Modular Computation for Matrices of Ore Polynomials

Howard Cheng

Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Canada
E-mail: howard.cheng@uleth.ca

George Labahn

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
E-mail: glabahn@uwaterloo.ca

We give a modular algorithm to perform row reduction of a matrix of Ore polynomials with coefficients in $\mathbb{Z}[t]$. Both the transformation matrix and the transformed matrix are computed. The algorithm can be used for finding the rank and left nullspace of such matrices. In the special case of shift polynomials, we obtain algorithms for computing a weak Popov form and for computing a greatest common right divisor (GCRD) and a least common left multiple (LCLM) of matrices of shift polynomials. Our algorithms improve on existing fraction-free algorithms and can be viewed as generalizations of the work of Li and Nemes on GCRDs and LCLMs of Ore polynomials. We define lucky homomorphisms, determine the appropriate normalization, as well as bound the number of homomorphic images required. Our algorithm is output-sensitive, such that the number of homomorphic images required depends on the size of the output. Furthermore, there is no need to verify the result by trial division or multiplication. When our algorithm is used to compute a GCRD and a LCLM of shift polynomials, we obtain a new output-sensitive modular algorithm.

1. Introduction

Ore polynomials provide a general setting for describing linear differential, difference and q-difference operators.[15] Systems of equations defined by these operators can be represented by matrices of Ore polynomials. In this paper we look at the problem of transforming such matrices into "simpler" ones using only certain row operations. Examples of such transformations include conversion to special forms, such as row-reduced and weak Popov normal forms.[3,6,10,14]

Performing row reductions on a matrix of Ore polynomials to these simpler forms allows one to determine its rank and left nullspace, giving the minimum number of equations needed to represent the system of equations.[2,6] If the transformation is invertible, the normal form in fact gives the matrix representing an equivalent system with a minimum number of equations. When the leading coefficient is triangular (as in the weak Popov form), the normal form allows one to rewrite high-order operators (e.g. derivatives) in terms of lower ones (Example 2.1). These transformations can also be applied to the computation of greatest common right divisors (GCRDs) and least common left multiples (LCLMs),[3,6,11–13] which represents the intersection and the union of the solution spaces of systems of equations.

The general problem of row reduction was discussed in Refs. 3 and 6. The problem of defining and computing Popov forms over non-commutative valuation domains such as rings of Ore polynomials was considered in Ref. 9, but efficient computation of Popov forms is not considered. In practice, row reductions can introduce significant coefficient growth which must be controlled. This is important in the case of Ore polynomials as coefficient growth is introduced in two ways—from multiplying on the left by powers of the indeterminate and from elimination by cross-multiplication.

Fraction-free algorithms were given in Refs. 3 and 6 to compute the rank and a left nullspace of matrices of Ore polynomials. When the matrix entries are shift polynomials, we obtained fraction-free algorithms for computing row-reduced and weak Popov forms, and for computing a greatest common right divisor (GCRD) or a least common left multiple (LCLM) of matrices of shift polynomials. It was shown that the fraction-free algorithms can be viewed as a generalization of the subresultant algorithm of Li[11,12] to the case of matrices. Fraction-free methods allow us to control the growth of intermediate results at a reasonable cost, leading to polynomial time algorithms.

Modular computation is generally faster than the corresponding fraction-free computation in a number of problems.[8,17] In this paper we are interested in modular algorithms for row reduction of matrices of Ore polynomials, as well as computing the associated transformation matrices. There are traditionally three major issues that must be addressed: the problem of "unlucky" homomorphisms, the number of images required for the reconstruction of the result, and the normalization of the result to compute consistent images under different homomorphisms.

In the case of polynomial matrices, these issues are overcome by formu-

lating the row reduction problem as a linear algebra problem. We obtained a modular algorithm which has a lower complexity than the fraction-free counterpart.[7] The algorithm is output-sensitive, so that the number of homomorphic images required depends on the size of the output. Furthermore, there is no need to verify the result by trial division or multiplication. We note that in many modular algorithms, this verification step can be a significant bottleneck. Experimental results showed that when the matrix entries are the commutative polynomials with coefficients in $\mathbb{Z}[t]$, the output-sensitive modular algorithm is significantly superior to the fraction-free algorithm. For matrices of Ore polynomials, the obvious modular algorithm consists of two parts. First, the problem in $\mathbb{Z}[t][Z;\sigma,\delta]$ are reduced to problems in $\mathbb{Z}_p[t][Z;\sigma,\delta]$. Next, evaluation maps $t \leftarrow \alpha$ are applied to reduce the problems to ones whose coefficients are in $\mathbb{Z}_p$. However, such evaluations are generally not Ore ring homomorphisms. As a result, it is not possible to apply the same technique used in the polynomial matrix case.

For the problem of computing GCRDs and LCLMs of Ore polynomials, Li mapped the problem into a linear algebra problem over $\mathbb{Z}_p$ by applying the evaluation map to the entries of the Sylvester matrix of the input polynomials.[13] In the case of Ore polynomial matrices, however, the dimensions and the configuration of the final coefficient matrix (the striped Krylov matrix[3,6]) are not known a priori. Thus, the approach of Li[13] cannot be applied directly in our case.

The purpose of this paper is to overcome these difficulties for matrices of Ore polynomials. We show how these issues are resolved by studying the linear algebra formulation of the problem. We also extend the approach of Cabay[5] to obtain output-sensitive algorithms which do not require trial division or multiplication to verify the results. The complexity of the new modular algorithm improves on the complexity of the fraction-free algorithm. Furthermore, the algorithm can be significantly faster when the size of output is small. We also obtain a new output-sensitive modular algorithm for computing GCRDs and LCLMs of shift polynomials. Experimental results confirm the performance of the modular algorithm as predicted by the complexity analysis.

The paper is organized as follows. In Sec. 2, we review the relevant definitions of matrices of Ore polynomials as well as the fraction-free elimination algorithm of Refs. 3 and 6. In Sec. 3, we give a linear algebra formulation of the problem and illustrate the difficulties in obtaining a modular algorithm. We then discuss the reduction $\mathbb{Z}[t][Z;\sigma,\delta] \rightarrow \mathbb{Z}_p[t][Z;\sigma,\delta]$. The

techniques are then extended to the reduction to linear algebra problems in $\mathbb{Z}_p$ in the next section. In Sec. 6 we study the complexity of the algorithms presented. Implementation considerations and experimental results are discussed in Sec. 7. Concluding remarks are discussed in the closing section.

2. Preliminaries

2.1. *Notation*

For any matrix A, we denote its elements by $A_{i,j}$. For any sets of row and column indices I and J, we denote by $A_{I,J}$ the submatrix of A consisting of the rows and columns indexed by I and J. For convenience, we use $*$ for I or J to denote the sets of all rows and columns.

For any vector of integers (also called *multi-index*) $\vec{\omega} = (\omega_1, \ldots, \omega_p)$, we denote by $|\vec{\omega}| = \sum_{i=1}^{p} \omega_i$. The function $\max(\cdot, \cdot)$ gives the vector whose components are the maximum of the corresponding components of its input vectors. We say that $\vec{v} \leq_{\text{lex}} \vec{w}$ if $\vec{v} = \vec{w}$ or if the leftmost nonzero entry in $\vec{v} - \vec{w}$ is negative. The vector $\vec{e}_i$ denotes the i-th unit vector (of the appropriate dimension) such that $(e_i)_j = \delta_{ij}$; we also have $\vec{e} = (1, \ldots, 1)$ (of the appropriate dimension). We denote by $\mathbf{I}_m$ the $m \times m$ identity matrix, and by $Z^{\vec{\omega}}$ the diagonal matrix having Z^{ω_i} on the diagonal.

2.2. *Definitions*

We first give some definitions on Ore polynomial matrices. These definitions are similar to those given in our previous work.[3,6,7]

We denote by $\mathbb{Z}$ the ring of integers, $\mathbb{Q}$ the field of rational numbers, and $\mathbb{Z}_p$ the finite field of p elements where p is prime.

Let k be any field and let $\sigma : k \to k$ be an injective endomorphism of k. Then, $\delta : k \to k$ is a *derivation* with respect to σ is an endomorphism of the additive group of k satisfying

$$\delta(rs) = \sigma(r)\delta(s) + \delta(r)s$$

for all r, $s \in k$. In this paper, we will examine Ore polynomial rings with coefficients in $\mathbb{Z}[t]$. That is, the ring $\mathbb{Z}[t][Z; \sigma, \delta]$ with σ an automorphism, δ a derivation and with the multiplication rule

$$Z \cdot a = \sigma(a)Z + \delta(a)$$

for all $a \in \mathbb{Z}[t]$. When $\delta = 0$, we call the polynomials *shift polynomials*. For brevity, we will use $\mathbb{Z}[t][Z]$ when the specific choices of σ and δ are not important.

Let $\mathbb{Z}[t][Z]^{m\times n}$ be the ring of $m \times n$ Ore polynomial matrices over $\mathbb{Z}[t]$. We shall adapt the following conventions for the remainder of this paper. Let $\mathbf{F}(Z) \in \mathbb{Z}[t][Z]^{m\times n}$, $N = \deg \mathbf{F}(Z)$, and write

$$\mathbf{F}(Z) = \sum_{j=0}^{N} F^{(j)} Z^j, \text{ with } F^{(j)} \in \mathbb{Z}[t]^{m\times n}.$$

We also write $c_j(\mathbf{F}(Z)) = F^{(j)}$ to denote the j-th coefficient matrix.

An Ore polynomial matrix $\mathbf{F}(Z)$ is said to have *row degree* $\vec{\nu} = \text{rdeg } \mathbf{F}(Z)$ if the i-th row has degree ν_i. The *leading row coefficient* of $\mathbf{F}(Z)$, denoted $\text{LC}_{\text{row}}(\mathbf{F}(Z))$, is the matrix whose entries are the coefficients of Z^N of the corresponding elements of $Z^{N\cdot\vec{e}-\vec{\nu}} \cdot \mathbf{F}(Z)$. An Ore polynomial matrix $\mathbf{F}(Z)$ is *row-reduced* if $\text{LC}_{\text{row}}(\mathbf{F}(Z))$ has maximal row rank. We also recall that the *rank* of $\mathbf{F}(Z)$ is the maximum number of $\mathbb{Q}[t][Z]$-linearly independent rows of $\mathbf{F}(Z)$, and that $\mathbf{U}(Z) \in \mathbb{Z}[t][Z]^{m\times m}$ is *unimodular* if there exists $\mathbf{V}(Z) \in \mathbb{Q}[t][Z]^{m\times m}$ such that $\mathbf{V}(Z) \cdot \mathbf{U}(Z) = \mathbf{U}(Z) \cdot \mathbf{V}(Z) = \mathbf{I}_m$. Some useful properties of matrices of Ore polynomials, such as linear independence and rank, can be found in Ref.[3]

Example 2.1. Consider the differential algebraic system

$$\begin{aligned} y_1''(t) + (t+2)y_1(t) + \quad & y_2''(t) + y_2(t) + \quad y_3'(t) + y_3(t) = 0 \\ y_1''(t) + y_1'(t) + 3y_1(t) + \quad & y_2^{(3)}(t) + 2y_2'(t) - y_2(t) + y_3^{(3)}(t) - 2t^2 y_3(t) = 0 \\ y_1'(t) + y_1(t) + y_2^{(3)}(t) + 2ty_2'(t) - y_2(t) + \quad & y_3^{(4)}(t) = 0. \end{aligned} \tag{1}$$

Let D denote the differential operator on $\mathbb{Q}(t)$ such that $D \cdot f(t) = \frac{d}{dt} f(t)$. Then the matrix form of Eq. (1) is:

$$\begin{bmatrix} D^2 + (t+2) & D^2 + 1 & D + 1 \\ D^2 + D + 3 & D^3 + 2D - 1 & D^3 - 2t^2 \\ D + 1 & D^3 + 2tD + 1 & D^4 \end{bmatrix} \cdot \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix} = \mathbf{0}. \tag{2}$$

The leading row coefficient (matrix of coefficients of the highest power of the corresponding row) is upper triangular. This allows us to rewrite the highest derivative in each row as a combination of other derivatives. For

example, we can eliminate the highest derivatives of $y_2(t)$ as follows:

$$\begin{aligned} y_2^{(3)}(t) &= -y_1''(t) - y_1'(t) - 3y_1(t) - 2y_2'(t) + y_2(t) - y_3^{(3)}(t) + 2t^2 y_3(t) \\ &= -((t+2)y_1(t) - y_2''(t) - y_2(t) - y_3'(t) - y_3(t)) - y_1'(t) - 3y_1(t) \\ &\quad - 2y_2'(t) + y_2(t) - y_3^{(3)}(t) + 2t^2 y_3(t) \\ &= -y_1'(t) - (t+5)y_1(t) + y_2''(t) - 2y_2'(t) + 2y_2(t) - y_3^{(3)}(t) + y_3'(t) \\ &\quad + (2t^2+1)y_3(t). \end{aligned}$$

2.3. *The FFreduce Elimination Algorithm*

We give a brief description of the FFreduce elimination algorithm[3,6] which forms much of the basis of our work. In this algorithm, we are interested in applying the following elementary row operations to the matrix $\mathbf{F}(Z)$:

(a) interchange two rows;
(b) multiply a row by a nonzero element in $\mathbb{Z}[t][Z]$;
(c) add a polynomial multiple of one row to another.

Formally, we may view a sequence of elementary row operations as a *transformation matrix* $\mathbf{U}(Z) \in \mathbb{Z}[t][Z]^{m\times m}$ with the result of these operations given by $\mathbf{T}(Z) = \mathbf{U}(Z) \cdot \mathbf{F}(Z)$. The application of these row operations do not change the rank of $\mathbf{F}(Z)$.[3,6] If the row multiplier of (b) is restricted to elements of $\mathbb{Z}[t]$ then $\mathbf{U}(Z)$ is unimodular. By applying these operations to eliminate low-order coefficients, one can compute the rank and the left nullspace of $\mathbf{F}(Z)$.

The elimination problem can be formalized as follows. An Ore polynomial vector $\mathbf{P}(Z) \in \mathbb{Z}[t][Z]^{1\times m}$ is said to have *order* $\vec{\omega}$ with respect to $\mathbf{F}(Z)$[a] if

$$\mathbf{P}(Z) \cdot \mathbf{F}(Z) = \mathbf{R}(Z) \cdot Z^{\vec{\omega}} \tag{3}$$

for some *residual* $\mathbf{R}(Z)$. The set of all vectors of a particular order $\vec{\omega}$ forms a $\mathbb{Q}[t][Z]$-module. The FFreduce algorithm computes a basis $\mathbf{M}(Z) \in \mathbb{Z}[t][Z]^{m\times m}$ of row degree $\vec{\mu}$ for this module, called an *order basis*, such that

(1) every row, $\mathbf{M}(Z)_{i,*}$, has order $\vec{\omega}$ for all $1 \le i \le m$;

[a]Orders in this paper will be with respect to $\mathbf{F}(Z)$ and it will not be explicitly stated for the remainder of the paper.

(2) the rows of $\mathbf{M}(Z)$ form a basis of the module of all vectors of order $\vec{\omega}$. That is, every $\mathbf{P}(Z) \in \mathbb{Q}[t][Z]^{1\times m}$ of order $\vec{\omega}$ can be written as $\mathbf{P}(Z) = \mathbf{Q}(Z) \cdot \mathbf{M}(Z)$ for some $\mathbf{Q}(Z) \in \mathbb{Q}[t][Z]^{1\times m}$;
(3) the leading column coefficient is normalized. That is, there exists a nonzero $d \in \mathbb{Z}[t]$ such that

$$\mathbf{M}(Z) = d \cdot Z^{\vec{\mu}} + \mathbf{L}(Z)$$

where $\deg \mathbf{L}(Z)_{k,l} \leq \mu_l - 1$.

If $\mathbf{M}(Z)$ is row-reduced, we say that it is a *reduced order basis*. Condition (3) implies that the row degree can be viewed as the number of times each row of $\mathbf{F}(Z)$ has been used as a pivot in the elimination process (see Example 3.2). An order basis of a particular order and degree, if it exists, is unique up to a constant multiple (see Theorem 4.4 in Ref. 3). In the FFreduce algorithm, the order is given as input but the degree of the order basis computed is not known in advance. The final row degree reached depends on the input and a pivoting scheme (also called a *computational path*) to be described later.

Let $\mathbf{R}(Z)$ be the residual corresponding to an order basis $\mathbf{M}(Z)$ of order $\vec{\omega} = \sigma \cdot \vec{e}$, such that

$$\mathbf{M}(Z) \cdot \mathbf{F}(Z) = \mathbf{R}(Z) \cdot Z^{\vec{\omega}} \tag{4}$$

If $\sigma = mN + 1$, then the number of nonzero rows in $\mathbf{R}(Z)$ is rank $\mathbf{F}(Z)$, and the rows in $\mathbf{M}(Z)$ corresponding to the zero rows in the residual gives a basis of the left nullspace of $\mathbf{F}(Z)$.[3,6] The order basis computed has row degree $\vec{\mu}$ such that $\mu_i \leq (mN + 1)n$. For the remainder of this paper, we only consider the order basis problem in Eq. (4) with $\vec{\omega} = \sigma \cdot \vec{e}$.

If $\mathbf{F}(Z)$ is a matrix of shift polynomials, we have $|\vec{\mu}| \leq \sigma \min(m, n)$, $\mu_i \leq \sigma$, and the trailing coefficient of $\mathbf{R}(\mathbf{Z})$ has rank $\mathbf{F}(Z)$ nonzero rows. In fact, the reduction can be terminated as soon as there are "enough" zero rows in $\mathbf{R}(Z)$.[2,3,6] In this case, one may perform row reduction on $\mathbf{F}(Z) \cdot Z^{-N}$ using Z^{-1} as the indeterminate. Reversing the coefficients of $\mathbf{M}(Z^{-1})$ and $\mathbf{R}(Z^{-1})$, we get

$$\mathbf{U}(Z) \cdot \mathbf{A}(Z) = \mathbf{T}(Z) \tag{5}$$

with $\mathbf{U}(Z)$ unimodular and $\mathbf{T}(Z)$ in row-reduced form. We can also choose the pivot rows used in the last n steps of the algorithm to construct $\hat{\mathbf{M}}(Z^{-1})$ and $\hat{\mathbf{R}}(Z^{-1})$ satisfying Eq. (4) with the trailing coefficient of $\hat{\mathbf{R}}(Z^{-1})$ in upper echelon form.[3,6] Reversing the coefficients in this case yields a weak Popov form.

Starting from $\mathbf{M}(Z) = \mathbf{I}_m$ and $\vec{\omega} = \vec{0}$, the FFreduce algorithm computes order bases for increasing $\vec{\omega}$ until the desired order is reached. The recursion formulas to increase the order of an order basis from $\vec{\omega}$ to $\vec{\omega} + \vec{e}_j$ are given by the following theorem (see Theorem 6.1 in Ref. 3):

Theorem 2.1. *Let $\mathbf{M}(Z)$ be an order basis of order $\vec{\omega}$ and degree $\vec{\mu}$, and $r_\ell = c_{\omega_j}\left((\mathbf{M}(\mathbf{Z}) \cdot \mathbf{F}(Z))_{\ell,j}\right)$. If $r_\ell = 0$ for all $\ell = 1, \ldots, m$, then $\mathbf{M}(Z)$ is an order basis of order $\vec{\omega} + \vec{e}_j$ and degree $\vec{\mu}$. Otherwise, we choose a pivot π such that*

$$\pi = \min_{1 \le \ell \le m} \left\{ \ell : r_\ell \neq 0, \mu_\ell = \min_{1 \le j \le m} \{\mu_j : r_j \neq 0\} \right\}, \tag{6}$$

and let $p_\ell = c_{\mu_\ell - 1 + \delta_{\pi,\ell}}\left(\mathbf{M}(Z)_{\pi,\ell}\right)$. Then an order basis $\widetilde{\mathbf{M}}(Z)$ of order $\vec{\omega} + \vec{e}_j$ and degree $\vec{\mu} + \vec{e}_\pi$ can be computed by

$$p_\pi \cdot \widetilde{\mathbf{M}}(Z)_{\ell,*} = r_\pi \cdot \mathbf{M}(Z)_{\ell,*} - r_\ell \cdot \mathbf{M}(Z)_{\pi,*} \quad \text{for } \ell \neq \pi; \tag{7}$$

$$\sigma\left(p_\pi\right) \cdot \widetilde{\mathbf{M}}(Z)_{\pi,*} = \left(r_\pi \cdot Z - \delta\left(r_\pi\right)\right) \cdot \mathbf{M}(Z)_{\pi,*} - \sum_{\ell \neq \pi} \sigma\left(p_\ell\right) \cdot \mathbf{M}(z)_{\ell,*}. \tag{8}$$

When the coefficients of the Ore polynomials come from an integral domain such as $\mathbb{Z}[t]$, no fraction is introduced while applying Eq. (7) and Eq. (8). We also note that the degree of the order basis computed depends on the pivot choices in Eq. (6) and cannot be predicted in advance.

3. Linear Algebra Formulation

Given row degree $\vec{\mu}$ and order $\vec{\omega}$, the coefficients in the order basis $\mathbf{M}(Z)$ can be viewed as a solution to a linear system of equations over the coefficient ring. By equating the coefficients of like powers, each row of the order basis satisfies a system of equations of the form

$$\begin{array}{c} Z^0 \quad \cdots \quad Z^{\mu_k - 1 + \delta_{1,k}} \\ \left[\cdots \;\middle|\; p_k^{(0)} \cdots p_k^{(\mu_k - 1 + \delta_{1,k})} \;\middle|\; \cdots \right] \end{array} \cdot \left[\begin{array}{c|c|c} & \vdots & \\ \hline \cdots & Z^0 \cdot F_{k,\cdot}(Z) & \cdots \\ & \vdots & \\ \cdots & Z^{\mu_k - 1 + \delta_{1,k}} \cdot F_{k,\cdot}(Z) & \cdots \\ \hline & \vdots & \end{array}\right] \begin{array}{l} \\ Z^0 \\ \\ \\ Z^{\vec{\omega} - \vec{e}} \\ \\ \end{array} = \mathbf{0}. \tag{9}$$

More formally, for any $\mathbf{P}(Z) \in \mathbb{Q}[t][Z]^{m \times n}$ we define

$$\mathbf{P}_{\vec{v}} = \left[P_{*,1}^{(0)} \cdots P_{*,1}^{(v_1)} | \cdots | P_{*,n}^{(0)} \cdots P_{*,n}^{(v_n)} \right]. \tag{10}$$

We also define (recall that $\vec{\omega} = \sigma \cdot \vec{e}$)

$$K(\vec{\mu},\vec{\omega}) = \begin{bmatrix} c_0(\ \ \mathbf{F}(Z)_{1,*}) & \cdots & c_{\sigma-1}(\ \ \mathbf{F}(Z)_{1,*}) \\ \vdots & & \vdots \\ c_0(Z^{\mu_1}\cdot\ \mathbf{F}(Z)_{1,*}) & \cdots & c_{\sigma-1}(Z^{\mu_1}\cdot\ \mathbf{F}(Z)_{1,*}) \\ \vdots & & \vdots \\ c_0(\ \ \mathbf{F}(Z)_{m,*}) & \cdots & c_{\sigma-1}(\ \ \mathbf{F}(Z)_{m,*}) \\ \vdots & & \vdots \\ c_0(Z^{\mu_m}\cdot\ \mathbf{F}(Z)_{m,*}) & \cdots & c_{\sigma-1}(Z^{\mu_m}\cdot\ \mathbf{F}(Z)_{m,*}) \end{bmatrix}. \tag{11}$$

Then the i-th row of the order basis satisfies

$$(\mathbf{M}_{i,*})_{\vec{\mu}-\vec{e}+\vec{e}_i} \cdot K(\vec{\mu}-\vec{e}+\vec{e}_i,\vec{\omega}) = \mathbf{0}. \tag{12}$$

The matrix $K(\vec{\mu},\vec{\omega})$ has dimensions $|\vec{\mu}+\vec{e}| \times |\vec{\omega}|$, and is called a *striped Krylov matrix* (with m stripes). This is a generalization of the well-known Sylvester matrix when $m = 2$ and $n = 1$.

Example 3.1. Let $\vec{\mu} = (2,2)$, $\vec{\omega} = (3,3)$, and

$$\mathbf{F}(Z) = \begin{bmatrix} 2Z^2+3tZ+6t^2 & Z^2-Z+2 \\ (t-1)Z+3t^3 & 3tZ+t \end{bmatrix} \in \mathbb{Z}[t][Z;\sigma,\delta]^{2\times 2}, \tag{13}$$

with $\sigma(a(t)) = a(t)$ and $\delta(a(t)) = \frac{d}{dt}a(t)$. Then

$$K(\vec{\mu},\vec{\omega}) = \left[\begin{array}{cc|ccc} 6t^2 & 2 & 3t & -1 & 2 & 1 \\ 12t & 0 & 6t^2+3 & 2 & 3t & -1 \\ 12 & 0 & 24t & 0 & 6t^2+6 & 2 \\ \hline 3t^3 & t & t-1 & 3t & 0 & 0 \\ 9t^2 & 1 & 3t^3+1 & t+3 & t-1 & 3t \\ 18t & 0 & 18t^2 & 2 & 3t^3+2 & t+6 \end{array}\right]. \tag{14}$$

We also define the matrix $K^*(\vec{\mu},\vec{\omega}) = K(\vec{\mu},\vec{\omega})_{*,J}$ where J is the lexicographically smallest set of column indices such that $K^*(\sigma n\cdot\vec{e},\vec{\omega})_{*,J}$ has full column rank (this is called the *rank profile* in Ref. 16). An order basis of degree $\vec{\mu}$ and order $\vec{\omega}$ exists if $K^*(\vec{\mu}-\vec{e},\vec{\omega})$ is nonsingular,[4] and in that case, Eq. (12) has a solution space of dimension one. The system can be transformed into the following system with a unique solution:

$$(\mathbf{M}_{i,*})_{\vec{\mu}-\vec{e}} \cdot K^*(\vec{\mu}-\vec{e},\vec{\omega}) = d\cdot\left[c_0\left(Z^{\mu_i}\cdot\mathbf{F}(Z)_{i,*}\right)\cdots c_{\sigma-1}\left(Z^{\mu_i}\cdot\mathbf{F}(Z)_{i,*}\right)\right] \tag{15}$$

where $d = \pm\det K^*(\vec{\mu}-\vec{e},\vec{\omega})$. In other words, we are interested in the Cramer solution of Eq. (15). Thus, the elements of the solution can be

written as determinants of submatrices of $K(\vec{\mu}, \vec{\omega})$. If $\mathbf{F}(Z) \in \mathbb{Z}[t][Z]^{m \times n}$, then the solution has entries in $\mathbb{Z}[t][Z]$ as well.

The FFreduce algorithm performs fraction-free elimination of the matrix $K(\vec{\mu}, \vec{\omega})$ efficiently by taking advantage of the inherent structure in the matrix $K(\vec{\mu}, \vec{\omega})$. Indeed the algorithm has the effect of maintaining only one row in each of the m stripes. It can be shown that p_π is the pivot in the previous step in computing $\mathbf{M}(Z)$, in a way similar to fraction-free Gaussian elimination of Bareiss.[1] The elements of $\mathbf{M}(Z)$ are Cramer solutions to Eq. (15). The order basis $\mathbf{M}(Z)$ can be viewed as the transformation matrix representing the row operations performed during fraction-free Gaussian elimination.

Example 3.2. Let $\mathbf{F}(Z)$ be defined as in Example 3.1, so that we are performing Gaussian elimination on $K(\cdot, \cdot)$. In the first step of FFreduce, we choose $\pi = 1$ to eliminate the first column so that the row considered in the first stripe is advanced to the second row of the stripe. This gives

$$\mathbf{M}(Z) = \begin{bmatrix} 6t^2 Z - 12t & 0 \\ -3t^3 & 6t^2 \end{bmatrix}. \tag{16}$$

We omit giving the residual explicitly due to coefficient growth. In the second step, we again choose $\pi = 1$ to eliminate the second column. Note that here, the first row in the second stripe is zero in the second column, even though the remaining rows in the stripe have nonzero entries in the column. The choice of pivot depends only on the current row of each stripe. This gives an order basis of order $(1, 1)$ and degree $(2, 0)$:

$$\mathbf{M}(Z) = \begin{bmatrix} -24tZ^2 + 24Z & 0 \\ 12t^2 & -24t \end{bmatrix}. \tag{17}$$

For the third column, the entry of the current row in the second stripe is nonzero, so that we choose $\pi = 2$ as the pivot and obtain an order basis of order $(2, 1)$ and degree $(2, 1)$. Thus, the first three elements of J are 1, 2, and 3. Note that at each step, we are solving a linear system whose coefficient matrix grows column-wise (as the order increases) and row-wise (as the degree of the order bases increases).

As we can see, the row degrees of the order bases computed correspond to the number of times a row in each stripe is used as a pivot in the elimination process. The column indices in J give the columns in which elimination is performed (i.e. there is a nonzero entry in the current row of one of the stripes).

If we perform the same computation in $\mathbb{Z}_3$, we see that the first column is zero and we will reach an order basis of order $(2,1)$ and degree $(1,1)$. Here, the first two elements of J are 2 and 3. On the other hand, performing the computation in $\mathbb{Z}_2$ results in an order basis of order $(2,1)$ and degree $(1,1)$, with the first two elements of J being 1 and 3. This shows that when a prime is "unlucky", the rank profile (and hence the set of column indices J) as well as the sequence of row degrees of order bases constructed (the computational path) can be different from the correct ones.

4. Reduction to $\mathbb{Z}_p[t][Z]$

In this section, we show how techniques used in modular algorithms for polynomial matrices[7] can be extended to Ore polynomial matrices for the reduction of $\mathbb{Z}[t][Z]$ to $\mathbb{Z}_p[t][Z]$. We omit proofs which are similar to the polynomial matrix case and instead only highlight the differences. We refer the reader to Ref. 7 for more details.

Given the Ore polynomial ring $\mathbb{Z}[t][Z;\sigma,\delta]$ and a prime p, we define the modular homomorphism ϕ_p by

$$\phi_p(A) = A \bmod p, \quad \text{for } A \in \mathbb{Z}[t][Z;\sigma,\delta]. \tag{18}$$

This gives a mapping from the Ore polynomial ring $\mathbb{Z}[t][Z;\sigma,\delta]$ to the ring $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$, where

$$\sigma_p(\phi_p(f)) = \phi_p(\sigma(f)) \text{ and } \delta_p(\phi_p(f)) = \phi_p(\delta(f)) \text{ for } f \in \mathbb{Z}[t]\ . \tag{19}$$

We are interested in reducing the computation over $\mathbb{Z}[t][Z;\sigma,\delta]$ to computations over $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$ for a certain number of primes p. The results are then combined using Chinese remaindering to obtain the desired result in $\mathbb{Z}[t][Z;\sigma,\delta]$. Three issues need to be addressed: normalization of the results in $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$, the detection of unlucky primes, and a termination criteria.

The main idea is that the computation of an order basis can be viewed as the computation of solutions to linear systems of the form in Eq. (15). As illustrated in Example 3.2, $r_\ell = 0$ for all $\ell = 1,\ldots,m$ in any one step implies that the corresponding column in the matrix $K(\cdot,\cdot)$ is linearly dependent. Otherwise, the choice of π refers to pivot row used in the elimination. Therefore, in order to be sure that the results computed in $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$ is the same as those computed in $\mathbb{Z}[t][Z;\sigma,\delta]$, we need to ensure that the same systems of equations are solved (i.e. the rows and columns appearing in $K^*(\cdot,\cdot)$ is the same) and that the sign of d in Eq. (15) is chosen consistently across all primes p.

The sign of d can be computed consistently by applying the following formula together with the recursion formulas in Eq. (7) and Eq. (8):

$$\epsilon_0 = 1$$
$$\epsilon_{k+1} = \begin{cases} \epsilon_k & \text{if } r_\ell = 0 \text{ for all } \ell, \\ \epsilon_k \cdot (-1)^{\sum_{i=\pi_{k+1}}^{m} (\mu_k)_i} & \text{otherwise,} \end{cases} \tag{20}$$

where π_k and $\vec{\mu}_k$ are the values of π and $\vec{\mu}$ at the k-th step of the algorithm (Lemma 5.1(c) in Ref. 4). Multiplying the results obtained using Eq. (7) and Eq. (8) by ϵ_k at each step, this normalization ensures that $d = \det K^*(\vec{\mu} - \vec{e}, \vec{\omega})$. If the results in $\mathbb{Z}_p[t][Z; \sigma_p, \delta_p]$ are computed by some other means (see Sec. 5), we simply need to ensure that the sign is consistent with Eq. (20).

4.1. *Lucky Homomorphisms*

Let $\operatorname{hc}(f)$ be the leading coefficient of $f \in Z[t][Z; \sigma, \delta]$ in t (the head coefficient). If $\phi_p(\operatorname{hc}(\sigma(t))) \neq 0$, then σ_p is an Ore ring homomorphism and $\mathbb{Z}_p[t][Z; \sigma_p, \delta_p]$ is an Ore polynomial ring.[11,13] Thus, all primes p such that $\phi_p(\operatorname{hc}(\sigma(t))) = 0$ are immediately discarded.

Let $\vec{\mu}$ and J be the degrees and index set computed for an order basis of order $\vec{\omega}$ in $\mathbb{Z}[t][Z; \sigma, \delta]$, and $\vec{\mu}_p$ and J_p be those computed in $\mathbb{Z}_p[t][Z; \sigma_p, \delta_p]$. Also, let d be the constant in Eq. (15) computed in $\mathbb{Z}[t][Z; \sigma, \delta]$. To ensure that the same systems of equations is solved in Eq. (15), we must ensure that $(\vec{\mu}, J) = (\vec{\mu}_p, J_p)$. This can be defined formally as follows.

Definition 4.1. The homomorphism ϕ_p is *lucky* if

(1) $\phi_p(d) \neq 0$;
(2) $|\vec{\mu}| = |\vec{\mu}_p|$; and
(3) $\phi_p(\operatorname{hc}(\sigma(t))) \neq 0$.

Otherwise, it is *unlucky*.

It can be shown that if ϕ_p is lucky, then $(\vec{\mu}, J) = (\vec{\mu}_p, J_p)$.[7] In that case, the order basis and residual computed in $\mathbb{Z}_p[t][Z; \sigma_p, \delta_p]$ are images of the desired results under ϕ_p. Moreover, if p is unlucky then it must divide $\operatorname{hc}(\sigma(t))$ or a minor of $K(\vec{\mu}, \vec{\omega})$, so the number of unlucky homomorphisms is finite.[7] More precisely, if κ is a bound on the coefficients of $Z^\ell \cdot \mathbf{F}(i, j)$ for $0 \leq \ell \leq mN + 1$, then the number of unlucky homomorphisms is at most

$$\max\left(\log_2 \operatorname{hc}(\sigma(t)), (mN+1)^2 n^2 \log_2\left(\kappa\sqrt{(mN+1)n}\right)\right) \tag{21}$$

as we are only interested in minors of size up to $(mN+1)n$. In practice, however, this is a very pessimistic estimate and unlucky homomorphisms are rarely encountered.

We note that the sequence of row degrees of the order bases constructed during the FFreduce algorithm represents the choice of pivots (see Example 3.2). We call this sequence a *computational path*. If we define the path $W = \{\vec{w}_k\}_{k=0,1,2,\ldots}$ by $\vec{w}_0 = \vec{0}$ and $\vec{w}_{k+1} = \vec{w}_k + \vec{e}_{k \bmod m+1}$, then W is the sequence of degrees followed by the FFreduce algorithm if $r_\ell \neq 0$ for all ℓ at every step. It was shown that the final degree $\vec{\mu}$ is the unique closest *normal point* to w (see Theorem 7.3 in Ref. 4). That is, if $K^*(\vec{v}-\vec{e},\vec{\omega})$ is nonsingular for some $\vec{v}$ such that $|\vec{v}| = |\vec{\mu}|$, then

$$|\max(\vec{0}, \vec{w}_k - \vec{\mu})| \leq |\max(\vec{0}, \vec{w}_k - \vec{v})| \text{ for all } k \geq 0. \tag{22}$$

Since $\vec{\mu}$ and J are not known a priori, we need criteria to compare the results computed under two homomorphisms and determine if one of them is unlucky. This is similar to the case for polynomial matrices.[7] This allows one to incrementally compute homomorphic images and detect unlucky homomorphisms by comparing the current image against the previous ones.

Theorem 4.1. *Suppose $\phi_p(hc(\sigma(t))) \neq 0$, $\phi_p(F^{(0)}) \neq \mathbf{0}$, and $\phi_q(F^{(0)}) \neq \mathbf{0}$. Then ϕ_p is unlucky if one of the following holds:*

(1) $|\vec{\mu}_p| = |\vec{\mu}_q|$ and $J_p >_{\text{lex}} J_q$;
(2) $|\vec{\mu}_p| = |\vec{\mu}_q|$, $J_p = J_q$, and $\vec{\mu}_q$ is closer to W than $\vec{\mu}_p$;
(3) $|\vec{\mu}_p| < |\vec{\mu}_q|$.

We remark that this theorem is used only to detect unlucky homomorphisms and cannot be used to detect lucky ones.

Assuming that we can compute the order basis $\mathbf{M}_p(Z)$ (and the corresponding row degree $\vec{\mu}_p$ and column index set J_p) for $\phi_p(F(Z))$ over $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$, Theorem 4.1 allows the results computed under two different homomorphisms be compared to detect unlucky homomorphisms.

4.2. *Termination*

As all coefficients in $\mathbf{M}(Z)$ are Cramer solutions of the linear systems in Eq. (15), they can be written as determinants of the coefficient matrices.[4] Using Hadamard's inequality, we can bound the size of the coefficients and terminate the modular algorithm when the product of the moduli exceeds this bound. Unfortunately, the Hadamard bound can be extremely pessimistic. A common approach is to reconstruct the results incrementally one

prime at a time using Garner's algorithm,[8] and verify the results when they do not change for a certain number (e.g. one) of additional homomorphisms. The verification step typically involves trial division or multiplication.

An approach of Cabay[5] was first used for solving systems of linear equations with a modular algorithm without the need for verification. This was extended to the case of polynomial matrix normal form computations.[7] This technique can be extended to Ore polynomial matrices.

Theorem 4.2. *Let* $\|f(t)\|_\infty = \max_i |f^{(i)}|$ *where* $f(t) = \sum_i f^{(i)} t^i$. *Suppose that for all* $1 \leq i \leq m$, $1 \leq j \leq n$, $0 \leq k < mN+1$, *and* $0 \leq \ell \leq mN+1$, *we have*

$$deg_t \left(c_k \left(Z^\ell \cdot \mathbf{F}(Z)_{i,j} \right) \right) \leq T; \tag{23}$$

$$\left\| c_k \left(Z^\ell \cdot \mathbf{F}(Z)_{i,j} \right) \right\|_\infty \leq \kappa. \tag{24}$$

Suppose the primes are ordered such that $p_1 < p_2 < \cdots$, *and that*

$$((mN+1)n)\kappa(T+1) \leq p_1 \cdots p_\tau \tag{25}$$

for all $j = 1, \ldots, n$. *If* $\widetilde{\mathbf{M}}(Z)$ *and* $\widetilde{\mathbf{R}}(Z)$ *are the reconstructed results in the modular algorithm and have not changed for* τ *additional primes, then* $\widetilde{\mathbf{M}}(Z)$ *and* $\widetilde{\mathbf{R}}(Z)$ *give a solution to Eq. (4).*

Proof. The proof of this theorem follows that of Theorem 6.1 in Ref. 7. We note that the coefficients of $\mathbf{M}(Z)$ are solutions to the linear systems of equations in Eq. (15), where the size of the matrix $K^*(\vec{\mu} - \vec{e}, \vec{\omega})$ is $(mN+1)n$. The only difference here is that we need to note that by Hadamard's inequality, $\deg_t(f) \leq (mN+1)nT$ where f is a coefficient in $\mathbf{M}(Z)$, and that $\|fg\|_\infty \leq (\min(\deg_t(f), \deg_t(g)) + 1)\|f\|_\infty \|g\|_\infty$. □

The early termination criteria is most useful if τ is small. In particular, if $\tau = 1$ then the proposed criteria is clearly an improvement over a traditional modular algorithm. This is often true in practical cases.[7] In the worst case, the termination condition of Eq. (25) is satisfied when the traditional Hadamard bound is reached. Thus, the early termination strategy is no worse than the traditional one.

Remark 4.1. The bound in Eq. (25) is based on the norm of a column in $K^*(\vec{\mu} - \vec{e}, \vec{\omega})$. In specific cases where σ and δ are known, the bound can be refined.

5. Reduction to $\mathbb{Z}_p$

While we do not suffer coefficient growth in the size of the integer coefficients when solving the problems in $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$, the growth of the degrees of the coefficients with respect to t still needs to be controlled. In this section, we show how the computation of an order basis in $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$ can be transformed into a number of linear algebra problems over $\mathbb{Z}_p$.

It has been shown that the evaluation homomorphisms $t \leftarrow \alpha$ are usually not Ore ring homomorphisms because Ore polynomial rings over $\mathbb{Z}_p$ must be commutative.[11,13] As a result, it is not possible to apply the same technique in the previous section to obtain a modular algorithm to compute order bases in $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$ by reducing the problem into ones with coefficients in $\mathbb{Z}_p[Z]$. However, when the problem in $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$ is viewed as the linear algebra problem in Eq. (15) over $\mathbb{Z}_p[t]$, each entry in the striped Krylov matrix can be reduced by an evaluation homomorphism to obtain a number of linear algebra problems in $\mathbb{Z}_p$. This is essentially the approach taken to compute a GCRD and a LCLM of Ore polynomials[11,13]—the image of the coefficient matrix over $\mathbb{Z}_p$ is constructed and Gaussian elimination is performed on this matrix over $\mathbb{Z}_p$.

In the Ore polynomial case, the coefficient matrix in Eq. (15) is the well-known Sylvester matrix and the computational path corresponds directly to the degree sequence of the polynomial remainder sequence.[8] In the Ore polynomial matrix case, however, we do not know of such a correspondence and neither the computational path nor the final configuration of the striped Krylov matrix is known a priori. Thus, it is not sufficient to simply construct the coefficient matrix and then operate on it.

Our modular algorithm to compute order bases in $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$ consists of the following steps:

(1) Choose a number of evaluation points α.
(2) For each evaluation point, apply the evaluation homomorphism $t \leftarrow \alpha$ and solve the reduced linear algebra problem over $\mathbb{Z}_p$.
(3) Combine the results obtained from lucky evaluation points by polynomial interpolation.

This is the traditional framework for modular algorithm. However, the application of the evaluation homomorphism and solution to the linear algebra problem is non-trivial.

5.1. *Applying Evaluation Homomorphisms and Computation in $\mathbb{Z}_p$*

Our approach can be seen as a generalization of the modular algorithm for Ore polynomials.[11,13] Instead of constructing the entire coefficient matrix $K(\cdot, \vec{\omega})$ in $\mathbb{Z}_p$ and performing Gaussian elimination on this matrix, we will incrementally construct the striped Krylov matrix. Whenever we choose the pivot π in an elimination step, the row corresponding to $Z^{\mu_\pi+1} \cdot \mathbf{F}(Z)_{\pi,*}$ needs to be added to the striped Krylov matrix. Once the row is added, elimination must be performed so that the new row also has the required order. However, this is not completely straightforward as the next example shows.

Example 5.1. Let $\mathbf{F}(Z)$ be defined as in Example 3.1. Applying the evaluation $t \leftarrow 0$, the image of the striped Krylov matrix $K(\vec{\mu}, \vec{\omega})$ over $\mathbb{Z}_5$ is

$$K = \left[\begin{array}{cc|cc|cc} 0 & 2 & 0 & 4 & 2 & 1 \\ 0 & 0 & 3 & 2 & 0 & 4 \\ 2 & 0 & 0 & 0 & 1 & 2 \\ \hline 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 1 & 1 & 3 & 4 & 0 \\ 0 & 0 & 0 & 2 & 2 & 1 \end{array}\right] . \tag{26}$$

When the third row in the first stripe is finally added to the matrix, we notice that the first column is linearly independent. In fact, there is no need to perform elimination on the added row.

The sequences of pivot rows and columns over $\mathbb{Z}_5$ is different from those over $\mathbb{Z}_5[t]$, but the set of pivot rows and columns used at the end are the same in this case as K has full rank. After appropriate normalization, the solutions computed over $\mathbb{Z}_5$ are images of the corresponding solutions in $\mathbb{Z}_5[t]$.

We now describe the process by which the elimination is performed. The computation of quantities such as $\vec{\mu}$ and π are similar to the FFreduce algorithm; they will not be given explicitly here. We will keep the striped Krylov matrix K in the form of matrix of coefficients in $\mathbb{Z}_p$. The entries in the transformation matrix $\mathbf{T}(Z)$ will be kept in polynomial form to simplify the manipulations required when a new row is added. In addition, we maintain a matrix $\mathbf{C}(Z)$ where $\mathbf{C}(Z)_{i,*} = Z^{\mu_i} \cdot \mathbf{F}(Z)_{i,*} \in \mathbb{Z}_p[t][Z; \sigma_p, \delta_p]^{1\times n}$. The vector $\mathbf{C}(Z)_{i,*}$ represents the last row of stripe i added to K and allows us to quickly add the next row.

The algorithm proceeds as follows.

(1) Initially, set $\mathbf{C}(Z) = \mathbf{F}(Z)$. Also initialize K to include the m rows consisting of coefficients of $\mathbf{F}(Z)_{i,*}$ up to order $\vec{\omega}$ evaluated at $t = \alpha$, and a transformation matrix $\mathbf{T}(Z) = I_m$.
(2) For each elimination step, the residuals r_i in Theorem 2.1 can be obtained by examining the appropriate row and column in K based on $\vec{\mu}$ and σ and the form of the striped Krylov matrix in Eq. (11). If $r_i \neq 0$ for some $1 \leq i \leq m$, add the step number j to set of column indices J. Once the pivot row is chosen, we apply standard row operations in $\mathbb{Z}_p$ (non-fraction-free) to eliminate all other rows of K for that column. We also need to add the next row for stripe π:
   (a) $\mathbf{C}(Z)_{\pi,*} \leftarrow Z \cdot \mathbf{C}(Z)_{\pi,*}$. This operation is performed in $\mathbb{Z}_p[t][Z; \sigma_p, \delta_p]$.
   (b) Evaluate $\mathbf{C}(Z)_{\pi,*}$ at $t = \alpha$. Add the coefficients as a new row to $\mathbf{K}$, and perform row interchanges so that the stripes in $\mathbf{K}$ are in the form in Eq. (11). Add a corresponding row to $\mathbf{T}(Z)$, whose only nonzero entry is $Z^{\mu_\pi+1}$ in column π.
   (c) Perform row operations to eliminate the added row up to column j using all rows that have previously been used as pivots. The same row operations are applied to $\mathbf{T}(Z)$. During the reduction, if the added row introduces a new linearly independent column, add the column index to J and repeat steps 2a to 2c.
(3) When the row reductions are complete, let K^* be the triangular submatrix of $K_{*,J}$ consisting of all but the last added row for each stripe. Compute the determinant $d = \det K^*$ as a product of the diagonal elements. Adjust the sign of d based on the row interchanges performed (see Eq. (20)).
(4) The i-th row of the order basis $\mathbf{M}(Z)$ and the residual $\mathbf{R}(Z)$ can be extracted from the rows corresponding to $Z^{\mu_i} \cdot \mathbf{F}(Z)_{i,*}$ in $d \cdot \mathbf{T}(Z)$ and $d \cdot K$.

Although we have lost the ability of FFreduce to take advantage of the structure of the striped Krylov matrix, we gained the ability to control coefficient growth. Note that coefficient growth is not completely eliminated, since the computation of $\mathbf{C}(Z)$ may introduce growth in the degree in t when $Z^k \cdot \mathbf{F}(Z)$. However, the growth arising from Gaussian elimination is eliminated. Furthermore, the degree in t does not grow when multiplying by Z in many practical cases (see Remark 6.3).

Example 5.2. Continuing from Example 5.1, we first start with

$$K = \left[\begin{array}{cc|cc|cc} 0 & 2 & 0 & 4 & 2 & 1 \\ 0 & 0 & 4 & 0 & 0 & 0 \end{array}\right]. \tag{27}$$

Eliminating the second column with pivot $\pi = 1$ requires no row operation. The next row for the first stripe is added to obtain

$$K = \left[\begin{array}{cc|cc|cc} 0 & 2 & 0 & 4 & 2 & 1 \\ 0 & 0 & 3 & 2 & 0 & 4 \\ \hline 0 & 0 & 4 & 0 & 0 & 0 \end{array}\right]. \tag{28}$$

In the next step, we choose $\pi = 2$ to eliminate the third column. Adding the next row and eliminating gives

$$K = \left[\begin{array}{cc|cc|cc} 0 & 2 & 0 & 4 & 2 & 1 \\ 0 & 0 & 0 & 2 & 0 & 4 \\ \hline 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 2 \end{array}\right]. \tag{29}$$

Next, we choose $\pi = 1$ to eliminate the fourth column. After adding the next row for the first stripe, we get:

$$K = \left[\begin{array}{cc|cc|cc} 0 & 2 & 0 & 4 & 2 & 1 \\ 0 & 0 & 0 & 2 & 0 & 4 \\ 2 & 0 & 0 & 0 & 1 & 2 \\ \hline 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{array}\right]. \tag{30}$$

The new row introduces a new linearly independent column. The next row for the first stripe will be added immediately. Continuing in this manner gives the coefficients of the residual $\mathbf{R}(Z)$ as the last row of each stripe.

5.2. *Lucky Homomorphisms and Termination*

Since our definition of lucky homomorphisms is originally based on the linear algebra formulation of the order basis problem, the same definition can be easily applied to the reduction of the linear algebra problem from $\mathbb{Z}_p[t]$ to $\mathbb{Z}_p$. In particular, Definition 4.1 and Theorem 4.1 can be applied simply by changing ϕ_p and ϕ_q to the appropriate evaluation homomorphisms.

Similarly, the termination criteria is also originally based on the linear algebra formulation in Eq. (15). Again, we may apply Hadamard's inequality to obtain bounds on the size (degrees in t) of the coefficients in the solutions, but we prefer to have an early termination condition that is sensitive to the size of the output. Since we are now dealing with linear systems

of equations, we can apply the technique of Cabay[5] directly after modifying the theorem to use the degree measure as the coefficient norm.

Theorem 5.1. *Suppose that for all* $1 \leq i \leq m$, $1 \leq j \leq n$, $0 \leq k < mN+1$, *and* $0 \leq \ell \leq mN+1$, *we have*

$$deg_t\left(c_k\left(Z^\ell \cdot \mathbf{F}(Z)_{i,j}\right)\right) \leq T. \tag{31}$$

Suppose that $\widetilde{\mathbf{M}}(Z)$ *and* $\widetilde{\mathbf{R}}(Z)$ *are the reconstructed results in the modular algorithm and have not changed for* T *additional evaluation points. Then* $\widetilde{\mathbf{M}}(Z)$ *and* $\widetilde{\mathbf{R}}(Z)$ *give a solution to Eq. (4) in* $\mathbb{Z}_p[t][Z;\sigma,\delta]$.

6. Complexity Analysis

To compare the new modular algorithm against the fraction-free FFreduce algorithm, we need to give the complexity of FFreduce in our context, as the analysis in our previous work[3,6] are for general coefficient domains and can be refined. The proof is similar to the ones in our previous work and is omitted.

Theorem 6.1. *Suppose that for all* $1 \leq i \leq m$, $1 \leq j \leq n$, $0 \leq k < mN+1$, *and* $0 \leq \ell \leq mN+1$, *we have*

$$deg_t\left(c_k\left(Z^\ell \cdot \mathbf{F}(Z)_{i,j}\right)\right) \leq T; \tag{32}$$

$$\left\|c_k\left(Z^\ell \cdot \mathbf{F}(Z)_{i,j}\right)\right\|_\infty \leq \kappa. \tag{33}$$

Then an order basis and a residual of order $\vec{\omega} = (mN+1)\cdot\vec{e}$ *over* $\mathbb{Z}[t][Z;\sigma,\delta]$ *can be computed in* $O((m+n)(mnN)^4T^2\mathbf{M}(mnN(\log(T\kappa))))$ *bit operations by the FFreduce algorithm, where* $O(\mathbf{M}(k))$ *is the complexity of multiply two* k*-bit integers.*

We now analyze the complexity of our algorithm. We first examine the complexity of the computation of the order basis over $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$.

Theorem 6.2. *Let* T *be a bound on the degree of the coefficients in* t, *such that*

$$deg_t\left(c_k\left(Z^\ell \cdot \mathbf{F}(Z)_{i,j}\right)\right) \leq T \tag{34}$$

for $1 \leq i \leq m$, $1 \leq j \leq n$, $0 \leq k < mN+1$, *and* $0 \leq \ell \leq mN+1$. *We also assume that two polynomials in* $\mathbb{Z}_p[t]$ *of degree* d *can be multiplied in* $O(d\log d)$ *operations in* $\mathbb{Z}_p$. *Then an order basis and a residual of order* $\vec{\omega} = (mN+1)\cdot\vec{e}$ *over* $\mathbb{Z}_p[t][Z;\sigma_p,\delta_p]$ *can be computed in* $O((mnN)^3mT(nN + T(\log mnNT)^2))$ *operations in* $\mathbb{Z}_p$.

Proof. Since the dimensions of the coefficient matrix in the system of equations in Eq. (15) is $(mN+1)n \times (mN+1)n$, it follows that the entries of the order basis and the residual have degrees in t bounded by $(mN+1)nT$ as they can be written as determinants of submatrices of the coefficient matrix. This implies that $O(mnNT)$ lucky evaluation points are needed. For each evaluation point, the evaluation can be done in $O((mnN)^2T)$ operations in $\mathbb{Z}_p$. The elimination can be done in $O((mnN)^3)$ operations. Finally, each coefficient can be interpolated in $O(\mathbf{PM}(mnNT)\log(mnNT))$ operations,[17] and we have $O(m^2nN)$ nonzero coefficients in $\mathbf{M}(Z)$ and $O(mnN)$ nonzero coefficients in $\mathbf{R}(Z)$, where $\mathbf{PM}(d)$ is the complexity for multiplying two degree d polynomials. The desired result now follows from the assumption on polynomial multiplication. □

Remark 6.1. In the analysis we have ignored the occurrences of unlucky homomorphisms as they rarely occur in practice. Also, we assumed that p is chosen large enough such that polynomial multiplication for degree d polynomials can be performed in $O(d \log d)$ operations.

We are now ready to give the complexity of the complete algorithm.

Theorem 6.3. *Suppose that for all* $1 \le i \le m$, $1 \le j \le n$, $0 \le k < mN+1$, *and* $0 \le \ell \le mN+1$, *we have*

$$deg_t\left(c_k\left(Z^\ell \cdot \mathbf{F}(Z)_{i,j}\right)\right) \le T; \tag{35}$$

$$\left\|c_k\left(Z^\ell \cdot \mathbf{F}(Z)_{i,j}\right)\right\|_\infty \le \kappa. \tag{36}$$

Then an order basis and a residual of order $\vec{\omega} = (mN+1)\cdot\vec{e}$ *over* $\mathbb{Z}[t][Z;\sigma,\delta]$ *can be computed in*

$$O(m(mnN)^3T(nN + T(\log mnNT)^2 + \log\log T\kappa)\mathbf{M}(mnN\log T\kappa))$$

bit operations by our modular algorithm.

Proof. By Hadamard's inequality, Eq. (15), and the inequality $\|fg\|_\infty \le (\min(\deg_t(f),\deg_t(g))+1)T\|f\|_\infty\|g\|_\infty$, we see that $\left\|c_k\left(\mathbf{M}(\mathbf{Z})_{i,j}\right)\right\|_\infty$ and $\left\|c_k\left(\mathbf{R}(\mathbf{Z})_{i,j}\right)\right\|_\infty$ have $O(mnN\log T\kappa))$ bit length. Choosing primes of size $O(\log T\kappa)$, it follows that $O(mnN)$ lucky primes are needed. For each prime, the reduction modulo p can be done in $O(mnNT)$ operations in $\mathbb{Z}_p$, and the computation of the order basis and residual in $\mathbb{Z}_p[t][Z;\sigma,\delta]$ can be done in $O((mnN)^3mT(nN+T(\log mnNT)^2))$ operations in $\mathbb{Z}_p$ by Theorem 6.2. We note that each operation can be performed in $O(\mathbf{M}(\log T\kappa))$ bit operations.

Finally, there are $O(m^2nN)$ nonzero coefficients in $\mathbf{M}(Z)$ and $O(mnN)$ nonzero coefficients in $\mathbf{R}(Z)$, each of which has degree $O(mnNT)$ in t. Each coefficient can be reconstructed in $O(\mathsf{M}(mnN \log T\kappa) \log(mnN \log T\kappa))$ bit operations by Chinese remaindering.[17] □

Remark 6.2. We see that the complexity of the modular algorithm improves on the complexity of the fraction-free algorithm, but the advantage of the reduced coefficient growth is offset by the use of a larger striped Krylov matrix. In the analysis we have used the worst case bound that the number of rows in the striped Krylov matrix can be a factor of nN greater than that of $\mathbf{R}(Z)$ and $\mathbf{M}(Z)$ in Ore polynomial matrix form. In practice, however, the striped Krylov matrix usually never grows to its full size because there can be many zero rows, or because we can terminate the elimination earlier in the case of matrices of shift polynomials. The advantage of the modular algorithm is more significant in these cases.

Remark 6.3. In many practical applications, $\deg_t(\sigma(a)), \deg_t(\delta(a)) \leq \deg_t(a)$ for all $a \in \mathbb{Z}[t]$. In these cases, one can simplify Eq. (35) to

$$\deg_t \left(c_k \left(\mathbf{F}(Z)_{i,j}\right)\right) \leq T. \tag{37}$$

7. Implementation Considerations and Experimental Results

The modular algorithm has been implemented in Maple 9.5. Although the modular algorithm has a better complexity than the fraction-free algorithm, the modular algorithm has a larger overhead especially for small inputs. As a result, a careful implementation is needed for the modular algorithm to perform better than the fraction-free algorithm on inputs of reasonable size. We list below some optimizations used. Although they are straightforward, these optimizations have significant effect on the running time of the algorithm.

- Memory allocation and deallocation can be a significant overhead in the algorithm. To minimize memory management, we allocate one large matrix K for each prime p and use the same matrix for different evaluation points. There is also no need to reallocate the matrix when a new row is added, as long as the original matrix is large enough. In addition, we may augment the matrix K with the identity matrix to represent the coefficients of the order basis.

- When a new row is added to K, we do not perform row interchanges to maintain the coefficient matrix K in striped Krylov matrix form. Instead, we simply add the new row to the bottom and maintain a list of row indices to refer to the correct row. This reduces data movements in the algorithm.
- The computations on $\mathbf{C}(Z)$ for keeping track of the next row to be added to K are identical for all evaluation points α under $\mathbb{Z}_p$. Therefore, we compute the sequence of $\mathbf{C}(Z)$ only once for each prime p and reuse the results for all evaluation points.
- The `LinearAlgebra:-Modular` package is used to efficiently perform linear algebra operations in $\mathbb{Z}_p$.

We compare the performance of the modular algorithm and the fraction-free algorithm (FFreduce) below. The experiments were performed on a computer with a Xeon 2.70GHz processor and 16GB of RAM. In the experiments, we generated random matrices of Ore polynomials such that $\sigma(a(t)) = a(t)$ and $\delta(a(t)) = \frac{d}{dt}a(t)$. The results are given in Table 1 and Table 2. As expected, we see that the modular algorithm performs better than the fraction-free algorithm as the input size increases. Also, we show that using rational arithmetic in $\mathbb{Q}[t]$ to perform row reduction is impractical.

Table 1. Comparison between modular and fraction-free algorithms for various input sizes ($\kappa = 5$, $T = 1$). Also shown are timings when the rational arithmetic in $\mathbb{Q}[t]$ is used (n/a means no result is computed after 7200 seconds).

m, n	N	FFreduce (s)	Modular (s)	Ratio	Rational (s)
2	1	0.037	0.178	0.208	3.073
2	2	0.089	0.278	0.320	n/a
2	4	1.453	3.226	0.449	n/a
2	8	21.984	26.789	0.821	n/a
2	16	75.048	118.796	0.630	n/a
3	1	0.564	1.150	0.490	n/a
3	2	3.928	6.032	0.652	n/a
3	4	64.498	55.991	1.150	n/a
3	8	401.708	339.383	1.190	n/a
4	2	54.213	46.776	1.160	n/a
4	4	1018.963	589.687	1.730	n/a
4	6	5107.881	3123.530	1.640	n/a
5	2	564.498	289.937	1.940	n/a
5	4	7770.646	4554.689	1.710	n/a
8	1	2434.662	1182.519	2.060	n/a
10	1	17124.143	11146.557	1.540	n/a

Table 2. Comparison between modular and fraction-free algorithms for various input sizes ($\kappa = 5$, $T = 2$).

m, n	N	FFreduce (s)	Modular (s)	Ratio
2	2	0.496	1.848	0.268
2	4	5.618	11.368	0.493
2	8	95.759	111.925	0.855
2	16	1723.170	1709.175	1.010
3	2	16.488	24.198	0.682
3	4	330.956	291.985	1.130
3	6	1373.799	1225.103	1.110

Although the modular algorithms is faster than the fraction-free algorithm for larger inputs, there were instances where the ratio between the running times of the fraction-free algorithm and the modular algorithm decreases slightly when the input size is increased. In these cases, the larger inputs actually gave output that are smaller than expected because the rank of the striped Krylov matrix is not full. Hence, the recurrence formulas in Eq. (7) and Eq. (8) are not applied as often, resulting in less coefficient growth in the output. Also, for larger inputs the overhead of garbage collection for the results computed under different evaluation points and different primes become more important. This is confirmed by observing that the running time for the modular algorithm improves significantly when the frequency of garbage collection is reduced[b]. A decision has to be made based on available memory in order to reduce running time while using a reasonable amount of memory.

8. Concluding Remarks

In this paper, we showed how to design an output-sensitive modular algorithm for performing row reductions on matrices of Ore polynomials. By examining the problem as a linear algebra problem in $\mathbb{Z}_p[t]$ and subsequently in $\mathbb{Z}_p$, we overcome the various issues in designing a modular algorithm—detection of unlucky homomorphisms, normalization, and termination. We have also shown that the modular algorithm is faster than the fraction-free algorithm for larger inputs both theoretically and experimentally.

A limitation in both the fraction-free and modular algorithms is that row reduction is performed on the low order terms. When the elements of

[b]Changing the kernel variable `gcfreq` to 5000000 from the default value of 1000000 reduces the running times by as much as 3 times in some cases.

the input matrix are shift polynomials, one may perform substitution to reverse the coefficients. However, this cannot be done for general Ore polynomials. We believe that it is possible to formulate row reduction based on high order terms as a nullspace computation, so that the modular algorithm described here can be applied to control coefficient growth for the computation of normal forms that are defined by leading coefficients such as the row-reduced form.

References

1. E. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comp.*, 22:565–578, 1968.
2. B. Beckermann, H. Cheng, and G. Labahn. Fraction-free row reduction of matrices of skew polynomials. In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 8–15. ACM, 2002.
3. B. Beckermann, H. Cheng, and G. Labahn. Fraction-free row reduction of matrices of Ore polynomials. *Journal of Symbolic Computation*, 41(5):513–543, 2006.
4. B. Beckermann and G. Labahn. Fraction-free computation of matrix rational interpolants and matrix GCDs. *SIAM J. Matrix Anal. and Appl.*, 22(1):114–144, 2000.
5. S. Cabay. Exact solution of linear equations. In *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, pages 392–398, 1971.
6. H. Cheng. *Algorithms for Normal Forms for Matrices of Polynomials and Ore Polynomials*. PhD thesis, University of Waterloo, 2003.
7. H. Cheng and G. Labahn. Output-sensitive modular algorithms for polynomial matrix normal forms. *To appear in Journal of Symbolic Computation*.
8. K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1992.
9. M. Giesbrecht, G. Labahn, and Y. Zhang. Computing valuation popov forms. In *Workshop on Computer Algebra Systems and their Applications (CASA'05)*, 2005.
10. T. Kailath. *Linear Systems*. Prentice-Hall, 1980.
11. Z. Li. *A Subresultant Theory for Linear Differential, Linear Difference and Ore Polynomials, with Applications*. PhD thesis, RISC-Linz, Johannes Kepler University, Linz, Austria, 1996.
12. Z. Li. A subresultant theory for ore polynomials with applications. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, pages 132–139. ACM, 1998.
13. Z. Li and I. Nemes. A modular algorithm for computing greatest common right divisors of ore polynomials. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 282–289. ACM, 1997.
14. T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, 35(4):377–401, 2003.

15. O. Ore. Theory of non-commutative polynomials. *Annals of Mathematics*, 34:480–508, 1933.
16. A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, Department of Computer Science, Swiss Federal Institute of Technology—ETH, 2000.
17. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition, 2002.