HAL
archives-ouvertes.fr

On the Complexity of Polynomial Matrix Computations

Pascal Giorgi, Claude-Pierre Jeannerod, Gilles Villard

▶ **To cite this version:**

Pascal Giorgi, Claude-Pierre Jeannerod, Gilles Villard. On the Complexity of Polynomial Matrix Computations. [Research Report] LIP RR-2003-2, Laboratoire de l'informatique du parallélisme. 2003, 2+15p. hal-02101878

HAL Id: hal-02101878
https://hal-lara.archives-ouvertes.fr/hal-02101878

Submitted on 17 Apr 2019

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

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

Laboratoire de l'Informatique du Parallélisme

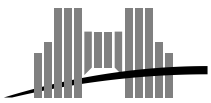
École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON n° 5668

On the Complexity of Polynomial Matrix Computations

Pascal Giorgi
Claude-Pierre Jeannerod
Gilles Villard

January 13, 2003

Research Report N° 2003-2



École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

On the Complexity of Polynomial Matrix Computations

Pascal Giorgi
Claude-Pierre Jeannerod
Gilles Villard

January 13, 2003

Abstract

We study the link between the complexity of polynomial matrix multiplication and the complexity of solving other basic linear algebra problems on polynomial matrices. By polynomial matrices we mean $n \times n$ matrices of degree d over $\mathsf{K}[x]$ where K is a commutative field. Under the straight-line program model we show that multiplication is reducible to the problem of computing the coefficient of degree d of the determinant. Conversely, we propose algorithms for minimal approximant computation and column reduction that are based on polynomial matrix multiplication; for the determinant, the straight-line program we give also relies on matrix product over $\mathsf{K}[x]$ and provides an alternative to the determinant algorithm of [16]. We further show that all these problems can be solved in particular in $O^{\sim}(n^{\omega}d)$ operations in K. Here the soft "O" notation indicates some missing $\log(nd)$ factors and ω is the exponent of matrix multiplication over K.

Keywords: Matrix polynomial, minimal basis, column reduced form, matrix gcd, determinant, polynomial matrix multiplication.

Résumé

On étudie le lien entre la complexité du produit de matrices polynomiales et celle d'autres opérations élémentaires de l'algèbre linéaire des matrices polynomiales. Par matrices polynomiales on entend des matrices $n \times n$ de degré d sur $\mathsf{K}[x]$, K étant un corps commutatif. Avec le modèle "straight-line" nous montrons que le produit se réduit au calcul du coefficient de degré d du déterminant. Réciproquement, on propose pour certains approximants matriciels minimaux et la forme colonne réduite des algorithmes basés sur le produit de matrices; pour le déterminant, on donne un programme "straight-line" également à base de produit matriciel, qui fournit une alternative à l'algorithme de [16]. On montre de plus que tous ces problèmes peuvent être résolus en particulier en $O^{\sim}(n^{\omega}d)$ opérations sur K. La notation $O^{\sim}$ contient des facteurs $\log(nd)$ et ω est l'exposant du produit de matrices sur K.

Mots-clés: Matrice polynomiale, base minimale, forme colonne réduite, pgcd matriciel, déterminant, produit de matrices polynomiales.

1 Introduction

The link between matrix multiplication and other basic linear algebra problems is well known under the algebraic complexity model. For K a commutative field, we will assume that the product of two $n \times n$ matrices over K can be computed in $O(n^\omega)$ operations in K. Under the model of computation trees over K, we know that ω is also the exponent of the problems of computing the determinant, the matrix inverse, the rank, the characteristic polynomial (we refer to the survey in [5, Chap.16]) or the Frobenius normal form [8, 15]. On an algebraic RAM, all these problems can be solved with $O^\sim(n^\omega)$ operations in K, hence the corresponding algorihms are optimal up to logarithmic terms. Here and in the rest of the paper, for any exponent e_1, $O^\sim(n^{e_1})$ denotes $O(n^{e_1} \log^{e_2} n)$ for any exponent e_2.

Much less is known for polynomial matrices and even less for integer matrices under the bit-complexity model. Difficulties come from the size of the data (and from carry propagation in the case of integer arithmetic) which make reductions between problems hard to obtain. In this paper we investigate the case of matrices of degree d in $\mathsf{K}[x]^{n\times n}$. This is motivated both by the interest in studying more concrete domains than abstract fields and by the two results [16] and [10, 21]. Storjohann has established and algorithm of cost $O^\sim(n^\omega d)$ for the determinant and the Smith normal form. We have shown that the matrix inverse can be computed by a straight-line program of length $O^\sim(n^3 d)$ (which is almost the size of the output). Besides, the latter estimate gives an alternative for the determinant in $O^\sim(n^\omega d)$ (see §4).

These two results first ask the following question: are problems on polynomial matrices – and especially the determinant problem – harder than polynomial matrix multiplication? By slightly extending the result of Baur & Strassen [1, Cor. 5] for matrices over a field, we answer positively for the determinant. We show in section 4 that if there is a straight-line program of length $D(n, d)$ over K which computes the coefficient of degree d of the determinant, then there is a straight-line program of length no more than $8D(n, d)$ which multiplies two $n \times n$ matrices of degree d.

Conversely, the second question is to know which polynomial matrix problems can be solved with roughly the same number of arithmetic operations than polynomial matrix multiplication. As seen above we already know that this is the case of the determinant problem [16] on an algebraic RAM using $O^\sim(n^\omega d)$ as an estimation of the complexity of matrix multiplication [6]. We will show in §4 that a different approach, which we have developed independently [10, 21], gives a straight-line program of length $O^\sim(n^\omega d)$ for the same problem.

Before studying the determinant, we first give analogous cost estimates with RAM programs for two other problems: we show in section 2 how to compute minimal bases and order d matrix approximants in $O^\sim(n^\omega d)$ operations in K; we show in section 3 that an invertible matrix can be column reduced in time $O^\sim(n^\omega d)$ as well. Note that here column reduction is roughly lattice basis reduction for $\mathsf{K}[x]$-modules.

We further study the complexities of each of the above problems in terms of general cost functions involving polynomial matrix multiplication. To do so, we denote by $\mathsf{MM}(n, d)$ the cost of multiplying two matrices of degree d in $\mathsf{K}[x]^{n\times n}$ (with $\mathsf{MM}(n) = \mathsf{MM}(n, 0)$) and assume without loss of generality that n and d are powers of two. As we shall see in section 2.2 our minimal basis algorithm works recursively on the degree and leads us to define the function

$$\mathsf{MM}'(n, d) = \sum_{i=0}^{\log d} 2^i \mathsf{MM}(n, 2^{-i} d).$$

This will be used in sections 2 and 3 for expressing the costs of matrix approximation and column reduction. In the same way, the determinant algorithms of section 4 work recursively on the

dimension and we give their complexities in terms of the function

$$\mathsf{MM}''(n,d) = \sum_{i=0}^{\log n} 2^i \mathsf{MM}(2^{-i}n, 2^i d).$$

For $\mathsf{MM}(n,d) = \Theta(n^\omega d \log d \log\log d)$ [6] these two functions further simplify. We have $\mathsf{MM}'(n,d) = O(\mathsf{MM}(n,d)\log d)$ for any value of ω and $\mathsf{MM}''(n,d) = O(\mathsf{MM}(n,d)\log n)$ if $\omega = 2$ and $\mathsf{MM}''(n,d) = O(\mathsf{MM}(n,d))$ if $\omega > 2$. Therefore

$$\mathsf{MM}'(m,d) = O\tilde{}(\mathsf{MM}(n,d)),\ \mathsf{MM}''(m,d) = O\tilde{}(\mathsf{MM}(n,d))$$

when taking $\mathsf{MM}(n,d) = \Theta\tilde{}(n^\omega d)$.

2 Minimal basis computation

Many problems on matrix polynomials reduce to computing minimal approximant bases (or σ-bases) [2]. Given a matrix power series $G \in \mathsf{K}[[x]]^{m\times n}$ and an approximation order $d \in \mathbb{N}$, these bases are nonsingular $m \times m$ polynomial matrices M such that

$$MG = O(x^d). \tag{1}$$

Minimality is made precise in Definition 2.1 below. It essentially expresses the fact that M has the smallest possible row degrees.

Minimal basis computations are motivated by the following two applications, which we shall develop later in the paper: in section 3 we will use the fact that the problem of column reducing a matrix can be solved by computing a Padé approximant and thus a minimal approximant basis; in section 4 we further use such approximants for recovering the polynomial matrix kernels that lead to the determinant. Note that a third application is the computation of minimal matrix polynomials of linearly generated matrix sequences as proposed in [23] and [20].

Our purpose in this section is to introduce polynomial matrix multiplication into the existing approximation algorithms. We achieve this by first adapting in §2.1 the σ-basis algorithm of Beckermann and Labahn [2] to exploit fast matrix mutiplication over K. Roughly, the algorithm of [2] works iteratively and computes a σ-basis from a $(\sigma - n)$-basis via n Gaussian elimination steps on vectors of K^m. How to replace these n steps on $m \times 1$ vectors with a single step on an $m \times n$ matrix was unclear. We solve this problem by resorting to the blocking approach of Coppersmith [7, 13] and using the matrix product-based LSP factorization algorithm of [9]. Polynomial matrix product then arises in §2.2 with a divide-and-conquer version of the method of §2.1 that generalizes the previous studies in [2, 19].

This divide-and-conquer version yields the cost of $O\tilde{}(n^\omega d)$ which improves upon the cost of $O\tilde{}(n^3 d)$ in [2, Theorem 6.2].

To define the type of approximants we compute, we consider as in [2] the formal power series vector

$$f(x) = G(x^n)[1, x, \ldots, x^{n-1}]^T \in \mathsf{K}[[x]]^m;$$

we further call (approximation) order of $v^T \in \mathsf{K}[x]^m$ and denote by $\operatorname{ord} v$ the integer

$$\operatorname{ord} v = \sup\{\tau \in \mathbb{N} \,:\, v(x^n)f(x) = O(x^\tau)\}.$$

For $\sigma \in \mathbb{N}$ we then define σ-bases with respect to the rows of G as follows.

Definition 2.1 *A σ-basis of G is a matrix polynomial M in $\mathsf{K}[x]^{m\times m}$ verifying:*

I) $\operatorname{ord} M^{(i,*)} \geq \sigma$ *for* $1 \leq i \leq m$;

II) *every* $v \in \mathsf{K}[x]^m$ *such that* $\operatorname{ord} v \geq \sigma$ *admits a unique decomposition* $v^T = \sum_{i=1}^m c^{(i)} M^{(i,*)}$ *where, for* $1 \leq i \leq m$, $c^{(i)} \in \mathsf{K}[x]$ *and* $\deg c^{(i)} + \deg M^{(i,*)} \leq \deg v$ *(minimality of the approximant).*

This definition coincides with [2, Definition 3.2, p.809] when the m components of the multiindex in [2] are the same. Also, since I) yields $M(x^n)G(x^n)[1, x, \ldots, x^{n-1}]^T = O(x^\sigma)$, it suffices to take $\sigma = nd$ to get approximant $M(x)$ in (1).

2.1 Via matrix multiplication

To introduce matrix multiplication into σ-basis computations of [2], we use the so-called LSP factorization [3, p. 103]: every matrix $A \in \mathsf{K}^{m\times m}$ of rank r can be written as $A = LSP$ where $L \in \mathsf{K}^{m\times m}$ is lower triangular with ones on the diagonal, $S \in \mathsf{K}^{m\times m}$ has $m - r$ zero rows and $P \in \mathsf{K}^{m\times m}$ is a permutation matrix; additionally, the nonzero rows of S form an $r \times m$ upper triangular matrix with nonzero diagonal entries. Let

$$1 \leq i_1 < i_2 < \cdots < i_r \leq m$$

be the indices of the nonzero rows of S. Each i_j is then uniquely defined as the smallest index such that the first i_j rows of A have rank j.

In Algorithm `SigmaBasis` below, we assume we compute LSP factorizations with the algorithm of [9] as described in [3, p. 103]: L, S, P are computed in $O(\mathsf{MM}(m))$ operations in K; furthermore, L, S are such that if the ith row of S is identically zero then the ith column of L is the ith unit vector. (See Appendix A.)

Algorithm `SigmaBasis`(G, d)
Input: $G \in \mathsf{K}[[x]]^{m\times n}$ with $m \geq n$ and $d \in \mathbb{N}$.
Output: a σ-basis $M \in \mathsf{K}[x]^{m\times m}$ with $\sigma = nd$.

$M := I_m$;
$\delta := 0 \in \mathbb{N}^m$;
for k from 1 to d do
  $\delta := \pi\delta$ where $\pi \in \mathsf{K}^{m\times m}$ sorts δ in decreasing order;
  $\Delta := x^{-(k-1)}\pi M G \bmod x$;
  $\Delta := \Delta$ augmented with $m - n$ zero columns;
  Compute the LSP factorization of Δ;
  $D := \operatorname{diag}(d_1, \ldots, d_m)$ where $d_i = x$ if $i \in \{i_1, \ldots, i_r\}$
    and $d_i = 1$ otherwise;
  $M := DL^{-1}\pi M$;
  $\delta := \delta + [d_1(0) - 1, \ldots, d_m(0) - 1]^T$;
od;
return M;

Lemma 2.2 *Algorithm* `BlockSigmaBasis` *is correct. Its cost is $O(\mathsf{MM}(m)d^2)$ or $O(n^\omega d^2)$ operations in K.*

Proof. Let $M_0 = I_m$ and, for $1 \leq k \leq d$, write $M_k(x)$ for the matrix M computed by step k. We see that the degree of M_k in x is no more than k and, assuming the algorithm is correct,

that $M_k G = O(x^k)$. The computation of Δ at step k thus costs $O(\mathsf{MM}(m)k)$ field operations. This dominates the cost of step k, for both LSP factorization and the update of M require only $O(\mathsf{MM}(m))$ operations in K. The overall complexity then follows.

To prove the algorithm is correct, note first that M_0 is a 0-basis of $G(x)$. Then, assuming for $k \in \{1, \ldots, m\}$ that $M_{k-1}(x)$ is an $n(k-1)$-basis of $G(x)$, we verify that $M_k(x) = D(x)L^{-1}\pi M_{k-1}(x)$ is an nk-basis of $G(x)$.

Let $N_{k-1}(x) = \pi M_{k-1}(x)$ and recall that P is the permutation matrix in the LSP factorization at step k. It follows that $N_{k-1}(x)$ is an $n(k-1)$-basis of $G(x)P^{-1}$. Algorithm FPHPS of [2, p. 810] with input parameters m, n,

$$F(x) = N_{k-1}(x^n)G(x^n)P^{-1}[1, x, \ldots, x^{n-1}]^T$$

and $(0, \ldots, 0) \in \mathbb{N}^m$ then returns an nk-basis of $G(x)P^{-1}$ after n steps. We denote this basis by $N_k(x)$. (Uniqueness of the output of FPHPS is explained in [2, p. 818].) As shown below, the two bases are related as

$$N_k(x) = D(x)L^{-1}N_{k-1}(x) \tag{2}$$

and hence $M_k = N_k$ is an nk-basis of GP^{-1} and G as well.

We now prove identity (2). Let $\Lambda = \Delta P^{-1}$ and let Λ_j be the jth column of Λ. Then

$$x^{-(k-1)n}F(x) \equiv \Lambda_1 + x\Lambda_2 + \cdots + x^{n-1}\Lambda_n \bmod x^n.$$

Since the rows of N_{k-1} have been sorted by permutation π, the first step of FPHPS simply consists in picking the first nonzero entry of Λ_1 – say, λ_1 with row index h_1 – and zeroing the lower entries of Λ_1 by using pivot λ_1. The h_1st row is then multiplied by x. In other words, $N_{k-1}(x)$ is transformed into $E_1(x)T_1N_{k-1}(x)$ where we define $E_1(x) = \mathrm{diag}(I_{h_1-1}, x, I_{m-h_1})$ and

$$T_1 = \left[\begin{array}{c|c|c} I_{h_1-1} & & \\ \hline & 1 & \\ \hline & t_1 & I_{m-h_1} \end{array}\right] \text{ with } t_1 \in \mathsf{K}^{m-h_1}. \tag{3}$$

Recalling that i_1 is the index of the first nonzero row of S in factorization $\Lambda = LS$, we verify first that $h_1 = i_1$: it follows from Appendix A that L and S can be partitioned as

$$\Lambda = LS = \left[\begin{array}{c|c|c} I_{i_1-1} & & \\ \hline & 1 & \\ \hline & l_1 & L' \end{array}\right] \left[\begin{array}{c|c} & \\ \hline \lambda_1 & s_1^T \\ \hline & S' \end{array}\right], \quad \lambda_1 \in \mathsf{K}\backslash\{0\}.$$

Here $L' \in \mathsf{K}^{(m-i_1)\times(m-i_1)}$, $S' \in \mathsf{K}^{(m-i_1)\times(n-1)}$ and λ_1 is indeed the first nonzero entry of Λ_1. Hence $h_1 = i_1$. Second, comparing the first column in both members of $T_1\Lambda = T_1LS$ yields $t_1 = -l_1$ and the i_1st column of T_1^{-1} is thus equal to the i_1st column of L. The first step of FPHPS yields eventually

$$x^{-(k-1)n}E_1(x^n)T_1F(x) \equiv x\Lambda_2' + \cdots + x^{n-1}\Lambda_n' \bmod x^n$$

where

$$[0|\Lambda_2'|\cdots|\Lambda_n'] = E_1(0)T_1\Lambda = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & L'S' \end{array}\right] \in \mathsf{K}^{m\times n}. \tag{4}$$

Let h_2 be the pivot index at step 2 and let T_2 and $E_2(x)$ be the associated transformation matrices. It follows from (4) that $h_2 > i_1$. Hence T_2 has the form $T_2 = \mathrm{diag}(I_{i_1}, T_2')$ and $E_1(x)$ commutes with T_2. Then, noticing that the ordering imposed by π is still the same, one can iterate by replacing T_1, LS and $i_1 < \cdots < i_r$ with respectively T_2', $L'S'$ and $i_2 - i_1 < \cdots < i_r - i_1$. We

eventually get $h_j = i_j$ for $1 \le j \le r$. Therefore, defining for $1 \le j \le r$ matrices $E_j(x)$ and T_j has done in (3) for $j = 1$, we have

$$N_k(x) = E_r(x) \cdots E_2(x) E_1(x) T_r \cdots T_2 T_1 N_{k-1}(x). \tag{5}$$

It follows that $E_r(x) \cdots E_2(x) E_1(x) = D(x)$ and that the i_jth column of T_j^{-1} equals the i_jth column of L. Noticing further that because of the structure of T_j the i_jth column of T^{-1} equals the i_jth column of T_j^{-1}, we have $T^{-1} = L$ and (2) follows from (5). □

2.2 Via polynomial matrix multiplication

To use polynomial matrix multiplication, we now give a divide-and-conquer version of Algorithm `SigmaBasis`. This version is based on the following "transitivity lemma", which may be seen as the counterpart of Theorem 6.1 in [2] and can be shown in the same way.

Lemma 2.3 *If M, M', M'' are the output of Algorithm* `SigmaBasis` *for input (G, d), $(G, d/2)$, $(x^{-d/2} M' G, d/2)$ respectively then $M = M'' M'$.*

Algorithm `BlockSigmaBasis`(G, d)
Input: $G \in \mathsf{K}[[x]]^{m \times n}$ with $m \ge n$ and $d \in \mathbb{N}$.
Output: a σ-basis $M \in \mathsf{K}[x]^{m \times m}$ with $\sigma = nd$.
Condition: $d = 0$ or $\log d \in \mathbb{N}$.

```
if d = 0 then M := I_m;
else if d = 1 then M := SigmaBasis(G,d);
else if d ≥ 2 then
   M' := BlockSigmaBasis(G, d/2);
   M'' := BlockSigmaBasis(x^{-d/2} M'G mod x^{d/2}, d/2);
   M := M''M';
fi;
return M;
```

Theorem 2.4 *Algorithm* `BlockSigmaBasis` *is correct. Its cost is* $1.5\mathsf{MM}'(m, d) + O(d\mathsf{MM}(m))$ *or* $\tilde{O}(m^\omega d)$ *operations in* K.

Proof. For correctness it suffices to show that the algorithm with input (G, d) uses only the first d coefficients of series G: when $d = 1$ this is true because of Algorithm `SigmaBasis`; if we assume this is true for a given $d/2$ then this is still true for d since $x^{-d/2} M' G \bmod x^{d/2}$ depends only on $G \bmod x^d$. Correctness then follows immediately from Lemma 2.3.

Now about complexity. First, it follows from Algorithm `SigmaBasis` that $\deg M \le d$. Hence the product $M'' M'$ costs $\mathsf{MM}(m, d/2)$. Second, since $\deg M' \le d/2$, the coefficient in x^i of $x^{-d/2} M' G \bmod x^{d/2}$ is the coefficient in $x^{i+d/2}$ of $M'(G \bmod x^d)$. This product costs $\mathsf{MM}(m, d)$. The cost $C(m, n, d)$ of Algorithm `BlockSigmaBasis` thus satisfies $C(m, n, 1) = O(\mathsf{MM}(m))$ and, for $d \ge 2$,

$$C(m, n, d) \le 2C(m, n, d/2) + \mathsf{MM}(m, d/2) + \mathsf{MM}(m, d).$$

This gives the bound $1.5\mathsf{MM}'(m, d) + O(d\mathsf{MM}(m))$. □

3 Column reduction

For $A \in \mathsf{K}[x]^{n\times n}$ we consider the problem of computing $C \in \mathsf{K}[x]^{n\times n}$ such that $C = AU$ is column reduced, U being a unimodular matrix over $\mathsf{K}[x]$. Column reduction is essentially lattice basis reduction for $\mathsf{K}[x]$-modules. To define the reduction, let d_j denote the degree of the jth column of C. The corresponding coefficient vector of x^{d_j} is the jth leading vector of C. We let $[C]_l$ be the matrix of these leading vectors.

Definition 3.1 *A matrix C is column reduced if its leading coefficient matrix satisfies* rank $[C]_l =$ rank C.

We refer to [14, 22] and the references therein for discussions on previous reduction algorithms and applications of the form especially in linear algebra and in linear control theory. If r is the rank of A, the best previously known cost for reducing A was $O(n^2rd^2)$ operations in K [14]. Thus in particular $O(n^3d^2)$ for a nonsingular matrix. Here we propose a different approach which takes advantage of fast polynomial matrix multiplication and gives in particular the complexity estimate $O\tilde{}(n^\omega d)$.

We assume that A of degree d is nonsingular in $\mathsf{K}[x]^{n\times n}$. The general case would require further developments. We compute a column reduced form of A by combining our techniques in [22] to the high-order lifting and the integrality certificate in [16, §9]. The main idea is to reduce the problem to the computation of a matrix Padé approximant whose side-effect is to normalize the involved matrices [22]. Let us first recall the definition of right matrix greatest common divisors.

Definition 3.2 *A right matrix gcd of $P \in \mathsf{K}[x]^{m\times n}$ and $A \in \mathsf{K}[x]^{n\times n}$ is any full row rank matrix G such that*

$$U \begin{bmatrix} P \\ A \end{bmatrix} = \begin{bmatrix} G \\ 0 \end{bmatrix}$$

with U unimodular.

Right gcd's are not unique, but if $[P^T \ A^T]^T$ has full column rank —here this is true by assumption— then, for given matrices P and A, all the gcd's are nonsingular and left equivalent (up to multiplication by a unimodular matrix on the left) in $\mathsf{K}[x]^{n\times n}$. This also leads to the notion of an irreducible matrix fraction description. (See for example [11] for a detailed study of matrix gcd's and fractions.)

Definition 3.3 *If a right gcd of P and A is unimodular then we say that P and A are relatively prime and that PA^{-1} is an irreducible right matrix fraction description.*

The whole algorithm for column reduction will be given in §3.3. The first step, detailed in §3.1, is to compute from A a strictly proper and irreducible right fraction description

$$H = RA^{-1} \in \mathsf{K}(x)^{n\times n}, \ R \in \mathsf{K}[x]^{n\times n}. \tag{6}$$

We recall that strictly proper means that H tends to zero when x tends to infinity. This implies that the degree of the jth column of R must be strictly lower than the degree of the jth column of A. Since the degrees of R and A are bounded by d, the second step of the method, studied in §3.2, is to compute from the first $2d+1$ terms of the expansion of H a right matrix Padé approximant

$$H = TC^{-1}$$

of H. Such an approximant, obtained from the results of §2 and [2], will have the additional property that C is column reduced. We will see that by the equivalence of irreducible fractions, C will be a column reduced form of A.

Like the algorithms in [16], our column reduction algorithm is randomized Las Vegas since the first step requires that $\det A(0) \neq 0$. Without loss of generality this may be assumed by choosing a random element x_0 in K and by computing a column reduced form C of $A(x + x_0)$. Indeed, a column reduced form of A is then recovered as $C(x - x_0)$.

3.1 A strictly proper and irreducible fraction

For a given A, its inverse A^{-1} may not be a strictly proper rational function, a case where $R = I$ is not a suitable choice in (6). We show that the integrality certificate of [16, §9] can be used here to find a target strictly proper function.

Lemma 3.4 *Let $A \in \mathsf{K}[x]^{n\times n}$ of degree d be such that $\det A(0) \neq 0$. For $h > (n-1)d$ define $R \in \mathsf{K}[x]^{m\times n}$ by*

$$I = \left(A^{-1} \bmod x^h\right) A + x^h R. \tag{7}$$

The fraction RA^{-1} is strictly proper and irreducible. If h is the closest power of 2 greater than $(n-1)d+1$, the $2d+1$ first terms of the expansion of RA^{-1} may be computed at the cost of $O(\mathsf{MM}(n,d)\log n) + \tilde{O}(n^2 d)$ operations in K.

Proof. Identity (7) is identity (12) in [16] with $B = I$ and $T = A$. This is a Euclidean matrix division with coefficients in reverse order. The fraction RA^{-1} is strictly proper because

$$RA^{-1} = x^{-h}A^{-1} - x^{-h}\left(A^{-1} \bmod x^h\right) \tag{8}$$

and $h > (n-1)d \geq \deg A^*$ where A^* is the adjoint matrix of A. On the other hand, there is a unimodular U such that

$$\begin{bmatrix} x^h R \\ A \end{bmatrix} = \begin{bmatrix} I & -(A^{-1} \bmod x^h) \\ 0 & I \end{bmatrix} \begin{bmatrix} I \\ A \end{bmatrix} = U \begin{bmatrix} I \\ 0 \end{bmatrix}.$$

Hence matrices $x^h R$ and A are relatively prime (see Definition 3.2) and the same is true for R and A.

For h as in the statement, the $2d+1$ terms of the expansion of RA^{-1} may be computed by high-order x^d-lifting [16, §8] with input parameters A, I, h and $2d+1$. The corresponding cost in [16, Proposition 13] is $\tilde{O}(n^\omega d)$. It can be seen from [16] that the algorithm actually runs at a cost of $O(\mathsf{MM}(n,d)\log n) + \tilde{O}(n^2 d)$ operations in K. □

3.2 Padé approximation and reduction

The key observation is that the descriptions TC^{-1} of H with C a column reduced form of A are those whose numerator and denominator matrices have minimal degrees (see Corollary 3.6 below). By definition they satisfy

$$\begin{bmatrix} H & -I \end{bmatrix} \begin{bmatrix} C \\ T \end{bmatrix} = 0 \bmod x^{2d+1}$$

and, as we shall see, their minimality implies that they must appear in any σ-basis of $G = [H \ -I]$ for $\sigma = n(2d+1)$. (Here we consider σ-bases with respect to the columns rather than the rows. Hence we transpose the matrices of section 2.)

To describe the set of all matrices T and C we use the notion of minimal basis of a module. For $M \in \mathsf{K}[x]^{n\times m}$, $m > n$, with rank n, let $N \in \mathsf{K}[x]^{m\times(m-n)}$ with columns forming a basis of the $\mathsf{K}[x]$-submodule $\ker M$. We denote by $d_1, d_2, \ldots, d_{m-n}$ the column degrees of N and assume they are ordered as $d_1 \leq d_2 \leq \cdots \leq d_{m-n}$. Then we have the following theorem and consequence.

Theorem 3.5 *[11, §6.5.4]. If N is column reduced then the column degrees $d'_1 \leq d'_2 \leq \cdots \leq d'_{m-n}$ of any other basis of $\ker M$ satisfy $d'_j \geq d_j$ for $1 \leq j \leq m-n$. We say that the columns of N form a minimal basis of $\ker M$.*

Corollary 3.6 *A basis $[C^T \ T^T]^T$ of $\ker G = \ker[H \ -I]$ is minimal if and only if C is a column reduced form of A.*

Proof. If $[C^T \ T^T]^T$ is a minimal basis then $H = TC^{-1}$ must be irreducible, otherwise the simplification of (T, C) by a right matrix gcd would lead to a basis with smaller degrees. The latter would contradicts Theorem 3.5. In addition since $[C^T \ T^T]^T$ is column reduced then C is column reduced. Indeed, H being strictly proper implies that T has column degrees strictly lower that those of C which thus dominate. By [11, Theorem 6.5-4] we further know that two irreducible descriptions TC^{-1} and RA^{-1} of the same function H have equivalent denominators. This means that there exists a unimodular U such that $C = AU$. Hence C is a column reduced form of A. Conversely, if C in a basis $[C^T \ T^T]^T$ is a column reduced form of A then by Theorem 6.5-4 cited above, TC^{-1} is an irreducible description of H. Since C is column reduced, the non-minimality of $[C^T \ T^T]^T$ as a basis of $\ker G$ would then contradict its irreducibility. □

We now show that, for σ large enough, a σ-basis with respect to the columns of $[H \ -I]$ leads to a minimal basis $[C^T \ T^T]^T$ as in the corollary, and hence to a column reduced form of A. We follow here the techniques in [10] for computing a minimal basis of the kernel of a polynomial matrix.

Lemma 3.7 *Let $N \in \mathsf{K}[x]^{2n \times 2n}$ be a σ-basis with respect to the columns of $G = [H \ -I]$. If $\sigma \geq n(2d+1)$, then the n columns of N of degree at most d define an irreducible description TC^{-1} of H with C a column reduced form of A.*

Proof. We first show that there may be at most one set of n columns of N of degree at most d. Then the minimality of the σ-basis will imply its existence and the fact that it leads to a fraction description of the form TC^{-1}.

If $[Q^T \ P^T]^T$ is a set of n columns of N of degrees bounded by d then

$$HQ - P \equiv 0 \bmod x^{2d+1}.$$

If $A^{-1}S$ is a left description of H, defined in the same way as RA^{-1} in Lemma 3.4, we get

$$SQ - AP \equiv 0 \bmod x^{2d+1}.$$

Since every matrix in the latter identity has degree at most d we deduce that

$$SQ - AP = 0. \tag{9}$$

It follows from the columns of a σ-basis N being linearly independent over $\mathsf{K}(x)$ [2] that $[Q^T \ P^T]^T$ has full column rank. Since (9) implies that

$$\begin{bmatrix} I & 0 \\ S & -A \end{bmatrix} \begin{bmatrix} Q \\ P \end{bmatrix} = \begin{bmatrix} Q \\ 0 \end{bmatrix},$$

we see that Q is invertible and satisfies

$$PQ^{-1} = A^{-1}S = H. \tag{10}$$

Another choice $[Q_1^T \ P_1^T]^T$ of n such columns would give $H = PQ^{-1} = P_1 Q_1^{-1}$. By [11, Theorem 6.5-4] the two descriptions would verify

$$\begin{bmatrix} Q \\ P \end{bmatrix} P_2 = \begin{bmatrix} Q_1 \\ P_1 \end{bmatrix}$$

and this would contradict the nonsingularity of the σ-basis. Hence the choice $[Q^T\ P^T]^T$ must be unique as announced.

Let $d_1, \ldots, d_n$ be the minimal degrees given by the columns of a minimal description $[C^T\ T^T]^T$ and let $v_1, \ldots, v_n$ be the corresponding columns. From II) in Definition 2.1, v_1 can be written as

$$v_1 = \sum_{j=1}^{2n} c_1^{(j)} N_j, \text{ with } \deg c_1^{(j)} + \deg N_j \leq d_1$$

where N_j is the jth column of the σ-basis N. Thus one column of N has degree bounded by d_1. Now assume that N has $k-1$ columns of degrees $d_1, \ldots, d_{k-1}$ with v_k not belonging to the corresponding submodule. As for $k = 1$, there exists a column of N, linearly independent with respect to the first $k-1$ chosen ones, of degree bounded by d_k. Therefore N contains n distinct columns of degrees bounded by $d_1, \ldots d_n$ and, by (10), in the kernel of $[H\ -I]$. Lemma 3.7 shows in conclusion that these n columns give C, a column reduced form of A, in their first n rows. □

We may notice that the result of the lemma would be true as soon as $\sigma > 2nd$ for the computation of an approximant of type $(d-1, d)$ as defined in [2].

3.3 Cost of the reduction

Our column reduction algorithm can be stated as follows.

Algorithm `ColumnReduction`(A)
Input: $A \in \mathsf{K}[x]^{n\times n}$ of degree d.
Output: $C = AU$ a column reduced form of A.
Condition: A is nonsingular.

Choice of a random x_0 in K;
if $\det A(x_0) = 0$ then **fail**; /* A is probably singular */
$B := A(x + x_0)$;

$h := (n-1)d + 1$;
$H := \left(B^{-1} - (B^{-1} \bmod x^h)\right)/x^h \bmod x^{2d+1}$;
$TC^{-1} :=$ a Padé approximant of H mod x^{2d+1};
return $C(x - x_0)$;

Its complexity follows from Lemma 3.4 concerning the computation of the first terms of H, and from Theorem 2.4 concerning the computation of the $n(2d+1)$-basis of Lemma 3.7.

Theorem 3.8 *A column reduced form of a nonsingular matrix A of degree d in $\mathsf{K}[x]^{n\times n}$ can be computed by a Las Vegas (certified) algorithm in $O(\mathsf{MM}'(n,d) + \mathsf{MM}(n,d)\log n)$ $+\tilde{O}(n^2 d)$ or $\tilde{O}(n^\omega d)$ operations in K.*

4 Matrix product & determinant

The link between matrix multiplication and determinant computation over a the field K is well known. We may refer to [5, Chap.16] for a survey of the question. If we have an algorithm for multiplying to matrices with $\mathsf{MM}(n)$ operations in K then we have an algorithm (algebraic RAM) for computing the determinant with $O(\mathsf{MM}(n))$ operations in K [4]. Conversely, the exponents (computation trees) of matrix multiplication and of determinant computation coincide [18, 1]. Furthermore, if we have a randomized Monte Carlo algorithm which computes the determinant

with $D(n)$ operations in K then we have a Monte Carlo algorithm for multiplying two matrices with $O(D(n))$ operations in K [8, Theorem 1.3].

In this section we show that similar results hold for polynomial matrices of degree d. In §4.1, using a slight extension of Baur & Strassen's idea [1, Cor. 5], we propose a reduction of polynomial matrix multiplication to determinant computation. Then in §4.2, based on the techniques in [16, 21, 10], we investigate the reverse reduction.

We use two models of computation, algebraic straight-line programs or algorithms on an algebraic RAM.

4.1 Polynomial matrix multiplication

Baur & Strassen [1, Cor. 5] in conjunction with [18, 4] have shown that a straight-line program of length $D(n)$ for computing the determinant of a matrix A in $\mathsf{K}^{n\times n}$ can be transformed into a program of length bounded by $O(D(n))$ for matrix multiplication. Indeed, the problem of multiplying two matrices can be reduced to matrix inversion [18, 4]. Then matrix inversion is reduced to the problem of computing the determinant by differentiation of the program of length $D(n)$ [12, 1].

The complexity estimate $O(D(n))$ for matrix multiplication relies on the computation of the partial derivatives of the determinant as a function in $\mathsf{K}[a_{1,1},\ldots,a_{i,j},\ldots,a_{n,n}]$. The $a_{i,j}$'s are indeterminates standing for the entries of the input matrix. It was not clear how to extend the result to polynomial matrices. The output of a program of length $D(n,d)$ over K which computes the determinant of a polynomial matrix is a function in $\mathsf{K}[x,a_{1,1},\ldots,a_{i,j},\ldots,a_{n,n}]$, that is, a set of functions in $\mathsf{K}[a_{1,1},\ldots,a_{i,j},\ldots,a_{n,n}]$. A straightforward idea could be to differentiate at least d such functions, but it is not known how to do it without increasing the complexity estimate $O(D(n,d))$.

Here we remark that having only one particular coefficient of the polynomial matrix determinant is sufficient for recovering the first $d+1$ coefficients of the polynomial entries of the adjoint matrix $A^* = (\det A)A^{-1} \in \mathsf{K}[x]^{n\times n}$. Hence we first compute A^* modulo x^{d+1} and from there, the multiplication of two matrices of degree d is easily deduced.

Let $A \in \mathsf{K}[x]^{n\times n}$ have degree d and denote its (i,j) entry by $a_{i,j} = \sum_{k=0}^{d} a_{i,j,k}x^k$. Let further $a^*_{i,j} = \sum_{k=0}^{nd-d} a^*_{i,j,k}x^k$ be the (i,j) entry of the adjoint matrix A^* of A and let $\Delta = \sum_{l=0}^{nd} \Delta_l x^l$ be the determinant of A. We have the following relation between the partial derivatives of coefficient Δ_l and some of the $a^*_{i,j,k}$'s.

Lemma 4.1 *The partial derivatives of the coefficients of the determinant and the coefficients of the adjoint matrix satisfy*

$$a^*_{j,i,l-k} = \frac{\partial \Delta_l}{\partial a_{i,j,k}},\ 0 \le l \le nd,\ 0 \le k \le d.$$

where, by convention, $a^*_{j,i,k} = 0$ *if* $k < 0$ *or* $k > nd - d$.

Proof. By Cramer's rule and since $\partial a_{i,j}/\partial a_{i,j,k} = x^k$, we have $\partial\Delta/\partial a_{i,j,k} = x^k a^*_{j,i}$. On the other hand, for $1 \le k \le d$ the coefficients $\Delta_0,\ldots,\Delta_{k-1}$ do not depend on variable $a_{i,j,k}$ and thus $\partial\Delta/\partial a_{i,j,k} = \sum_{l=k}^{nd} \partial\Delta_l/\partial a_{i,j,k}x^l$. Therefore

$$\sum_{l=0}^{nd-d} a^*_{j,i,l}x^{k+l} = \sum_{l=0}^{nd-k} \frac{\partial \Delta_{k+l}}{\partial a_{i,j,k}}x^{k+l}$$

and the result follows by identifying the coefficients. □

The theorem below is given for a programs over K which compute the particular coefficient Δ_d. It thus remains valid for programs over K which compute the whole determinant in $\mathsf{K}[x]$.

Theorem 4.2 *If there is a straight-line program of length $D(n,d)$ over K which computes the $(d+1)$st coefficient of the determinant of an $n \times n$ matrix of degree d, then there is a straight-line program of length no more than $8D(n,d)$ which multiplies two $n \times n$ matrices of degree d.*

Proof. It follows from Lemma 4.1 with $l = d$ that the first $d+1$ coefficients of $a^*_{j,i}$ are given by

$$a^*_{j,i,d-k} = \frac{\partial \Delta_d}{\partial a_{i,j,k}}, \quad 0 \leq k \leq d.$$

By computing the partial derivatives [12, 1] of the given program for the determinant coefficient Δ_d we thus have a program of length bounded by $4D(n,d)$ for computing $A^* \bmod x^{d+1}$. We conclude by applying this result twice to the well known $3n \times 3n$ matrix

$$A = \begin{bmatrix} I_n & A_1 & \\ & I_n & A_2 \\ & & I_n \end{bmatrix} \text{ with } A_1, A_2 \in \mathsf{K}[x]^{n \times n} \text{ of degree } d.$$

The associated adjoint matrix is the matrix of degree $2d$

$$A^* = \begin{bmatrix} I_n & -A_1 & A_1A_2 \\ & I_n & -A_2 \\ & & I_n \end{bmatrix}.$$

One can thus recover $A_1A_2 \bmod x^{d+1}$ from $A^* \bmod x^{d+1}$. To get higher order terms, notice that if $A_1A_2 = x^dH + L$ then $\overline{H} = \overline{A}_1\overline{A}_2 \bmod x^{d+1}$ where $\overline{M} = \sum_{i=0}^{d} M_{d-i}x^i$ is the "mirror" polynomial matrix of $M = \sum_{i=0}^{d} M_i x^i$. Therefore $\overline{H}$ and thus H can be recovered from $\overline{A}^* \bmod x^{d+1}$. □

Following Giesbrecht [8, Theorem 1.3] we may state an analogous result for algorithms on an algebraic RAM: if we have a randomized Monte Carlo algorithm which computes Δ_d with $D(n,d)$ operations in K then we have a Monte Carlo algorithm for multiplying two matrices of degree d with $O(D(n,d))$ operations in K.

4.2 Polynomial matrix determinant

Over K, algorithms for reducing determinant computation to matrix multiplication work recursively in $O(\log n)$ steps. Roughly, step i involves $n/2^i$ products of $2^i \times 2^i$ matrices. (See for example [17, 4].) When looking for the determinant of a polynomial matrix, both Storjohann's algorithm [16] and the straight-line program we derive below from our previous studies in [21, 10] also work in $O(\log n)$ steps. They involve polynomial matrices of dimensions $2^i \times 2^i$ and degree $nd/2^i$ (this accounts for the definition of function $\mathsf{MM}''(n,d)$ in introduction).

In this section we study the costs of two different methods for computing the determinant of an $n \times n$ polynomial matrix of degree d. These costs are functions of $\mathsf{MM}''(n,d)$ and $\mathsf{MM}'(n,d)$ and reduce both to $O^{\sim}(n^\omega d)$ when taking $\mathsf{MM}(n,d) = \Theta(n^\omega d \log d \log\log d)$.

The first method is Storjohann's high order lifting on an algebraic RAM [16]. We recall it briefly in §4.2.1 below for the sake of completeness. We then present in §4.2.2 an alternative approach for straight-line programs, which has been developed independently [21, 10].

4.2.1 Lifting determinant algorithms

Storjohann has given in [16, Proposition 24] a Las Vegas algorithm for computing the determinant of a polynomial matrix in $O^{\sim}(n^{\omega}d)$ operations. Without going into details since there is no change of the method, we point out that Storjohann's result actually gives the following.

Theorem 4.3 *[16] The determinant of an $n \times n$ polynomial matrix of degree d can be computed by a Las Vegas algorithm in $O(\mathsf{MM}''(n,d) + \mathsf{MM}(n,d)\log^2 n) + O^{\sim}(n^2 d)$ or $O^{\sim}(n^{\omega}d)$ operations in K.*

This complexity estimate may be deduced from the lines of [16]. The term in $O(\mathsf{MM}''(n,d))$ comes from the integrality certificate and the Smith form computations of [16, Propositions 17 & 22]. The term in $O(\mathsf{MM}(n,d)\log^2 n)$ comes from the high-order lifting of [16, Prop. 13] performed at each step of the $O(\log n)$ steps of the main iteration [16, §13].

4.2.2 Straight-line determinant

Given $A \in \mathsf{K}[x]^{n\times n}$ of degree d and sufficiently generic, the straight-line approach presented in [21, 10] computes the inverse of A as $A^{-1} = B^{-1}U$ where $U \in \mathsf{K}[x]^{n\times n}$ and $B \in \mathsf{K}[x]^{n\times n}$ is diagonal of degree nd. One can further recover the determinant of A from B alone as we explain now. By definition of the inverse, $U = (\det A)^{-1}BA^*$ where A^* is the adjoint matrix of A. Generically, $\deg\det A = \deg B = nd$ and $\det A$ is coprime with each entry $a^*_{i,j}$ of A^*. It follows that the diagonal entries $b_{i,i}$ of B are nonzero constant multiples of $\det A$. Since $\det A(0)$ is generically nonzero, the determinant of a generic A is thus equal to $(\det A(0))b_{i,i}/b_{i,i}(0)$ for $1 \le i \le n$.

To compute B in Algorithm `Determinant` below, we proceed as for Algorithm `Inverse` in [10]: we diagonalize the input matrix in $\log n$ steps, starting with

$$A \to UA = \begin{bmatrix} \overline{U} \\ \underline{U} \end{bmatrix} \begin{bmatrix} A_L & A_R \end{bmatrix} = \begin{bmatrix} \overline{U}A_L & \\ & \underline{U}A_R \end{bmatrix} \tag{11}$$

where $A_L, A_R \in \mathsf{K}[x]^{n\times n/2}$ and where $\overline{U}, \underline{U} \in \mathsf{K}[x]^{n/2\times n}$ are minimal bases of the left kernels of A_R, A_L respectively. These minimal bases are as in Theorem 3.5, for left kernels.

Algorithm `Determinant`(A)
Input: $A \in \mathsf{K}[x]^{n\times n}$ of degree d.
Output: $\det A$.
Condition: $\det A(0) \neq 0$, $\gcd(a^*_{i,j}, \det A) = 1$, $\log n \in \mathbb{N}$.

$B :=$ copy(A);
for i from 1 to $\log n$ do
  /* B is block-diagonal with 2^{i-1} blocks $B_i^{(j)}$ */
  for j from 1 to 2^{i-1} do
    $\underline{U}_i^{(j)} :=$ a minimal basis of $\ker B_{i,L}^{(j)}$;
    $\overline{U}_i^{(j)} :=$ a minimal basis of $\ker B_{i,R}^{(j)}$;
  od;
  $U_i := \mathrm{diag}\left(\begin{bmatrix} \overline{U}_i^{(1)} \\ \underline{U}_i^{(1)} \end{bmatrix}, \ldots, \begin{bmatrix} \overline{U}_i^{(2^{i-1})} \\ \underline{U}_i^{(2^{i-1})} \end{bmatrix}\right)$;
  $B := U_i B$;
od;
return $(\det A(0))b_{1,1}/b_{1,1}(0)$;

The analysis of this algorithm is similar to the one of the inversion algorithm in [10] and we simply recall the key point for complexity: although the minimal bases in (11) can have degrees as large as nd, they generically have degrees equal to d – a property which carries over the next step – and can then be recovered from the rows of any σ-bases of A_L, A_R with $\sigma \geq n(2d+1)$ [10, Properties 1 & 2]. Generically, matrices $B_{i,L}^{(j)}$ and $B_{i,R}^{(j)}$ thus have dimensions $2^{1-i}n \times 2^{-i}n$ and degree $2^{i-1}d$; it then follows from Theorem 2.4 that minimal kernel bases $\underline{U}_i^{(j)}$ and $\overline{U}_i^{(j)}$ can be computed in $O(\mathsf{MM}'(2^{-i}n, 2^i d))$ operations in K. On the other hand, the cost of matrix update $B := U_i B$ is $O(2^i \mathsf{MM}(2^{p-i}, 2^i d))$ where $\mathsf{MM}(n,d) \leq \mathsf{MM}'(n,d)$. Hence the result below, recalling that the cost of computing $\det A(0)$ is bounded by $O(\mathsf{MM}(n))$.

Theorem 4.4 *The determinant of an $n \times n$ polynomial matrix of degree d can be computed by a straight-line program over K of length $O(\sum_{i=1}^{\log n} 2^i \mathsf{MM}'(2^{-i}n, 2^i d))$ or $O\tilde{}(n^\omega d)$.*

5 Conclusion

In this paper we reduced polynomial matrix multiplication to determinant computation and conversely, under the straight-line model. Under the algebraic RAM model, we reduced the tasks of computing a σ-basis and column reduced form to the one of multiplying square polynomial matrices; as we have seen, similar reductions follow from [16] for the problems of computing the determinant and the Smith normal form.

However, in $\mathsf{K}[x]^{n\times n}$ it is still unclear whether
- Hermite and Frobenius normal forms,
- associated transformation matrices (even for the column reduced form),
- the characteristic polynomial,

can be computed in $O\tilde{}(\mathsf{MM}(n,d))$ or $O\tilde{}(n^\omega d)$ operations in K as well. Another related question is to know whether the straight-line approach of section 4.2.2 yields a $O\tilde{}(n^3 d)$ algorithm for computing the inverse of a polynomial matrix.

References

[1] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[2] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 15(3):804–823, 1994.

[3] D. Bini and V.Y. Pan. *Polynomial and Matrix Computations, Vol 1: Fundamental Algorithms.* Birkhauser, Boston, 1994.

[4] J. Bunch and J. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.*, 28:231–236, 1974.

[5] P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1997.

[6] D.G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.

[7] D. Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Mathematics of Computation*, 62(205):333–350, 1994.

[8] M. Giesbrecht. *Nearly optimal algorithms for canonical matrix forms*. PhD thesis, Department of Computer Science, University of Toronto, 1993.

[9] O.H. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3:45–56, 1982.

[10] C.-P. Jeannerod and G. Villard. Straight-line computation of the polynomial matrix inverse. Research Report 2002-47, Laboratoire LIP, ENS Lyon, France, 2002. `http://www.ens-lyon.fr/LIP/Pub/rr2002.html`.

[11] T. Kailath. *Linear systems*. Prentice Hall, 1980.

[12] S. Linnainmaa. Taylor expansion of the accumulated rounding errors. *BIT*, 16:146–160, 1976.

[13] A. Lobo. *Matrix-free linear system solving and applications to symbolic computation*. PhD thesis, Dept. Comp. Sc., Rensselaer Polytech. Instit., Troy, New York, Dec. 1995.

[14] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*. To appear.

[15] A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, Institut für Wissenschaftliches Rechnen, ETH-Zentrum, Zurich, Switzerland, November 2000.

[16] A. Storjohann. High-Order Lifting (Extended Abstract). In *International Symposium on Symbolic and Algebraic Computation, Lille, France*, pages 246–254. ACM Press, July 2002.

[17] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.

[18] V. Strassen. Vermeidung von Divisionen. *J. Reine Angew. Math.*, 264:182–202, 1973.

[19] E. Thomé. Fast computation of linear generators for matrix sequences and application to the block Wiedemann algorithm. In *International Symposium on Symbolic and Algebraic Computation, London, Ontario*, pages 323–331. ACM Press, July 2001.

[20] W.J. Turner. *Black box linear algebra with the LinBox library*. PhD thesis, North Carolina State University, Raleigh, NC USA, May 2002.

[21] G. Villard. Computation of the inverse and determinant of a matrix (summary by E. Thomé). *Algorithms Seminar, may 2002*, F. Chyzak (ed.), INRIA Rocquencourt, France, 2003.

[22] G. Villard. Computing Popov and Hermite forms of polynomial matrices. In *International Symposium on Symbolic and Algebraic Computation, Zurich, Suisse*, pages 250–258. ACM Press, July 1996.

[23] G. Villard. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In *International Symposium on Symbolic and Algebraic Computation, Maui, Hawaii, USA*, pages 32–39. ACM Press, July 1997.

A Computed LSP factorization

In Section 2 we assumed we use the LSP algorithm of [9] as described in [3, p. 103]. In general, LSP factorization is not unique: for example

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \\ * & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} & 1 \\ 1 & \end{bmatrix}. \tag{12}$$

However, with this choice of algorithm, some columns of L are prescribed as follows.

Property A.1 *If the ith row of S is identically zero then the ith column of L is equal to the ith unit vector.*

In (12) the computed factor L is thus the one with $* = 0$.

Proof. We use the notation of the solution to Problem 2.7c in [3, p. 103] and we assume without loss of generality that m is a power of two. The result is clear for $m = 1$ since we take $L = P = \begin{bmatrix} 1 \end{bmatrix}$. For $m \geq 2$, assume that the result holds for $m/2$ and recall that in [3, p. 103] factors L, S are computed as

$$L = \begin{bmatrix} L_1 & \\ G & L_2 \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} S_1' & BP_2^{-1} \\ & S_2 \end{bmatrix} \tag{13}$$

where L_1, $S_1 = [S_1', B]$ and L_2, S_2, P_2 stem from LSP factorizations of respective dimensions $m/2 \times m$ and $m/2 \times (m - r)$ where $r \leq m/2$ is the rank of S_1. (We shall describe G later.) We consider two cases, depending on whether $i > m/2$ or $i \leq m/2$. If $i > m/2$ then the $(i - m/2)$th row of S_2 is zero. By assumption, the $(i-m/2)$th column of L_2 is therefore equal to the $(i-m/2)$th unity vector and the result follows from the shape of L in (13). If $i \leq m/2$ then the ith row of S_1 is zero, for $S_1 = [S_1', B]$ and P_2 is a permutation matrix. Hence, by assumption, the ith column of L_1 is the ith unity vector. To prove that the ith column of G is zero, recall first how G is defined in [3, p. 103]: $G = FS_1^{(-1)}$ where F is a matrix of order $m/2$ whose last $m/2 - r$ columns are zero; additionally $S_1^{(-1)}$ is a matrix of order $m/2$ that transforms S_1 into

$$S_1^{(-1)} S_1 = \begin{bmatrix} I_r & * \\ O & O \end{bmatrix} \in \mathsf{K}^{m/2 \times m}.$$

In particular $S_1^{(-1)}$ contains a row permutation such that the ith row of S_1 corresponds to a row of index greater than r in $S_1^{(-1)} S_1$. The same permutation acts on the columns of F through transformation $G = FS_1^{(-1)}$ and the ith column of G is therefore set to zero by definition of F. □