A deterministic algorithm for inverting a polynomial matrix

Wei Zhou, George Labahn, Arne Storjohann

David R. Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1

Abstract

Improved cost estimates are given for the problem of computing the inverse of an $n \times n$ matrix of univariate polynomials over a field. A deterministic algorithm is demonstrated that has worst case complexity $(n^3 s)^{1+o(1)}$ field operations, where $s \geq 1$ is an upper bound for the average column degree of the input matrix. Here, the "$+o(1)$" in the exponent indicates a missing factor $c_1(\log ns)^{c_2}$ for positive real constants c_1 and c_2. As an application we show how to compute the largest invariant factor of the input matrix in $(n^\omega s)^{1+o(1)}$ field operations, where ω is the exponent of matrix multiplication.

1. Introduction

We consider the problem of computing the inverse of a matrix of polynomials over an abstract field $\mathbb{K}$. Let $\mathbf{F} \in \mathbb{K}[x]^{n \times n}$ be an $n \times n$ matrix over the ring of univariate polynomials $\mathbb{K}[x]$, and let $d \geq 1$ be a bound on the degrees of entries of $\mathbf{F}$. Recall that the determinant $\det \mathbf{F}$ can have degree up to nd and that the adjugate (or classical adjoint) $\det \mathbf{F} \cdot \mathbf{F}^{-1}$ is a polynomial matrix with entries of degree up to nd. Thus, $\mathbf{F}^{-1}$ can require on the order of $n^3 d$ field elements to represent: a factor of n more than required to write down $\mathbf{F}$.

In a surprising result, Jeannerod and Villard [4] give an algorithm to compute $\mathbf{F}^{-1}$ for a generic input matrix of dimension a power of 2 that has a cost of $(n^3 d)^{1+o(1)}$ field operations from $\mathbb{K}$. Here, and in the remainder of the paper, the "$+o(1)$" in the exponent of cost estimates indicates a missing factor $c_1(\log nd)^{c_2}$ for positive real constants c_1 and c_2. The inversion algorithm of Jeannerod and Villard [4] works for arbitrary input matrices. However, the $(n^3 d)^{1+o(1)}$ running time bound is obtained only for inputs that have dimension a power of 2, and which satisfy the genericity requirement that the $n^2(d+1)$ coefficients of $\mathbf{F}$ do not cause a particular polynomial of degree $n^2(d+1)$ to vanish. The genericity requirement ensures that all matrices arising during the construction

Email address: `{w2zhou,glabahn,astorjoh}@uwaterloo.ca` (Wei Zhou, George Labahn, Arne Storjohann)

have uniform row and column degrees. Jeannerod and Villard's recipe is the first essentially optimal inversion algorithm for polynomial matrices, at least for generic matrices with dimension a power of 2, improving on the previously known algorithms which have cost $(n^{\omega+1}d)^{1+o(1)}$, where ω is the exponent of matrix multiplication. More recently, an alternative inversion algorithm is given by Storjohann [9]: the algorithm is Las Vegas randomized and has expected cost $(n^3d)^{1+o(1)}$ field operations for all input matrices. For a survey of previous work on polynomial matrix inversion we refer to [4, 9].

In this paper we extend the algorithm of Jeannerod and Villard [4] to work for arbitrary input matrices while maintaining a worst case deterministic $(n^3d)^{1+o(1)}$ bound on the running time in all cases. We illustrate the differences between the algorithm of [4] and our extension using a pair of simple examples.

To understand the behaviour of the inversion algorithm [4] for generic inputs it will suffice to consider a 4×4 input matrix of degree 3. In our examples we only show the degree profile of the matrices, that is, the degrees of the polynomials inside the matrix and not the polynomials themselves. Blocks of the matrix that are necessarily zero are left blank. The algorithm begins by computing a matrix $\mathbf{A}_1$ such that

$$\overset{\text{degs } \mathbf{F}}{\left[\begin{array}{cccc} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ \hline 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{array}\right]} \overset{\text{degs } \mathbf{A}_1}{\left[\begin{array}{cc|cc} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{array}\right]} = \left[\begin{array}{cc|cc} 6 & 6 & & \\ 6 & 6 & & \\ \hline & & 6 & 6 \\ & & 6 & 6 \end{array}\right].$$

The first 2 columns of $\mathbf{A}_1$ comprise a kernel basis for the last 2 rows of $\mathbf{F}$ while the last 2 columns of $\mathbf{A}_1$ comprise a kernel basis for the first 2 rows of $\mathbf{F}$. The algorithm now proceeds recursively on the two 2×2 diagonal blocks of $\mathbf{F}\cdot\mathbf{A}_1$, continuing until the matrix is diagonalized. For this example two levels of recursion suffices to obtain a diagonalization $\mathbf{B}$ of the input matrix.

$$\mathbf{F}\overset{\text{degs } \mathbf{A}_1}{\left[\begin{array}{cccc} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{array}\right]} \overset{\text{degs } \mathbf{A}_2}{\left[\begin{array}{cc|cc} 6 & 6 & & \\ 6 & 6 & & \\ \hline & & 6 & 6 \\ & & 6 & 6 \end{array}\right]} = \overset{\text{degs } \mathbf{B}}{\left[\begin{array}{cccc} 12 & & & \\ & 12 & & \\ & & 12 & \\ & & & 12 \end{array}\right]}. \tag{1}$$

By multiplying (1) on the left by $\mathbf{F}^{-1}$ and on the right by $\mathbf{B}^{-1}$ a structured decomposition is obtained for $\mathbf{F}^{-1}$. The genericity condition required for the cost analysis in [4] ensures that the property "dimension $\times$ degree $= nd$" holds for all the recursive subproblems. In general, for a generic input matrix $\mathbf{F}$ of degree d, and dimension n a power of 2, the decomposition has the form

$$\mathbf{F}^{-1} = \mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_{\log n}\cdot\mathbf{B}^{-1}, \tag{2}$$

with $\mathbf{B} = (\det \mathbf{F})\cdot\mathbf{I}_n$ and $\mathbf{A}_{i+1}$ block diagonal with blocks of dimension $n/2^i$ and degree 2^id, $0 \le i \le \log n - 1$. Thus, if $T(n,d)$ denotes the running time of the method to compute the structured decomposition on the right hand side of (2), then

$$T(n,d) \le 2\,T(n/2, 2d) + (n^{\omega} d)^{1+o(1)}, \tag{3}$$

and it follows that $T(n,d) \in (n^{\omega}d)^{1+o(1)}$. Note that each of the output matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{\log n}, \mathbf{B}$ in the inverse decomposition requires at most $n^2(d+1)$ field elements to represent, so the total size of the output is $O(n^2 d \log n)$ field elements. In [4] it is also shown that that multiplying together the decomposition to obtain $\mathbf{F}^{-1}$ explicitly can be done in time $(n^3 d)^{1+o(1)}$.

For a non-generic input matrix the degrees of columns in the kernel basis in the $\mathbf{A}_i$ matrices need not be uniform. Even a so-called minimal kernel basis, for which the sum of the column degrees is minimal, can have skew column degrees. As a result, the recurrence (3) for the running time, which is based on the single degree bound d, no longer applies. To overcome the difficulty of non-uniform column degrees we capture more precisely the size of the input matrix to the kernel computations using a tuple $[d_1, d_2, \ldots, d_n]$ such that column i of the input is bounded in degree by d_i. We also make use of *shifted* column degrees to guide the computations and to prove size bounds for the intermediate matrices arising during the compuations. The notion of shifted column degree and the key subroutines we use are recalled with examples in Section 2. Incorporating shifted column degrees allows us to apply the new algorithms from Zhou et al. [13] (see also [11]) for shifted kernel basis computations and skew degree matrix multiplications. Instead of analyzing algorithms in terms of an upper bound $d \ge 1$ for the matrix degree, we can analyse in terms of a bound $s \ge 1$ for the *average* column degree. We show that the inversion algorithm for generic input matrices [4] can be adapted to a non-generic input matrix in such a way as to support a similar recurrence for the running time as given by (3) but with d replaced by s.

To illustrate the inversion algorithm with the computations guided by shifted degrees, consider a 5×5 input matrix $\mathbf{F}$. (This particular example is given explicitly in Section 3.) The degrees of entries of $\mathbf{F}$ are

$$\operatorname{degs} \mathbf{F} = \begin{bmatrix} 1 & 3 & 4 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 2 & 2 & 4 & 1 \\ \cdot & 2 & 4 & \cdot & \cdot \\ 0 & 2 & 3 & 1 & \cdot \end{bmatrix}$$

with a $\cdot$ indicating a zero polynomial. For this input the algorithm computes the following decomposition.

$$\mathbf{F} \overset{\operatorname{degs} \mathbf{A}_1}{\begin{bmatrix} 1 & 4 & \cdot & 0 & 1 \\ \cdot & 2 & \cdot & 2 & \cdot \\ \cdot & 0 & \cdot & 1 & \cdot \\ 0 & 1 & \cdot & 0 & \cdot \\ \cdot & \cdot & 0 & \cdot & 0 \end{bmatrix}} \overset{\operatorname{degs} \mathbf{A}_2}{\left[\begin{array}{ccc|cc} 2 & 3 & 2 & & \\ 1 & 4 & 1 & & \\ 4 & 5 & 3 & & \\ \hline & & & 5 & \cdot \\ & & & 1 & 1 \end{array}\right]} \overset{\operatorname{degs} \mathbf{A}_3}{\left[\begin{array}{cc|c|cc} 4 & 0 & & & \\ 3 & 2 & & & \\ \cline{1-3} & & 0 & & \\ \hline & & & 0 & \\ & & & & 0 \end{array}\right]} = \overset{\operatorname{degs} \mathbf{B}}{\begin{bmatrix} 9 & & & & \\ & 7 & & & \\ & & 3 & & \\ & & & 5 & \\ & & & & 1 \end{bmatrix}}$$

In general, the modified inversion algorithm for non-generic inputs returns a list of matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{\lceil \log n \rceil}, \mathbf{B}$ satisfying

$$\mathbf{F}^{-1} = \mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_{\lceil \log n \rceil} \cdot \mathbf{B}^{-1}. \tag{4}$$

The $\mathbf{A}_i$ matrices will be block diagonal, with $\mathbf{A}_2$ having blocks of dimension bounded by $\lceil n/2 \rceil$, $\mathbf{A}_3$ having blocks of dimension bounded by $\lceil \lceil n/2 \rceil /2 \rceil$, etc. The algorithm produces the output matrices in (4) in time $(n^\omega s)^{1+o(1)}$. The structured decomposition still requires only $O(n^2 s \log n)$ space. We also show that the explicit inverse can be computed as the product of the matrices on the right hand side of (4) in time $(n^3 s)^{1+o(1)}$.

As an application of the inversion algorithm we show that the largest invariant factor of the input matrix $\mathbf{F}$ can be recovered from the diagonalization $\mathbf{B}$ in (4). Recall that the largest invariant factor of $\mathbf{F}$ is the minimal degree monic polynomial p such that $p \cdot \mathbf{F}^{-1}$ has only polynomial entries. We show that p is equal to the least common multiple of the diagonal entries of $\mathbf{B}$ and thus our work establishes that p can be computed deterministically in $(n^\omega s)^{1+o(1)}$ field operations. The previously fastest algorithm [8] for largest invariant factor is randomized and computes p in expected time $(n^\omega d)^{1+o(1)}$. We also show how to compute the sequence of matrix powers $V^2, V^3, \ldots, V^n$ of an arbitrary $V \in \mathbb{K}^{n \times n}$ in time $(n^3)^{1+o(1)}$.

The remainder of this paper is as follows. Section 2 includes some background on the tools required for the inversion computation described in Section 3. This is followed in Section 4 by the theorems giving the new complexity results for multiplying together the $\mathbf{A}_i$ matrices. Section 5 gives some applicaitons of the fast inversion algorithm and Section 6 concludes.

2. Preliminaries

In this section we give the basic cost model and definitions and properties of *shifted degree* and *kernel basis*, which are essential for our computation.

2.1. Cost model

Algorithms are analyzed by bounding the number of arithmetic operations in the coefficient field $\mathbb{K}$ on an algebraic random access machine. We will frequently use the fact that the cost of multiplying two polynomial matrices with dimension n and degree bounded by $d \geq 1$ is $(n^\omega d)^{1+o(1)}$ field operations from $\mathbb{K}$, where ω is the exponent of matrix multiplication. We refer to the book by von zur Gathen and Gerhard [10] for more details and references about polynomial and matrix multiplication.

2.2. Shifted degrees

Our methods depend extensively on the concept of *shifted* degrees of polynomial matrices [1]. For a column vector $\mathbf{p} = [p_1, \ldots, p_n]^T$ of univariate polynomials over a field $\mathbb{K}$, its column degree, denoted by $\operatorname{cdeg} \mathbf{p}$, is the maximum of the degrees of the entries of $\mathbf{p}$, that is,

$$\operatorname{cdeg} \mathbf{p} = \max_{1 \le i \le n} \deg p_i .$$

The *shifted column degree* generalizes this standard column degree by taking the maximum after adjusting the degrees by a given integer vector that is known as a *shift*. More specifically, the shifted column degree of $\mathbf{p}$ with respect to a shift $\vec{s} = [s_1, \ldots, s_n] \in \mathbb{Z}^n$, or the *$\vec{s}$-column degree* of $\mathbf{p}$ is

$$\operatorname{cdeg}_{\vec{s}} \mathbf{p} = \max_{1 \le i \le n} [\deg p_i + s_i] = \operatorname{cdeg}(x^{\vec{s}} \cdot \mathbf{p}),$$

where

$$x^{\vec{s}} = \operatorname{diag}\left(x^{s_1}, x^{s_2}, \ldots, x^{s_n}\right).$$

For a matrix $\mathbf{P}$, we use $\operatorname{cdeg} \mathbf{P}$ and $\operatorname{cdeg}_{\vec{s}} \mathbf{P}$ to denote respectively the list of its column degrees and the list of its shifted $\vec{s}$-column degrees. When $\vec{s} = [0, \ldots, 0]$, the shifted column degree specializes to the standard column degree.

Shifted degrees have been used previously in polynomial matrix computations and in generalizations of some matrix normal forms [2]. The shifted column degree is equivalent to the notion of *defect* commonly used in the literature. If one views a maximum degree as a position of importance then one can think of shifted degrees as alternative ways to weigh positions. A good example illustrating this can be found in [1] where one uses maximum degree as a way of determining pivot rows for normal forms. The use of shifted degrees allows one to specify alternate pivot choices for these forms.

Along with shifted degrees we also make use of the notion of a polynomial matrix being column reduced. A polynomial matrix $\mathbf{A} \in \mathbb{K}[x]^{m \times n}$ is column reduced if the leading column coefficient matrix, that is the matrix

$$\operatorname{lcoeff} \mathbf{A} = [\operatorname{coeff}(a_{ij}, x, d_j)]_{1 \le i \le m, 1 \le j \le n}, \text{ with } [d_1, \ldots, d_n] = \operatorname{cdeg} \mathbf{A},$$

has full rank. A polynomial matrix $\mathbf{A}$ is $\vec{s}$-column reduced if $x^{\vec{s}} \cdot \mathbf{A}$ is column reduced.

The usefulness of the shifted degrees can be seen from their applications in polynomial matrix computation problems [11, 12, 13]. One of its uses is illustrated by the following lemma, which follows directly from the definition of shifted degree.

Lemma 1. *Let $\vec{s}$ be a shift whose entries bound the corresponding column degrees of $\mathbf{A} \in \mathbb{K}^{* \times m}$. Then for any polynomial matrix $\mathbf{B} \in \mathbb{K}[x]^{m \times *}$, the column degrees of $\mathbf{A} \cdot \mathbf{B}$ are bounded by the corresponding $\vec{s}$-column degrees of $\mathbf{B}$.*

A closely related result involving the shifted degrees is the following lemma [11, Lemma 2.19], which can be viewed as a stronger version of the *predictable degree property* [5, Theorem 6.3-13].

Lemma 2. *Let $\mathbf{A} \in \mathbb{K}[x]^{* \times m}$ be a $\vec{s}$-column reduced matrix with no zero columns and with $\operatorname{cdeg}_{\vec{s}} \mathbf{A} = \vec{t}$. Then any matrix $\mathbf{B} \in \mathbb{K}[x]^{m \times *}$ satisfies $\operatorname{cdeg}_{\vec{t}} \mathbf{B} = \operatorname{cdeg}_{\vec{s}} (\mathbf{A} \cdot \mathbf{B})$.*

An essential subroutine needed in our computation, also based on the use of the shifted degrees, is the efficient multiplication of a pair of matrices $\mathbf{A}\cdot\mathbf{B}$ with unbalanced degrees. The following result follows as a special case of [13, Theorem 3.7]. The notation $\sum\vec{s}$, for any list $\vec{s}$, denotes the sum of all entries in $\vec{s}$.

Theorem 3. *Let $\mathbf{A}\in\mathbb{K}[x]^{n\times m}$ and $\mathbf{B}\in\mathbb{K}[x]^{m\times m}$ be given with $m\le n$. Assume, without loss of generality, that $\mathbf{B}$ has no zero columns. Suppose $\vec{s}\in\mathbb{Z}_{\ge0}^{n}$ is a shift that bounds the corresponding column degrees of $\mathbf{A}$. If ξ is an upper bound for both $\sum\vec{s}$ and $\sum cdeg_{\vec{s}}\mathbf{B}$, then the product $\mathbf{A}\cdot\mathbf{B}$ can be computed in $(nm^{\omega-1}(1+\xi/m))^{1+o(1)}$ field operations from $\mathbb{K}$.*

Note that ξ/n in the cost estimate in Theorem 3 is an upper bound on the average column degree of $\mathbf{A}$ and the average $\vec{s}$-column degree of $\mathbf{B}$. The cost estimate uses $1+\xi/n$ in order to correctly handle the case when $\vec{s}$ contains some zero entries. In particular, we always have $1+\xi/n\ge1$. Also note that to use the proof of [13, Theorem 3.7], we may assume that the entries of $\vec{s}$ are ordered in increasing order. This can be done by ordering the columns of $\mathbf{A}$ and the rows of $\mathbf{B}$ in the same way as $\vec{s}$.

2.3. Kernel bases

Let $\mathbf{F}\in\mathbb{K}[x]^{m\times n}$ be a matrix of polynomials over a field $\mathbb{K}$. The kernel of $\mathbf{F}\in\mathbb{K}[x]^{m\times n}$ is the $\mathbb{K}[x]$-module

$$\{\mathbf{p}\in\mathbb{K}[x]^{n}\mid\mathbf{F}\cdot\mathbf{p}=0\}$$

with a kernel basis of $\mathbf{F}$ being a basis of this module.

Definition 4. Given $\mathbf{F}\in\mathbb{K}[x]^{m\times n}$, a polynomial matrix $\mathbf{N}\in\mathbb{K}[x]^{n\times *}$ is a $\vec{s}$-minimal (right) kernel basis of $\mathbf{F}$ if $\mathbf{N}$ is a kernel basis of $\mathbf{F}$ and $\mathbf{N}$ is $\vec{s}$-column reduced. We also call a $\vec{s}$-minimal (right) kernel basis of $\mathbf{F}$ a $(\mathbf{F},\vec{s})$-kernel basis.

The following result [13, Theorem 3.4] shows that the use of shifted degrees can ensure that the sizes of certain shifted kernel bases do not get too big.

Lemma 5. *Suppose $\mathbf{F}\in\mathbb{K}[x]^{m\times n}$ and $\vec{s}\in\mathbb{Z}_{\ge0}^{n}$ is a shift with entries bounding the corresponding column degrees of $\mathbf{F}$. Then the sum of the $\vec{s}$-column degrees of any $\vec{s}$-minimal kernel basis of $\mathbf{F}$ is bounded by $\sum\vec{s}$.*

Example 1. Let

$$\mathbf{F}=\begin{bmatrix} x & -x^3 & -2x^4 & 2x & -x^2\\ 1 & -1 & -2x & 2 & -x\\ -3 & 3x^2+x & 2x^2 & -x^4+1 & 3x\end{bmatrix}$$

a 3×5 matrix over $\mathbb{Z}_7[x]$ and let $\vec{s}=(1,3,4,4,2)$, the column degrees of $\mathbf{F}$. Then

$$\mathbf{N}=\begin{bmatrix} -1 & x\\ -x^2 & 0\\ -3\,x & 0\\ -3 & 0\\ 0 & 1\end{bmatrix}\quad\text{with}\quad \operatorname{lcoeff}_{\vec{s}}\mathbf{N}=\begin{bmatrix} 0 & 1\\ -1 & 0\\ -3 & 0\\ 0 & 0\\ 0 & 1\end{bmatrix}$$

is a $\vec{s}$-minimal kernel basis of $\mathbf{F}$ satisfying $\mathbf{F} \cdot \mathbf{N} = \mathbf{0}$ and its $\vec{s}$-shifted leading coefficient has full rank. Note that the sum of the $\vec{s}$-column degrees of $\mathbf{N}$ (in this case 7) is less than $\sum \vec{s} = 14$ as implied in Lemma 5. $\square$

We will also need the following result [13, Lemma 3.15] to compute kernel bases by rows.

Lemma 6. *Let $\mathbf{G} = \left[\mathbf{G}_1^T, \mathbf{G}_2^T\right]^T \in \mathbb{K}[x]^{m \times n}$ and $\vec{t} \in \mathbb{Z}^n$ a shift vector. If $\mathbf{N}_1$ is a $\left(\mathbf{G}_1, \vec{t}\right)$-kernel basis with $\vec{t}$-column degrees $\vec{u}$, and $\mathbf{N}_2$ is a $\left(\mathbf{G}_2\mathbf{N}_1, \vec{u}\right)$-kernel basis with $\vec{u}$-column degrees $\vec{v}$, then $\mathbf{N}_1\mathbf{N}_2$ is a $\left(\mathbf{G}, \vec{t}\right)$-kernel basis with $\vec{t}$-column degrees $\vec{v}$.*

Another essential subroutine we need is the minimal kernel basis algorithm with shift recently reported by the authors [13, Theorem 4.2].

Theorem 7. *Let $\mathbf{F} \in \mathbb{K}^{m \times n}$ with $m \leq n$ and $\vec{s}$ be a shift bounding the corresponding column degrees of $\mathbf{F}$. Then a $\vec{s}$-minimal kernel basis of $\mathbf{F}$ can be computed in $\left(n^{\omega}(1 + \xi/n)\right)^{1+o(1)}$ field operations from $\mathbb{K}$, where $\xi = \sum \vec{s}$.*

3. The inversion algorithm

In order to reduce a matrix $\mathbf{F}$ into diagonal form Jeannerod and Villard [4] first considers a matrix $\mathbf{A}_1$ such that $\mathbf{F} \cdot \mathbf{A}_1 = \mathbf{R}$ has just two diagonal blocks. More specifically, if we separate $\mathbf{F}$ into

$$\mathbf{F} = \left[\begin{array}{c} \mathbf{F}_u \\ \hline \mathbf{F}_d \end{array}\right],$$

with $\mathbf{F}_u$ and $\mathbf{F}_d$ consisting of the upper $\lceil n/2 \rceil$ and lower $\lfloor n/2 \rfloor$ rows of $\mathbf{F}$, respectively, then a matrix $\mathbf{A}_1 = [\mathbf{N}_\ell, \mathbf{N}_r]$ consisting of kernel bases $\mathbf{N}_\ell$ and $\mathbf{N}_r$ of $\mathbf{F}_d$ and $\mathbf{F}_u$, respectively, gives

$$\mathbf{F} \cdot \mathbf{A}_1 = \left[\begin{array}{c} \mathbf{F}_u \\ \hline \mathbf{F}_d \end{array}\right] \cdot [\mathbf{N}_\ell, \mathbf{N}_r] = \begin{bmatrix} \mathbf{F}_u\mathbf{N}_\ell & \mathbf{F}_u\mathbf{N}_r \\ \mathbf{F}_d\mathbf{N}_\ell & \mathbf{F}_d\mathbf{N}_r \end{bmatrix} = \begin{bmatrix} \mathbf{R}_u & 0 \\ 0 & \mathbf{R}_d \end{bmatrix}. \tag{5}$$

If $\mathbf{F}$ is nonsingular, then the column dimensions of $\mathbf{N}_\ell$ and $\mathbf{N}_r$ match the row dimensions of $\mathbf{F}_u$ and $\mathbf{F}_d$, respectively, making the diagonal blocks $\mathbf{R}_u$ and $\mathbf{R}_d$ square.

Example 2. Let

$$\mathbf{F} = \begin{bmatrix} x & -x^3 & -2x^4 & 2x & -x^2 \\ 1 & -1 & -2x & 2 & -x \\ -3 & 3x^2 + x & 2x^2 & -x^4 + 1 & 3x \\ 0 & x^2 & x^4 + 2x^3 - 2x^2 & 0 & 0 \\ 1 & -x^2 + 2 & -2x^3 - 3x & 2x + 2 & 0 \end{bmatrix},$$

a 5×5 matrix over $\mathbb{Z}_7[x]$ with column degrees $\vec{s} = (1,3,4,4,2)$. Then the $\vec{s}$-minimal kernel basis of the upper 3 rows and lower 2 rows of $\mathbf{F}$ are given by

$$\mathbf{N}_{1,\ell}^{(1)} = \begin{bmatrix} -2x-2 & -x^4+x^2-1 & 0 \\ 0 & -x^2-2x+2 & 0 \\ 0 & 1 & 0 \\ 1 & -2x+2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{N}_{1,r}^{(1)} = \begin{bmatrix} -1 & x \\ -x^2 & 0 \\ -3x & 0 \\ -3 & 0 \\ 0 & 1 \end{bmatrix},$$

respectively, with $\mathbf{N}_{1,r}^{(1)}$ determined in Example 1. Multiplying $\mathbf{F}$ on the right by $\mathbf{A}_1 = [\mathbf{N}_{1,\ell}^{(1)}, \mathbf{N}_{1,r}^{(1)}]$ then gives

$$\begin{aligned} \mathbf{F} \cdot \mathbf{A}_1 &= \left[\begin{array}{c|c} \mathbf{R}_u & \\ \hline & \mathbf{R}_d \end{array}\right] \\ &= \left[\begin{array}{ccc|cc} -2x^2 & -x^3+3x^2+3x & -x^2 & & \\ -2x & -x^4+2x^2+3x+1 & -x & & \\ -x^4-x & 2x^5-2x^4+3x^2-2 & 3x & & \\ \hline & & & -3x^5-x^3 & 0 \\ & & & x & x \end{array}\right]. \end{aligned}$$

Notice that the column degrees of both $\mathbf{R}_u$ and $\mathbf{R}_d$ are bounded by the $\vec{s}$-column degrees of $\mathbf{N}_{1,\ell}^{(1)}$and $\mathbf{N}_{1,r}^{(1)}$, respectively. In particular, the sum of the column degrees of both $\mathbf{R}_u$ and $\mathbf{R}_d$ are each bounded by $\sum \vec{s} = 14$. $\square$

In general, the matrices $\mathbf{F}$ and $\mathbf{A}_1$ can have unbalanced degrees. But the use of shifted degrees allows them to be efficiently multiplied making use of Theorem 3. The same process can then be repeated recursively on $\mathbf{R}_u$ and $\mathbf{R}_d$, until we reach the base case where the dimension becomes 1. This gives a recursive algorithm, shown in Algorithm 1, which returns a sequence of matrices $\mathbf{A}_1, \ldots, \mathbf{A}_{\lceil \log n \rceil}$ that transforms the input matrix $\mathbf{F}$ into a diagonal matrix $\mathbf{B}$.

Example 3. Continuing with Example 2, let $\mathbf{R}_u$ and $\mathbf{R}_d$ be the two diagonal blocks of dimension 3×3 and 2×2, respectively. To apply Lemma 6 we base the subsequent shifted kernel computation on the shifted column degrees of the previously computed kernels. Recall that $\vec{s} = (1,3,4,4,2)$. The $\vec{s}$-column degrees of $\mathbf{N}_{1,\ell}^{(1)}$ and $\mathbf{N}_{1,r}^{(1)}$ are $(4,5,2)$ and $(5,2)$, respectively. Repeating our previous computation for $\mathbf{R}_u$ we obtain $(4,5,2)$-minimal kernel bases $\mathbf{N}_{2,\ell}^{(1)}$ and $\mathbf{N}_{2,r}^{(1)}$ for the top 2 rows and bottom row, respectively, given by

$$[\mathbf{N}_{2,\ell}^{(1)}, \mathbf{N}_{2,r}^{(1)}] = \left[\begin{array}{cc|c} 1 & 3x^2-3x & 3 \\ 0 & -2x & 0 \\ -2x^3-2 & 3x^2-x+1 & 1 \end{array}\right]$$

with

$$\mathbf{R}_u \cdot [\mathbf{N}_{2,\ell}^{(1)}, \mathbf{N}_{2,r}^{(1)}] = \left[\begin{array}{cc|c} 2x^5 & x^3 & \\ 2x^4 & 2x^5+x^3+x^2-3x & \\ \hline & & -3x^4 \end{array}\right].$$

Similarly for the 2×2 matrix $\mathbf{R}_d$ we determine $(5,2)$-minimal kernel bases $\mathbf{N}_{2,\ell}^{(2)}$ and $\mathbf{N}_{2,r}^{(2)}$ for the top and bottom row, respectively, obtaining

$$[\mathbf{N}_{2,\ell}^{(2)}, \mathbf{N}_{2,r}^{(2)}] = \left[\begin{array}{c|c} 1 & 0 \\ -1 & 1 \end{array}\right]$$

with

$$\mathbf{R}_d \cdot [\mathbf{N}_{2,\ell}^{(2)}, \mathbf{N}_{2,r}^{(2)}] = \left[\begin{array}{c|c} -3x^5 - x^3 & \\ \hline & x \end{array}\right].$$

Observe how the size — the sum of the shifted degrees of the minimal bases — is bounded by $\sum \vec{s}$, where $\vec{s}$ is the original bound on the column degrees of $\mathbf{F}$. The $(4,5,2)$-minimal bases have shifted degrees $(5,6)$ and 4 on the left while the $(5,2)$-minimal basis on the left has shifted degrees 5 and 1. In all cases these shifted degrees bound the recursive matrix polynomials $\mathbf{R}_u$ and $\mathbf{R}_d$ at each step and hence their size (sum of bound of column degrees is always at most $\sum \vec{s}$ for the original bound $\vec{s}$.

Setting now

$$\begin{aligned} \mathbf{A}_2 &= \left[\begin{array}{c|c} [\mathbf{N}_{2,\ell}^{(1)}, \mathbf{N}_{2,r}^{(1)}] & \\ \hline & [\mathbf{N}_{2,\ell}^{(2)}, \mathbf{N}_{2,r}^{(2)}] \end{array}\right] \\ &= \left[\begin{array}{ccc|cc} 1 & 3x^2 - 3x & 3 & & \\ 0 & -2x & 0 & & \\ -2x^3 - 2 & 3x^2 - x + 1 & 1 & & \\ \hline & & & 1 & 0 \\ & & & -1 & 1 \end{array}\right] \end{aligned}$$

we get

$$\mathbf{F} \cdot \mathbf{A}_1 \cdot \mathbf{A}_2 = \begin{bmatrix} 2x^5 & x^3 & & & \\ 2x^4 & 2x^5 + x^3 + x^2 - 3x & & & \\ & & -3x^4 & & \\ & & & -3x^5 - x^3 & \\ & & & & x \end{bmatrix}. \qquad (6)$$

It remains to diagonalize the 2×2 leading submatrix $\mathbf{R}_u \cdot [\mathbf{N}_{2,\ell}^{(1)}, \mathbf{N}_{2,r}^{(1)}]$ of (6). The $(4,5,2)$-column degrees of $\mathbf{N}_{2,\ell}^{(1)}$ are $(5,6)$. The final step of the recursion gives the $(5,6)$-minimal kernel bases $\mathbf{N}_{3,\ell}^{(1)}$ and $\mathbf{N}_{3,r}^{(1)}$, where

$$[\mathbf{N}_{3,\ell}^{(1)}, \mathbf{N}_{3,r}^{(1)}] = \left[\begin{array}{c|c} -3x^4 + 2x^2 + 2x + 1 & 1 \\ 3x^3 & -2x^2 \end{array}\right]$$

with

$$\mathbf{R}_u \cdot [\mathbf{N}_{2,\ell}^{(1)}, \mathbf{N}_{2,r}^{(1)}] \cdot [\mathbf{N}_{3,\ell}^{(1)}, \mathbf{N}_{3,r}^{(1)}] = \left[\begin{array}{c|c} x^9 - 3x^7 + 2\,x^5 & \\ \hline & 3x^7 - 2x^5 - x^3 \end{array}\right].$$

Note that $\mathbf{A}_3$ is a block diagonal matrix with 4 diagonal blocks. Because the original matrix dimension 5 is not a power of two, the algorithm simply takes $[\mathbf{N}^{(3)}_{k,\ell}, \mathbf{N}^{(3)}_{k,r}] = I_1$ for $k = 2, 3, 4$. Thus, with

$$\begin{aligned}
\mathbf{A}_3 &= \begin{bmatrix} [\mathbf{N}^{(3)}_{1,\ell}, \mathbf{N}^{(3)}_{1,r}] & & & \\ & [\mathbf{N}^{(3)}_{2,\ell}, \mathbf{N}^{(3)}_{2,r}] & & \\ & & [\mathbf{N}^{(3)}_{3,\ell}, \mathbf{N}^{(3)}_{3,r}] & \\ & & & [\mathbf{N}^{(3)}_{4,\ell}, \mathbf{N}^{(3)}_{4,r}] \end{bmatrix} \\
&= \begin{bmatrix} 1 + 2x^2 + 2x - 3x^4 & 1 & & & \\ 3x^3 & -2x^2 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}
\end{aligned}$$

we obtain the final diagonal form $\mathbf{B}$ is then given as

$$\mathbf{F} \cdot \mathbf{A}_1 \cdot \mathbf{A}_2 \cdot \mathbf{A}_3 = \begin{bmatrix} x^9 - 3x^7 + 2x^5 & & & & \\ & 3x^7 - 2x^5 - x^3 & & & \\ & & -3x^4 & & \\ & & & -3x^5 - x^3 & \\ & & & & x \end{bmatrix}.$$

Notice how the sizes of intermediate computations (as measured by the sum of column degrees) are continue to be controlled during the recursion. The input column degrees are $\vec{s} = (1, 3, 4, 4, 2)$ with sum 14 and the initial $\vec{s}$-minimal kernel bases have $\vec{s}$-column degrees $(4, 5, 2)$ and $(5, 2)$, in each case summing to less than 14. The initial diagonal blocks $\mathbf{R}_u$ and $\mathbf{R}_d$ then have column degrees bounded by $(4, 5, 2)$ and $(5, 2)$, respectively, and hence their sums are also less than the sum of the initial column degrees. Recursing on the upper block implies sums are at most 11 while on the lower block the sums are at most 7. In all cases the sums are at most 14 for any shifted minimal kernel basis computation. $\square$

Algorithm 1 recurses on two problems of about half the dimension. The following theorem establishes a bound on the cost of the algorithm. The key idea of the cost analysis is to show that the sum of column degrees of each of the recursive problems is bounded by the sum of the column degrees of $\mathbf{F}$. In addition to the input matrix $\mathbf{F}$ the algorithm takes as input a shift $\vec{s}$ that bounds the corresponding degrees of columns of $\mathbf{F}$. For the top level call to the algorithm we can choose $\vec{s} = \operatorname{cdeg} \mathbf{F}$, in which case the parameter s in the following theorem is the average column degree of $\mathbf{F}$ plus one.

Theorem 8. *Algorithm 1 is correct. The cost of the algorithm is bounded by $(n^\omega s)^{1+o(1)}$ field operations from $\mathbb{K}$, where $s = 1 + (\sum \vec{s})/n$.*

Proof. First note that the base case $n = 1$ returns a correct output. Next, the algorithm will report fail at line 3 if and only if $\mathbf{F}$ is singular. Finally, Lemma 1

Algorithm 1 Inverse($\mathbf{F}, \vec{s}$)

```
Input: $\mathbf{F} \in \mathbb{K}[x]^{n\times n}$, $\vec{s} \in \mathbb{Z}_{\geq 0}^{n}$ such that entries of $\vec{s}$ bound the corresponding
    column degrees of $\mathbf{F}$
Output: $\mathcal{A} = \left[\mathbf{A}_1, \ldots, \mathbf{A}_{\lceil \log n \rceil}\right], \mathbf{B}$ with $\mathbf{A}_1, \ldots, \mathbf{A}_{\lceil \log n \rceil}, \mathbf{B} \in \mathbb{K}[x]^{n\times n}$ such
    that $\mathbf{B}$ is diagonal and $\mathbf{A}_1 \ldots \mathbf{A}_{\lceil \log n \rceil}\mathbf{B}^{-1} = \mathbf{F}^{-1}$ if $\mathbf{F}$ is nonsingular, or fail
    if $\mathbf{F}$ is singular.
1: if $\mathbf{F} = 0$ then fail endfi;
   if $n = 1$ then return $\{[\,], \mathbf{F}\}$ endif;
2: $\mathbf{F} := \begin{bmatrix} \mathbf{F}_u \\ \mathbf{F}_d \end{bmatrix}$ with $\mathbf{F}_u$ consisting of the upper $\lceil n/2 \rceil$ rows of $\mathbf{F}$;
3: $\mathbf{N}_r :=$ MinimalKernelBasis($\mathbf{F}_u, \vec{s}$);
   $\mathbf{N}_\ell :=$ MinimalKernelBasis($\mathbf{F}_d, \vec{s}$);
   if ColumnDimension($[\mathbf{N}_\ell, \mathbf{N}_r]$) $\neq n$ then fail endif;
4: $\mathbf{R}_u := \mathbf{F}_u \mathbf{N}_\ell$;
   $\mathbf{R}_d := \mathbf{F}_d \mathbf{N}_r$;
5: $\left\{\mathcal{A}^{(1)}, \mathbf{B}_1\right\} :=$ Inverse($\mathbf{R}_u, \mathrm{cdeg}_{\vec{s}}\mathbf{N}_\ell$);
   $\left\{\mathcal{A}^{(2)}, \mathbf{B}_2\right\} :=$ Inverse($\mathbf{R}_d, \mathrm{cdeg}_{\vec{s}}\mathbf{N}_r$);
6: $\mathcal{A} := \left[[\mathbf{N}_\ell, \mathbf{N}_r], \mathrm{diag}(\mathcal{A}_1^{(1)}, \mathcal{A}_1^{(2)}), \ldots, \mathrm{diag}(\mathcal{A}_{\lceil \log n \rceil - 1}^{(1)}, \mathcal{A}_{\lceil \log n \rceil - 1}^{(2)})\right]$;
   # Note: If $\mathcal{A}_{\lceil \log n \rceil - 1}^{(2)}$ is not defined then substitute the identity matrix.
7: return $\{\mathcal{A}, \mathrm{diag}(\mathbf{B}_1, \mathbf{B}_2)\}$;
```

gives that the column degrees of $\mathbf{R}_u$ and $\mathbf{R}_d$ are bounded by the corresponding $\vec{s}$-column degrees of $\mathbf{N}_\ell$ and $\mathbf{N}_r$, respectively. Thus, the arguments in the recursive calls in line 5 satisfy the precondition of the algorithm. Correctness now follows using strong induction on n.

For $\xi \geq 1$, let $T(n, \xi)$ be a bound on the cost of the algorithm with input ($\mathbf{F} \in \mathbb{K}[x]^{n\times n}, \vec{s} \in \mathbb{Z}^n$) that satisfies $\sum \vec{s} \leq \xi$. To obtain a recurrence relation for T we first claim that the cost of the nonrecursive work done in the algorithm is bounded by $(n^\omega(1 + \xi/n))^{1+o(1)}$. Note that lines 1, 2, 6 and 7 do not require any field operations. The claim for line 3 follows using the algorithm supporting Theorem 7. By Lemma 5 we have

$$\sum \mathrm{cdeg}_{\vec{s}}\mathbf{N}_\ell \leq \sum \vec{s} \quad \text{and} \quad \sum \mathrm{cdeg}_{\vec{s}}\mathbf{N}_r \leq \sum \vec{s}, \tag{7}$$

so the claim for line 4 follows from Theorem 3.

Finally, consider line 5. Since the column degrees of $\mathbf{R}_u$ are bounded by the corresponding $\vec{s}$-column degrees of $\mathbf{N}_\ell$, it follows from (7) that $\sum \mathrm{cdeg}\,\mathbf{R}_u \leq \sum \vec{s}$. Similarly, $\sum \mathrm{cdeg}\,\mathbf{R}_d \leq \sum \vec{s}$. This shows that

$$T(n, \xi) \leq T(\lfloor n/2 \rfloor, \xi) + T(\lceil n/2 \rceil, \xi) + (n^\omega(1 + \xi/n))^{1+o(1)}.$$

Solving this recurrence shows that $T(n, \xi)$ is bounded by $(n^\omega(1 + \xi/n))^{1+o(1)}$. Since $\xi \leq n(s - 1)$ the result follows. □

The use of shifted degrees allowed us to bound the cost of our inversion computation in terms of a bound $s \geq 1$ for the *average* column degree instead of the *maximal* column degree. Since $\mathbf{F}^{-1} = ((\mathbf{F}^T)^{-1})^T$, the complexity bound $(n^3 s)^{1+o(1)}$ we have established for polynomial matrix inversion holds more generally with $s - 1$ equal to the minimum of the average of the row degrees and the average of the column degrees of $\mathbf{F}$.

In some cases it is also possible to efficiently handle matrices which have both some rows and some columns of large degree, for example a matrix polynomial having degrees

$$\operatorname{degs} \mathbf{F} = \begin{bmatrix} nd & nd & \cdots & nd \\ nd & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ nd & 0 & \cdots & 0 \end{bmatrix}. \tag{8}$$

First note that an *a priori* bound for $\deg \det \mathbf{F}$ (and for the degree of any minor of $\mathbf{F}$) is given by ns, leading to the bound $O(n^3 s)$ for the space required to write down $\mathbf{F}^{-1}$. For some input matrices, including the one shown in (8), the upper bound ns on $\deg \det \mathbf{F}$ can be pessimistic. Considering only the degrees of entries, the best *a priori* bound for $\deg \det \mathbf{F}$ is given from the definition of the determinant:

$$\deg \det \mathbf{F} \leq \max_{\sigma \in S_n} \sum_{i=1}^{n} \deg \mathbf{F}_{i,\sigma},$$

where S_n is the set of all permutations of $(1, 2, \ldots, n)$.

The minimum of the average row degree and average column degree of $\mathbf{F}$ in (8) is $s = nd$. However the definition of the determinant gives $\deg \det \mathbf{F} \leq 2nd$. Partial linearization [3, Section 6] can be used to handle such inputs efficiently. Up to a row and column permutation, assume that $\deg \mathbf{F}_{i,i}$ bounds the degree of all entries in the trailing submatrix $\mathbf{F}_{i\ldots n, i\ldots n}$, $1 \leq i \leq n$, and let $E = \sum_{i=1}^{n} \deg \mathbf{F}_{i,i}$. Then corresponding to $\mathbf{F}$ there exists a matrix $\bar{\mathbf{F}}$ that satisfies the following properties: Dimension$(\bar{\mathbf{F}}) < 3n$; $\deg \bar{\mathbf{F}} \leq \lceil E/n \rceil$; the principal $n \times n$ submatrix of $\bar{\mathbf{F}}^{-1}$ is equal to $\mathbf{F}^{-1}$. The matrix shown in (8), for example, would be transformed [3, Corollary 3] into a matrix $\bar{\mathbf{F}}$ with dimension $3n - 1$ and degree d.

4. Cost of multiplying the output matrices

Throughout this section, let $\mathbf{A}_1, \ldots, \mathbf{A}_{\lceil \log n \rceil}, \mathbf{B}$ be the output of Algorithm 1 for an input $(\mathbf{F} \in \mathbb{K}[x]^{n \times n}, \vec{s} \in \mathbb{Z}^n)$. To compute $\mathbf{A} = \mathbf{A}_1 \cdots \mathbf{A}_{\lceil \log n \rceil}$ we simply multiply the matrices in sequential order. Let $\mathbf{M}_i = \mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_i$, $1 \leq i \leq \lceil \log n \rceil$. At step i, for $i = 1, 2, \ldots, \lceil \log n \rceil - 1$, we compute the product $\mathbf{M}_{i+1} = \mathbf{M}_i \cdot \mathbf{A}_{i+1}$.

The matrix $\mathbf{A}_{i+1}$ consists of 2^i diagonal blocks, that is,

$$\mathbf{A}_{i+1} = \operatorname{diag}(\mathbf{A}_{i+1}^{(1)}, \ldots, \mathbf{A}_{i+1}^{(2^i)}),$$

with each diagonal block $\mathbf{A}_{i+1}^{(k)} = [\mathbf{N}_{i+1,\ell}^{(k)}, \mathbf{N}_{i+1,r}^{(k)}]$ consisting of two kernel bases computed in one of the subproblems. The matrix $\mathbf{M}_i$ can be decomposed as

$$\mathbf{M}_i = [\mathbf{M}_i^{(1)}, \ldots, \mathbf{M}_i^{(2^i)}]$$

with 2^i column blocks, where the column dimension of $\mathbf{M}_i^{(k)}$ corresponds to the column dimension of $\mathbf{A}_{i+1}^{(k)}$, $1 \le k \le 2^i$. The matrix

$$\mathbf{M}_{i+1} = [\mathbf{M}_{i+1}^{(1)}, \mathbf{M}_{i+1}^{(2)}, \ldots, \mathbf{M}_{i+1}^{(2\cdot 2^i-1)}, \mathbf{M}_{i+1}^{(2\cdot 2^i)}]$$

is thus defined by the matrix products

$$\mathbf{M}_{i+1}^{(2k-1)} = \mathbf{M}_i^{(k)} \cdot \mathbf{N}_{i+1,\ell}^{(k)} \quad \text{and} \quad \mathbf{M}_{i+1}^{(2k)} = \mathbf{M}_i^{(k)} \cdot \mathbf{N}_{i+1,r}^{(k)} \tag{9}$$

for $1 \le k \le 2^i$.

It is interesting to note that each $\mathbf{M}_i^{(k)}$ is in the kernel of a subset of the rows of the original input matrix $\mathbf{F}$. This can be seen from the fact that the product $\mathbf{F} \cdot \mathbf{M}_i$ is a matrix with only diagonal blocks nonzero. The following theorem makes precise the type of kernel elements and the rows they annihilate.

Theorem 9. *Let $\mathbf{F}_i^{(k)}$ be a matrix consisting of the rows of $\mathbf{F}$ that have the same row indices as the column indices of $\mathbf{M}_i^{(k)}$ in $\mathbf{M}_i$, and $\bar{\mathbf{F}}_i^{(k)}$ consisting of the remaining rows of $\mathbf{F}$. Then $\mathbf{M}_i^{(k)}$ is a $\vec{s}$-minimal kernel basis of $\bar{\mathbf{F}}_i^{(k)}$.*

Proof. The shifted minimal kernel basis computation in Algorithm 1 use the shift specified by Lemma 6. The only rows of $\mathbf{F}$ not used in computing $\mathbf{M}_i^{(k)}$ are the rows from $\mathbf{F}_i^{(k)}$. □

We can now establish good bounds on the sizes of the matrices involved in the products shown in (9) and subsequently bound the cost of the multiplications.

Lemma 10. *Let $\vec{t} = cdeg_{\vec{s}}\mathbf{M}_i^{(k)}$. Then*

(a) $\sum \vec{t} \le \sum \vec{s}$, and

(b) $\sum cdeg_{\vec{t}}\mathbf{N}_{i+1,\ell}^{(k)} \le \sum \vec{s}$ and $\sum cdeg_{\vec{t}}\mathbf{N}_{i+1,r}^{(k)} \le \sum \vec{s}$.

Proof. The bound for $\sum \vec{t}$ follows from Lemma 5 as a corollary of Theorem 9. For the second two bounds, Lemma 2 gives that $\text{cdeg}_{\vec{t}}\mathbf{N}_{i+1,\ell}^{(k)} = \text{cdeg}_{\vec{s}}\mathbf{M}_{i+1}^{(2k-1)}$ and $\text{cdeg}_{\vec{t}}\mathbf{N}_{i+1,r}^{(k)} = \text{cdeg}_{\vec{s}}\mathbf{M}_{i+1}^{(2k)}$. But now the first claim in the lemma gives that $\sum \text{cdeg}_{\vec{s}}\mathbf{M}_{i+1}^{(2k-1)} \le \sum \vec{s}$ and $\sum \text{cdeg}_{\vec{s}}\mathbf{M}_{i+1}^{(2k)} \le \sum \vec{s}$, so the result follows. □

Lemma 11. *For a given i and k the matrix multiplications in (9) can be done in time $(n(n/2^i)^{\omega-1}(1+\xi/(n/2^i)))^{1+o(1)}$, where $\xi = \sum \vec{s}$.*

Proof. It will suffice to bound the cost of the multiplication $\mathbf{M}_i^{(k)} \cdot \mathbf{N}_{i+1,\ell}^{(k)}$ since the other multiplication is similar. Let $\vec{t} = \mathrm{cdeg}_{\vec{s}}\,\mathbf{M}_i^{(k)}$, as in Lemma 10. Note that $\vec{t}$ bounds the corresponding column degrees of $x^{\vec{s}} \cdot \mathbf{M}_i^{(k)}$. By Lemma 10 we have $\sum \mathrm{cdeg}(x^{\vec{s}} \cdot \mathbf{M}_i^{(k)}) \le \xi$ and $\sum \mathrm{cdeg}_{\vec{t}}\,\mathbf{N}_{i+1,\ell}^{(k)} \le \xi$. Up to a factor of 2, the column dimensions of $\mathbf{M}_i^{(k)}$ and $\mathbf{N}_{i+1,\ell}^{(k)}$ are $n/2^i$, so the result now follows from Theorem 3. □

The product $(((\mathbf{A}_1\mathbf{A}_2)\mathbf{A}_3)\cdots\mathbf{A}_{\log_n})$ can be computed by performing the products in (9) for $i = 1, 2, \ldots, \lceil \log n \rceil - 1$ and $k = 1, \ldots, 2^i$. Multiplying the cost estimate of Lemma 11 by $2^i \times \log n$, substituting $\xi = \sum \vec{s}$ and $\omega = 3$, and simplifying, gives the following result.

Theorem 12. *The product* $\mathbf{A} = \mathbf{A}_1 \cdots \mathbf{A}_{\lceil \log n \rceil}$ *can be computed in* $(n^3 s)^{1+o(1)}$ *operations from* $\mathbb{K}$*, where* $s = 1 + (\sum \vec{s})/n$.

5. Applications

We show how the inverse algorithm for non-generic input can be used to obtain improved complexity bounds for solving three other problems: determining the largest invariant factor of a polyomial matrix, computing a sequence of powers of a constant matrix, and solving a linear system with multiple right hand sides. All the results we mention here follow as a corollary of Theorem 8.

5.1. Largest invariant factor

The largest invariant factor of a matrix of polynomials $\mathbf{F} \in \mathbb{K}[x]^{n\times n}$ is defined as the ratio of the determinant and the gcd of all $n-1$ minors of $\mathbf{F}$. It coincides with the last diagonal entry of the Smith normal form of $\mathbf{F}$. Alternatively, the largest invariant factor is the minimal degree monic polynomial p having the property that $p \cdot \mathbf{F}^{-1}$ is a polynomial matrix. From Theorem 9, each column $\mathbf{a}$ of $\mathbf{A} = \mathbf{A}_1 \cdots \mathbf{A}_{\lceil \log n \rceil}$ is a kernel basis for a subset of $n-1$ rows of $\mathbf{F}$. It follows that the gcd g of the entries in $\mathbf{a}$ must be 1 to allow the kernel element $\mathbf{a}/g$ to be generated by the kernel basis $\mathbf{a}$. Since $\mathbf{A} = \mathbf{F}^{-1}\mathbf{B}$, the diagonal entries of the matrix $\mathbf{B}$ have the smallest possible degrees among that of all full rank diagonal matrix that can be multiplied to $\mathbf{F}^{-1}$ to get a polynomial matrix. The least common multiple of the diagonal entries of $\mathbf{B}$ is therefore equal to p. In particular, if $\mathbf{F}$ has aveage column degree s, the largest invariant factor of $\mathbf{F}$ can be computed in time $(n^\omega s)^{1+o(1)}$.

Example 4. Let $\mathbf{F}$ be given as in Example 2, with the final diagonal form

$$\mathbf{B} = \mathrm{diag}(x^9 - 3x^7 + 2x^5,\ 3x^7 - 2x^5 - x^3, -3x^4, -3x^5 - x^3,\ x) \in \mathbb{Z}_7[x] \quad (10)$$

as in Example 3. The largest invariant factor of $\mathbf{F}$ is the least common multiple of the entries of $\mathbf{B}$, which is equal to $x^9 - 3x^7 + 2x^5$. For comparison, the Smith form of $\mathbf{F} \in \mathbb{Z}_7\,[x]$ is $\mathrm{diag}(1,\ 1\,, x\,, x^2\,, x^9 - 3x^7 + 2x^5)$. □

5.2. Fast computation of a sequenc of matrix powers

The largest invariant factor of a matrix is a divisor of the determinant and plays the same role as the minimal polynomial of a scalar matrix. Indeed, if $\mathbf{F} = xI - V$ with $V \in \mathbb{K}^{n\times n}$ then the largest invariant factor of $\mathbf{F}$ is precisely the minimal polynomial of V, a divisor of the characteristic polynomial. Deterministic algorithms are already known to compute the characteristic polynomial [6] in $O(n^\omega \log n)$ and the minimal polynomial [7] in $O(n^\omega(\log n)(\log\log n))$ field operations. On the other hand, if we consider the matrix $\mathbf{F} = I - xV$, then the x-adic expansion $\mathbf{F}^{-1} = I + xV + x^2V^2 + \ldots$ reveals the powers of V. By computing $\mathbf{F}^{-1}$ explicitly using Algorithm 1 and then taking the truncated series expansions of the entries, we see that the sequence of matrix powers $I, V, V^2, \ldots, V^n$ for an arbitrary $V \in \mathbb{K}^{n\times n}$ can be computed deterministically in $(n^3)^{1+o(1)}$ operations from $\mathbb{K}$.

5.3. Linear system solving over $\mathbb{K}(x)$ with multiple right hand sides

Consider the problem of computing, for a given $\mathbf{F} \in \mathbb{K}[x]^{n\times n}$, multiple linear system solutions

$$\mathbf{v}_1\mathbf{F}^{-1}, \mathbf{v}_2\mathbf{F}^{-1}, \ldots, \mathbf{v}_k\mathbf{F}^{-1} \tag{11}$$

for vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k \in \mathbb{K}[x]^{1\times n}$. Suppose each $\mathbf{v}_i$ is bounded in degree by s, the average column degree of $\mathbf{F}$. If the representation $\mathbf{F}^{-1} = \mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_{\lceil\log n\rceil}\mathbf{B}^{-1}$ has been precomputed, the $\mathbf{v}_i\mathbf{F}^{-1} = \mathbf{v}_i\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_{\lceil\log n\rceil}\mathbf{B}^{-1}$ can be computed in time $(n^2s)^{1+o(1)}$. Thus, if $k \in O(n^{\omega-2})$ then all k of the linear system solutions in (11) can be computed in succession in $(n^\omega s)^{1+o(1)}$ time.

6. Conclusion

In this paper, following [11, 12, 13], we have derived cost estimates only up to logarithmic factors. It would be useful to derive more informative and precise cost estimates which make explicit the logarithmic factors and make use of cost functions for polynomial multiplication. The first step in such a project is to analyse more precisely the algorithms from [13] on which our algorithm depends. Note that the algorithms in [13] depend on those in [12], which would also need to be reanalysed.

The algorithm in this paper, as well as those in [12, 13], may make attractive candidates for implementation not only becuse of their good asymptotic complexity but because they are deterministic. In particular, the algorithms are applicable over arbitrary fields, especially also small finite fields, without the need for field extenstions. Algorithm 1 has been implemented in Maple but only to check for correctness of the recursive steps. The algorithm has not yet been implemented in an environment that supports fast matrix multiplication, something which we leave to future work. The main challenge is to obtain optimized implementations of the subroutines in [13] for shifted kernel basis computation and skew degree matrix polynomial multiplication.

REFERENCES

[1] B. Beckermann, G. Labahn, and G. Villard. Shifted normal forms of polynomial matrices. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ISSAC'99, pages 189–196, 1999.

[2] B. Beckermann, G. Labahn, and G. Villard. Normal forms for general polynomial matrices. *Journal of Symbolic Computation*, 41(6):708–737, 2006.

[3] S. Gupta, S. Sarkar, A. Storjohann, and J. Valeriote. Triangular x-basis decompositions and derandomization of linear algebra algorithms over $\mathsf{K}[x]$. *Journal of Symbolic Computation*, 47(4):422–453, October 2012. Festschrift for the 60th Birthday of Joachim von zur Gathen.

[4] C. P. Jeannerod and G. Villard. Essentially optimal computation of the inverse of generic polynomial matrices. *Journal of Complexity*, 21(1):72–86, 2005.

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

[6] W. Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theoretical Computer Science*, 36:309—317, 1985.

[7] A. Storjohann. Deterministic computation of the Frobenius form (Extended Abstract). In *Proc. 42nd Annual Symp. Foundations of Comp. Sci.*, pages 368–377, Los Alamitos, California, 2001. IEEE Computer Society Press.

[8] A. Storjohann. High–order lifting and integrality certification. *Journal of Symbolic Computation*, 36(3–4):613–648, 2003.

[9] A. Storjohann. On the complexity of inverting integer and polynomial matrices. *Computational Complexity*, 2010. Accepted for publication.

[10] J. von zur Gathen and J Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2nd edition edition, 2003.

[11] W. Zhou. *Fast Order Basis and Kernel Basis Computation and Related Problems*. PhD thesis, University of Waterloo, 2012.

[12] W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *Journal of Symbolic Computation*, 47:793–819, 2012.

[13] W. Zhou, G. Labahn, and A. Storjohann. Computing minimal nullspace bases. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ISSAC'12, pages 375–382. ACM, 2012.