrm{Ker}(M_l)_Z$, so lemmas 2 and 3 as well as theorem 1 remain valid for algorithm 2.

We note that there is also a "leading" variant of algorithm 1 that works with M_h rather than M_t: one uses $\mathrm{Ker}(M_h^t)_Z$ in steps (4) and (5), then picks an entry of v corresponding to a row of M of minimal valuation rather than maximal degree in step (7), which becomes

$$(7) \quad \text{Choose } i_0 \in I \text{ such that } \forall i \in I, \nu_{i_0}(M) \leq \nu_i(M)$$

where $\nu_i(M)$ denotes the minimum valuation in X of all the elements of the i^{th} row of M. Finally, X^{-1} is replaced by X in the definition of A in step (8), which becomes

$$(8) \quad A \leftarrow 1_n \text{ with } {i_0}^{\text{th}} \text{ row replaced by } Xv^t.$$

Lemmas 2 and 3 are easily seen to remain valid with the above modifications, as well as theorem 1, except that we now have $\mathrm{rk}_R(M_h') = \mathrm{rk}_{R[X,X^{-1};\sigma]}(M)$ instead of M_l', which is the essential reason for using the leading rather than the trailing version at times (as in section 4.3 below). Of course, the above modifications can be applied to algorithm 2 as well.

While we have mentionned only ranks and kernels in the discussion so far, our algorithm, like any RRT, can be used to find linear dependencies between vectors with entries in $R[X, X^{-1}; \sigma]$, as well as to solve inhomogeneous systems of the form $MZ = b$, since this can be reduced to finding the kernel of the augmented matrix $[M \mid b]$.

3 Complexity and experimental results

Let $M = \sum_{k=l}^{h} M_j X^k$ be a matrix with entries in $R[X, X^{-1}; \sigma]$ and $d = h - min(0, l) + 1$. The number of loops of algorithm 1 at most nd by Lemma 2, so we only need to count the cost of each loop. Using formula (1) for updating M at step (9), we must compute d products of the form $v^t M_k$, each costing nm multiplications of R. Counting an application of σ^{-1} to be one operation in R, left-multiplying each $v^t M_k$ by X^{-1} also costs nm operations, so step (9) has an arithmetic complexity of $\mathcal{O}(nmd)$. Computing $\mathrm{Ker}(M_l^t)_Z$ can be done in $\mathcal{O}(n^2 m)$ operations in R when R is a commutative integral domain. When R is an effective left Ore domain, counting the computation of a nonzero common left multiple to be one operation in R, then noncommutative elimination also has an arithmetic complexity of

$\mathcal{O}(n^2m)$, so the worst-case arithmetic complexity of algorithm 1 is $\mathcal{O}(n^2md^2+n^3md)$. When $n=m=d=\mathcal{O}(\mu)$, the complexity of computing $\text{rk}(M)$ is then $\mathcal{O}(\mu^5)$. In the commutative univariate case, this is the same than the complexity of Chinese remaindering, although we expect row–reduction to perform somewhat better because proving the rank with Chinese remaindering always requires nd modular images, while the bound of Lemma 2 is generally pessimistic. Computing the rank with the weak Popov form of [14] has a complexity of $\mathcal{O}(nmd^2\text{rk}(M))$, which is better than row–reduction when $\text{rk}(M) << \mu$, but is the same when $\text{rk}(M)=\mathcal{O}(\mu)$.

Given the similarities in arithmetic complexity with the above two methods, algorithm 1 has been implemented on top of the Σ^{it} library [7] by G. Chatley (`chatley@iitk.ac.in`) and extensive benchmarks carried out. For polynomials over finite fields (where the arithmetic and binary complexity are the same) the results, shown in figure 1 below, confirm the above analysis, namely that when n, m, d and $\text{rk}(M)$ all have the same orders, algorithm 1 lies between weak Popov forms and Chinese remaindering, the timings being proportional with small constant ratios (less than 2). In the case of full rank matrices, algorithm 1 and Chinese

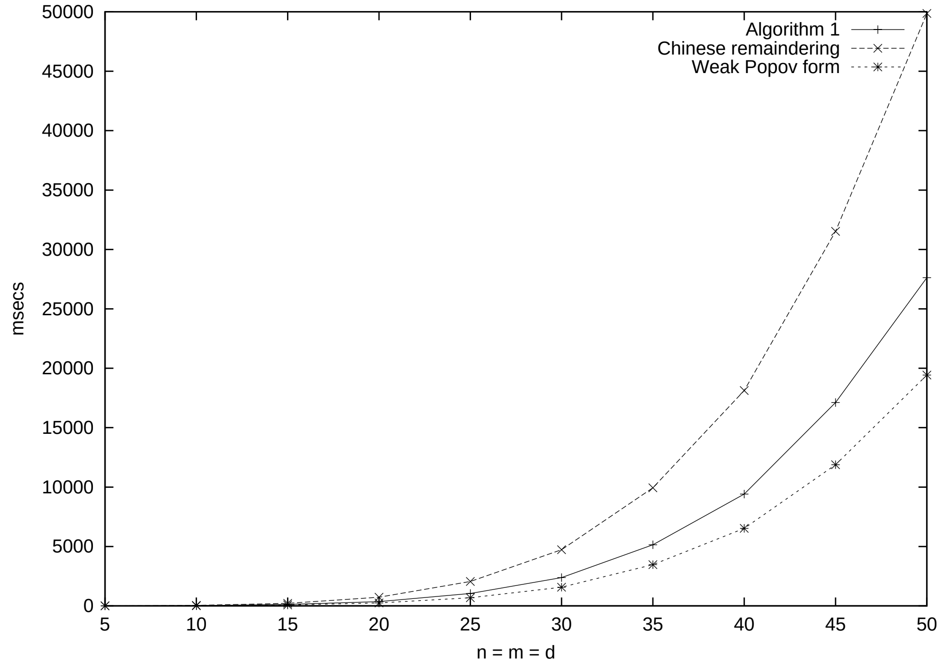

Figure 1: Results for square matrices of rank $n-1$ over $F[x]$, F a finite field.

remaindering outperformed the weak Popov form by an order of magnitude, since the rank can be proven full after a "small" number of loops (see theorem 2 below). So the fastest overall approach in practice is either to do a random evaluation check and proceed with weak Popov form if the rank is not full, or to apply row–reduction directly in all cases. Also as expected, the above three methods all outperform fraction–free elimination by one order of magnitude, as illustrated by figure 2 below, where the timings for two–step fraction–free elimination on the same examples have been added.

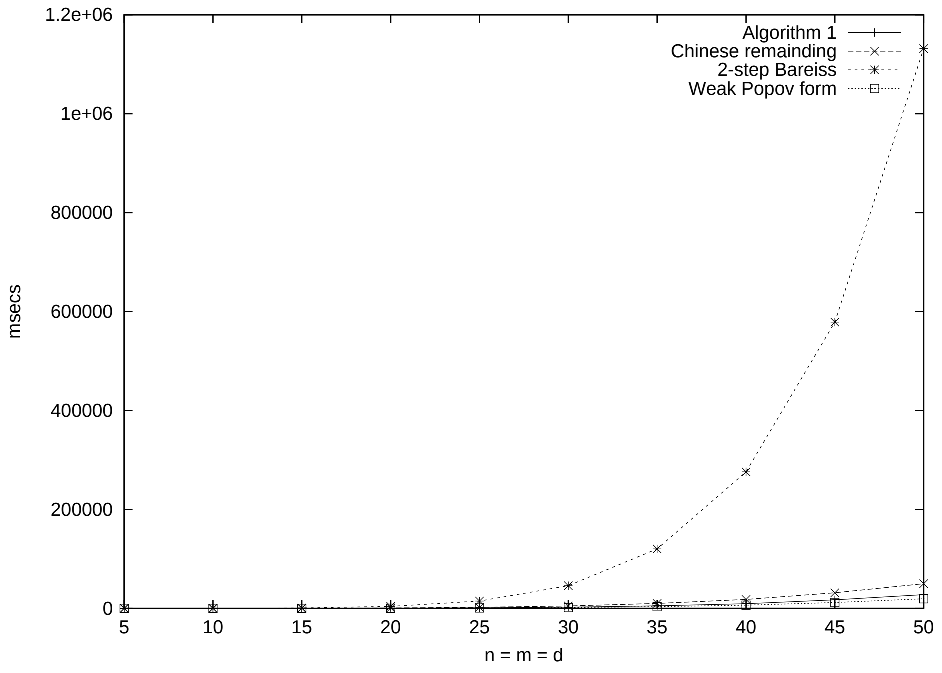

Figure 2: Comparison of two–step fraction–free elimination to the curves of figure 1.

Unlike the weak Popov form, row–reduction does not require the coefficients to be from a field. However, as for the weak Popov form, it suffers from growth of the coefficients in R, so its practical usefulness is either for coefficients in finite fields, where such growth does not occur, or for matrices of skew-polynomials, for which the other fast methods are not applicable.

4 Applications

4.1 Matrices over Weyl algebras

Corollary 1 means that our algorithms are applicable so far to nested skew Laurent polynomials. Such rings are however isomorphic to localisations of the Weyl algebras, so we can apply our algorithms to perform linear algebra on matrices over Weyl algebras. Let C be a commutative field and $A_m(C)$ be the Weyl algebra $C[x_1, \dots, x_m, \partial_1, \dots, \partial_m]$ where the product is given by the commutation rules

$$\partial_i \partial_j = \partial_j \partial_i, \qquad x_i x_j = x_j x_i, \qquad \text{and} \qquad \partial_i x_j - x_j \partial_i = \delta_{ij}\,,$$

where δ_{ij} is 1 if $i = j$ and 0 if $i \neq j$. Let $C[n_1, \dots, n_m]$ be the usual polynomial ring in m variables, and $R_m(C)$ be the nested skew Laurent polynomial ring

$$R_m(C) = C[n_1, \dots, n_m][X_1, X_1^{-1}; \sigma_1] \cdots [X_m, X_m^{-1}; \sigma_m]$$

where σ_i is the automorphism of $R_m(C)$ over C defined by $\sigma_i(n_j) = n_j + \delta_{ij}$ and $\sigma_i(X_j) = X_j$. Then, the map

$$\phi_m : C[x_1, x_1^{-1}, \dots, x_m, x_m^{-1}, \partial_1, \dots, \partial_m] \to R_m(C)$$

given by

$$\phi_m(x_i) = X_i^{-1} \quad \phi_m(x_i^{-1}) = X_i \quad \text{and} \quad \phi_m(\partial_i) = (n_i + 1) X_i \tag{3}$$

extends to a C-algebra isomorphism between those two left Ore domains. Therefore, ranks and kernels of matrices with entries in $A_m(C)$ can be computed by applying ϕ_m, using algorithm 1 or 2 on their images, and applying ϕ_m^{-1} to the basis of the kernels (although this gives generators in the localisation, they can be multiplied by suitable powers of the x_i's to obtain generators in $A_m(C)$). As mentioned earlier, this means that we can also find linear dependences over $A_m(C)$ and solve linear systems with coefficients in $A_m(C)$.

4.2 Deterministic x-adic lifting for solving linear systems

Let F be a commutative field and $F[X]$ a commutative univariate polynomial ring over F. Taking σ to be identity on F, algorithms 1 and 2 are applicable to matrices with entries in $F[X]$. Let A be a nonsingular $n \times n$ matrix with entries in $F[X]$ and $b \in F[X]^n$ be given. The asymptocally fastest way to compute the unique solution $z \in F(x)^n$ of $Az = b$ is by using p-adic lifting [10], where the p-adic expansion of z is computed for an irreducible $p \in F[X]$ that does not divide $\det(A)$. Since the computations are done in $F[X]/(p)$, choosing $p = X - \alpha$ for some $\alpha \in F$ is preferable. When F is large enough, a suitable α can be chosen at random, but this could be impossible over small finite fields, where a higher-degree p may be required. Our algorithm can be used as a nonsingular alternative to the singular x-adic lifting of [15] in the following way: let (T, A') be the result of applying algorithm 1 or 2 to A. Then, $A' = TA$ and theorem 1 implies that $\mathrm{rk}_F(A'(0)) = n$ since A is nonsingular. Therefore, $A'(0)$ can be inverted in F and nonsingular x-adic lifting can be applied to the modified system $A'z = Tb$, whose unique solution z is also the unique solution of $Az = b$. In practice, we do not need to compute explicitly the RRT T, it is sufficient to carry out on b the elementary row operations being carried out on A throughout the algorithm, which yields Tb.

In the commutative polynomial case, the number of loops performed by the algorithm on nonsingular inputs can be given quite precisely (this result was already presented as Lemma 3.5 of [6], where commutative row–reduction was described).

Theorem 2 *Let R be a commutative integral domain and M be a nonsingular square matrix with entries in the commutative polynomial ring $R[X]$. If X does not divide every entry in M, then algorithm 1 terminates after exactly N loops, where $N \geq 0$ is such that $X^N \mid \det(M)$ and $X^{N+1} \nmid \det(M)$.*

Proof. Since X does not divide every entry in M, then $M(0) \neq 0$, so write $M = \sum_{k=0}^{h} M_k X^k$ before step (9), and let M' be M after that step. Then, $M' = AM$ where A is the matrix computed at step (8). It follows that $EM' = VM$ where E is the identity matrix with the i_0th diagonal element replaced by X and V is the identity matrix with the i_0th row replaced by v^t. Noting that $\det(V) = v_{i_0} \neq 0$ and taking determinants on both sides, we get

$$X \det(M') = v_{i_0} \det(M)\,,$$

so a power of X is divided out of $\det(M)$ every pass through the loop. Theorem 1 implies that $(TM)(0)$ is nonsingular, where T is the row tranformation produced by the algorithm, therefore $\det(TM)$ is not divisible by X, which implies that we go exactly N times through that loop, where $X^N \mid \det(M)$ and $X^{N+1} \nmid \det(M)$. □

Since each loop in algorithm 1 costs $\mathcal{O}(n^2 d + n^3)$ operations in F (see sect. 3) and nonsingular x-adic lifting has a complexity of $\mathcal{O}(n^3 d^{1+\epsilon})$ where $0 < \epsilon \leq 1$ depends on the multiplication algorithm in $F[X]$, we see that our desingularisation procedure does not change the complexity as long as $N << nd$, which is generally[1] the case. We then get the same arithmetic complexity than [15], but we expect nonsingular x-adic lifting to have less overhead in practice than their algorithm.

4.3 Desingularisation of linear recurrence systems

This application of our algorithm was described in [3]. Let C be a commutative field, $C^{\mathbb{Z}}$ be the commutative ring of functions from $\mathbb{Z}$ to C and σ be the shift automorphism of $C^{\mathbb{Z}}$ given by $(\sigma f)(n) = f(n+1)$ for all $f \in C^{\mathbb{Z}}$. Let R be a subring of $C^{\mathbb{Z}}$ satisfying the following properties:

(i) R is an integral domain.

(ii) R is closed under σ.

[1]There are of course matrices for which $N \approx nd$, but there are as many matrices for which approximately nd random points must be tried before a nonsingular reduction is found.

(iii) $\forall f \in R \setminus \{0\}, \{n \in \mathbb{Z}$ s.t. $f(n) = 0\}$ is finite and can be computed.

The classical example of such a ring is the polynomial ring $R = C[n]$, but rings such as $C[q^n]$ or $C[n, q^n]$ where $q \in C^*$ is not a root of unity also have those properties [2]. Viewing the elements of R as C-valued sequences, consider the system of linear recurrence equations

$$\sum_{k=l}^{h} M_k(n) Z_{n+k} = G(n) \quad \text{for all } n > \mu \tag{4}$$

where the $M_k(n)$ are $p \times q$ matrices with entries in R, $M_l(n)$ and $M_h(n)$ are not identically 0, $G(n)$ is a vector with entries in $C^{\mathbb{Z}}$, and μ is either a fixed integer or $-\infty$, in which case the recurrences are valid for all $n \in \mathbb{Z}$.

We say that $m \in \mathbb{Z}$ is a *singularity* of the system (4) if $\mathrm{rk}(M_h(m)) < q$. When $p = q$ and m is not a singularity, then (4) can be used to compute uniquely Z_{m+h} given $Z_{m+l}, \dots, Z_{m+h-1}$, so we are interested in systems having finitely many singularities. When $p < q$, then every $m \in \mathbb{Z}$ is a singularity, so suppose that $p \ge q$. In that case, our algorithm can be used to transform the system, when it is possible, into an equivalent one with finitely many singularities, thereby "desingularizing" it: consider the skew Laurent polynomial ring $R[X, X^{-1}; \sigma]$ and the $p \times q$ matrix $M = \sum_{k=l}^{h} M_k X^k$. Applying the "leading" variant of algorithm 1 or 2 to M yields a row transformation T and $M' = TM$ such that $\mathrm{rk}_R(M'_h) = \mathrm{rk}_{R[X,X^{-1};\sigma]}(M) = r$ and M' has exactly r nonzero rows. If $r < q$, then (4) is underdetermined and cannot be desingularized. If $r = q$, then M'_h has at least one nonzero $q \times q$ minor, and the singularities of (4) must be among its finite set of zeroes. Furthermore, M' yields the system

$$\sum_{k=l}^{h} M'_k(n) Z_{n+k} = G'(n) \quad \text{for all } n > \mu \tag{5}$$

where $G'(n)$ is the result of updating $G(n)$ inside the loop of algorithm 1 or 2 via $G \leftarrow v^t G$. If any zero row of M' corresponds to a nonzero entry of $G'(n)$, then (5) has no solutions. Otherwise, taking the q nonzero rows of M' and the corresponding entries in $G'(n)$ turns (5) into a square system of full rank. By construction, it is clear that any solution of (4) must be a solution of (5), but the converse is not necessarily true. To recover the solutions of (4) from those of (5), we must add the following steps to algorithm 1 or 2: first initialize a set of constraints $B \leftarrow \emptyset$, then inside the loop add to B the linear constraint

$$\sum_{k=l}^{h} [M_k(m)]_{i_0} Z_{m+k} = G_{i_0}(m)$$

for each $m \in \mathbb{Z}$ such that $v_{i0}(m) = 0$. At the end, the solutions of (4) are exactly the solutions of (5) that satisfy all the constraints in B.

Another question that arises whenever the solutions of (4) are the coefficients of some sort of series expansions is whether it has solutions of finite support. The following correction to Theorem 4 of [3] gives an upper bound on the support of such solutions when the system is not underdetermined.

Theorem 3 *Let $d \in \mathbb{Z}$ be such that $G(d) \neq 0$ and $G(n) = 0$ for all $n > d$ ($d = -\infty$ if G is identically 0), and suppose that (4) has a nonzero solution Z such that $Z_N \neq 0$ for some $N \in \mathbb{Z}$ satisfying $Z_n = 0$ for all $n > N$. Then, either $N \leq l + \max(\mu, d)$ or $rk(M_l(N-l)) < q$.*

Proof. Suppose that $N > l + \max(\mu, d)$. Then, $N - l > \mu$, so applying (4) to $n = N - l$ yields

$$\sum_{k=l+1}^{h} M_l(N-l) Z_{N+k-l} + M_l(N-l) Z_N = G(N-l)\,.$$

For $k > l$, $N + k - l > N$, so $Z_{N+k-l} = 0$. In addition, $N - l > d$, so $G(N-l) = 0$ and we obtain $M_l(N-l)Z_N = 0$. Since $Z_N \neq 0$, we must have $\mathrm{rk}(M_l(N-l)) < q$. □
The condition $N \leq l + \max(\mu, d)$ yields a finite number of positive values for N, while the rank condition is a problem similar to desingularisation: if $p < q$, then the system is underdetermined and no bound can be found. Otherwise, applying the "trailing" variant of algorithm 1 or 2 to $M = \sum_{k=l}^{h} M_k X^k$ yields a row transformation T and $M' = TM$ such that $\mathrm{rk}_R(M'_l) = \mathrm{rk}_{R[X,X^{-1};\sigma]}(M) = r$ and M' has exactly r nonzero rows. If $r < q$, then (4) is underdetermined and no bound can be found. If $r = q$, then M'_l has at least one nonzero $q \times q$ minor, and all the values of $N - l$ such that $\mathrm{rk}(M'_l(N-l)) < q$ must be among its finite set of zeroes. However, when $\mu \neq -\infty$, its value changes during the algorithm, so the first bound $N \leq l + \max(\mu, d)$ has to be updated as follows: initialize $\mu_i \leftarrow \mu$ for $1 \leq i \leq p$, then update μ_{i_0} inside the loop via $\mu_{i_0} \leftarrow 1 + \max_{i \in I}(\mu_i)$. At the end of algorithm, return $\mu' = \max_i \mu_i$. The finite set of bounds for the solutions of (4) is then given by

$$N \leq l + \max(\mu', d) \quad \text{or} \quad \mathrm{rk}(M'_l(N-l)) < q\,.$$

Finally, we note that when R is the polynomial ring $C[n]$, there are fast modular algorithms for computing the kernels required by algorithms 1 and 2 [13, 15]. Since those methods have better complexity than Gaussian elimination in $C[n]$ (see figure 2), using those methods inside our algorithm yields a better binary complexity than the EG–elimination of [1], which relies on "careful" Gaussian elimination.

4.4 Solving linear functional systems

This application, described in [3], relies on the following additional property of the isomorphism ϕ_1 given by (3) in the univariate case: for any differential operator $L \in A_1(C) = C[x, \partial_x]$ and any power series $y = \sum_{n \geq 0} y_n x^n \in C[[x]]$, the sequence $(\phi_1 L)(y_0, y_1, \ldots)$ is the coefficient sequence of $L(y)$ [4]. Therefore, the formal series solutions of a differential system $A(x, \partial_x)Y = F(x)$, where A is a matrix with entries in $A_1(C)$, can be found by solving the linear recurrence system $MZ = G$, where M is the matrix whose entries in $C[n][X, X^{-1}; \sigma]$ are the images of the entries of A by ϕ_1, and G is the sequence of coefficients of the formal series expansions of $F(x)$. When the system $AZ = F$ is not underdetermined, then the recurrence $MZ = G$ can be desingularized by the leading variant of our algorithm (see sect. 4.3), thereby allowing the formal Taylor series solutions to be computed. Furthermore, the *polynomial* solutions of $AZ = F$ correspond exactly to the series solutions with finite support of $MZ = G$, so an upper bound on the degrees of such solutions can be computed by the trailing variant of our algorithm as explained above. The desingularization procedure also yields a bound on the order of the pole at $x = 0$ of the *rational* solutions of $AZ = F$, so performing it at all the singularities of the system allows its rational solutions to be computed. In particular, differential systems of the form $Y' = A(x)Y + F(x)$ where $A(x)$ is a matrix with entries in $C(x)$ are of full rank, so their solutions can be computed using this approach.

This approach is not restricted to differential systems: choosing an appropriate persistent sequence of $C[x]$ as expansion basis for the power series, one can find isomorphisms with properties similar to those of ϕ_1 between other operator algebras and $R[X, X^{-1}; \sigma]$ for some suitable commutative integral domain R. This allows our approach to be also applied to difference and q-difference systems, as well as mixed differential/q-difference systems, we refer to [3] for additional details. Note finally that there are several choices for the basis of $C[x]$ to use, and that some of them are preferable since they yield recurrence systems with $\mu = -\infty$ in (4), thereby avoiding having to update μ during the bounding process. A more detailed discussion of basis selection for particular classes of equations is presented by [12].

References

[1] S.A. Abramov. EG–eliminations. *Journal of Difference Equations and Applications*, 5:393–433, 1999.

[2] S.A. Abramov and M. Bronstein. Hypergeometric dispersion and the orbit problem. In C. Traverso, editor, *Proceedings of ISSAC'2000*, pages 8–13. ACM Press, 2000.

[3] S.A. Abramov and M. Bronstein. On solutions of linear functional systems. In B. Mourrain, editor, *Proceedings of ISSAC'2001*, pages 1–6. ACM Press, 2001.

[4] S.A. Abramov, M. Petkovšek, and A. Ryabenko. Special formal series solutions of linear operator equations. *Discrete Mathematics*, 210:3–25, 2000.

[5] E.H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22:565–578, 1968.

[6] B. Beckermann and G. Labahn. Recursiveness in matrix rational interpolation problems. *Journal of Computational and Applied Mathematics*, 77:5–34, 1997.

[7] M. Bronstein. Σ^{it}– a strongly-typed embeddable computer algebra library. In J. Calmet, editor, *Proceedings of DISCO'96*, LNCS 1128, pages 22–33. Springer, 1996.

[8] F. Chyzak and B. Salvy. Non-commutative elimination in Ore algebras proves multivariate identities. *J. Symbolic Computation*, 26(2):187–228, August 1998.

[9] P.M. Cohn. *Free Rings and their Relations.* Academic Press, London, 1971.

[10] J.D. Dixon. Exact solution of linear equations using p-adic expansions. *Numerische Mathematik*, 40:137–141, 1982.

[11] T. Kailath. *Linear Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[12] D. Khmelnov. Basis selection for linear functional systems solutions. *Programmirovanie*, To appear, 2002.

[13] M.T. McClellan. The exact solution of systems of linear equations with polynomial coefficients. *Journal of the ACM*, 20:563–588, 1973.

[14] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. Technical Report 356, Dpt. of Computer Science, ETH Zurich, 2000.

[15] T. Mulders and A. Storjohann. Rational solutions of singular linear systems. In C. Traverso, editor, *Proceedings of ISSAC'2000*, pages 242–249. ACM Press, 2000.

Contents

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399