A Practical Implementation of a Modular Algorithm for Ore Polynomial Matrices

Howard Cheng[1*] and George Labahn[2]

[1] Department of Mathematics and Computer Science
University of Lethbridge, Lethbridge, Canada
howard.cheng@uleth.ca

[2] Symbolic Computation Group
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Canada
glabahn@uwaterloo.ca

....

Abstract

We briefly review a modular algorithm to perform row reduction of a matrix of Ore polynomials with coefficients in $\mathbb{Z}[t]$, and describe a practical implementation in Maple that improves over previous modular and fraction-free versions. The algorithm can be used for finding the rank, left nullspace, and the Popov form of such matrices.

1 Introduction

Ore domains provide a general setting for describing the arithmetic of linear differential, difference, and q-difference operators. Systems of differential, difference and q-difference equations can then be defined via matrices of Ore operators (polynomials) evaluated at unknown functions. One can then make use of matrix constructions to investigate such systems. For example, performing row reduction on a matrix of Ore polynomials to simpler forms allows one to determine its rank and left nullspace which give the minimum number of equations needed to represent the system of equations [1]. When a transformation is invertible, then the normal form 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), then the normal form allows one to rewrite high-order operators (e.g. derivatives) in terms of lower ones [3]. These transformations can also be applied to the computation of greatest common right divisors (GCRDs) and least common left multiples (LCLMs) [2, 7, 8, 9], which represents the intersection and the union of the solution spaces of systems of equations.

The FFreduce algorithm [2] is a procedure for row reducing a matrix of Ore operators which performs row operations in a fraction-free way to reduce to simpler form while still controlling coefficient growth. This algorithm computes the rank and left nullspace of these matrices, and can be used to compute the row-reduced and weak Popov forms of shift polynomial matrices [2], as well as the Popov form of general Ore polynomial matrices [4]. It can also be used to compute a greatest common right divisor (GCRD) and a least common left multiple (LCLM) of such matrices. Besides their general use with systems of equations, LCLMs are also used in nonlinear control theory in order to define the notion of transfer function in some cases [6].

[*] Correspondence to: University of Lethbridge, 4401 University Drive, Lethbridge, Alberta, T1K 3M4, Canada.

A modular version of the FFreduce algorithm was developed by the authors to reduce the computational complexity [3]. In the modular algorithm, it was observed that the evaluation reduction $\mathbb{Z}_p[t][Z;\sigma,\delta] \rightarrow \mathbb{Z}_p[Z;\sigma,\delta]$ is not generally an Ore ring homomorphism [9]. Instead of performing the row operations on the Ore polynomial matrices directly, the problem was converted to one involving a system of linear equations over $\mathbb{Z}_p$. Larger striped Krylov matrices over $\mathbb{Z}_p$ was constructed and row reductions were performed on these matrices. Each Krylov matrix was constructed dynamically—rows were added depending on which row is selected as the pivot in each step. This was needed to ensure that the correct image was computed during the reduction in the presence of potential unlucky homomorphisms, even though unlucky homomorphisms occur rarely in practice. Thus, the modular algorithm was a trade-off between not exploiting polynomial arithmetic (or equivalently, the structure of the matrix) and the improved efficiency of coefficient arithmetic in simpler domains.

One obstacle in obtaining further improvement was that the row operations to reduce the Krylov matrix have to be done one step at a time, because it is not possible to construct the entire Krylov matrix *a priori* or the wrong system of solutions may have been solved. As a result, the only linear algebra subroutines in the `LinearAlgebra:-Modular` package in Maple used to accelerate the computation were operations on individual rows instead of the entire matrix. The resulting implementation has to switch back and forth between high-level Maple code and low-level compiled linear algebra subroutines that are significantly faster. In practice, the resulting modular algorithm was only faster than the corresponding fraction-free algorithm for very large inputs.

In this work, we investigate the applicability of linear algebra subroutines on blocks of matrices to speed up the computation. Assuming that the first evaluation point is "lucky," the Krylov matrices for the remaining evaluation points can be constructed and the entire matrix can be reduced with a few calls to the appropriate linear algebra subroutines. This allows more sophisticated implementations of linear algebra subroutines to speed up the reduction process (e.g. [5]).

2 Notation and Definitions

The definitions given here are similar to those in our previous works [2, 3].

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 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 $Z^{\vec{\omega}}$ the diagonal matrix having Z^{ω_i} on the diagonal.

Let k be any field and let $\sigma : k \rightarrow k$ be an injective endomorphism of k. Then, a *derivation* $\delta : k \rightarrow k$ 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. The *row degree* of an Ore polynomial matrix $\mathbf{F}(Z)$ is $\vec{\nu} = \operatorname{rdeg} \mathbf{F}(Z)$ if the i-th row has degree ν_i. Some useful properties of matrices of Ore polynomials, such as linear independence and rank, can be found in [2].

The problem of row reduction of Ore polynomial matrices 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 $\mathbf{F}(Z)$ if

$$\mathbf{P}(Z) \cdot \mathbf{F}(Z) = \mathbf{R}(Z) \cdot Z^{\vec{\omega}} \tag{1}$$

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

1. every row, $\mathbf{M}(Z)_{i,*}$, has order $\vec{\omega}$ for all $1 \le i \le m$;
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 $\deg \mathbf{L}(Z)_{k,l} \le \mu_l - 1$.

An order basis represents all row operations to eliminate a specified number of low-order coefficients. An order basis of a particular order and degree, if it exists, is unique up to a constant multiple [2, Theorem 4.4]. When $\vec{\omega} = (mN+1)\cdot\vec{e}$ and $\mathbf{R}(Z)$ is the corresponding residual, the rows in $\mathbf{M}(Z)$ corresponding to the zero rows in $\mathbf{R}(Z)$ give a basis for the left nullspace of $\mathbf{F}(Z)$. However, it is not known *a priori* the row degree $\vec{\mu}$ of the order basis. A row-reduced form and weak Popov form, together with the unimodular transformation matrix, can be extracted from $\mathbf{M}(Z)$ and $\mathbf{R}(Z)$ if $\mathbf{F}(Z)$ is a matrix of shift polynomials [2]. In the general case of matrices of Ore polynomials, the computation of the Popov form can be formulated as a left nullspace computation and can be extracted from the result of an order basis computation [3].

3 The Modular Algorithm

A modular algorithm was given in [3] to compute the order basis and the residual. The fraction-free algorithm [2] can be reduced easily from $\mathbb{Z}[t][Z]$ to $\mathbb{Z}_p[t][Z]$ using Chinese remaindering. The usual issue of normalization of the image, detecting unlucky homomorphisms, and termination can be dealt with as described in [3]. It should be noted that the

[*]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.

algorithm is output-sensitive in that the number of primes used is determined by the output size, and there is no need to verify the result (e.g. by trial division).

However, the reduction from $\mathbb{Z}_p[t][Z]$ to $\mathbb{Z}_p[Z]$ was not possible because the evaluation homomorphism $t \leftarrow \alpha$ is generally not an Ore ring homomorphism. Instead, we formulate the order basis problem as a system of linear equations over $\mathbb{Z}_p$ and perform Gaussian elimination on the coefficient matrix. The method we follow is similar to polynomial GCD computation by Gaussian elimination on the well-known Sylvester matrix [10]. It can also be considered an extension to the modular algorithm for Ore polynomial GCRD computation of Li and Nemes [9].

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

$$\begin{array}{c} \begin{array}{ccccc} & Z^0 & \cdots & Z^{\mu_k-1+\delta_{1,k}} & \end{array} \\ \left[\begin{array}{c|ccc|c} \cdots & p_k^{(0)} & \cdots & p_k^{(\mu_k-1+\delta_{1,k})} & \cdots \end{array}\right] \end{array} \cdot \begin{array}{c} \begin{array}{ccc} Z^0 & \cdots & Z^{\vec{\omega}-\vec{e}} \end{array} \\ \left[\begin{array}{c|c|c} & \vdots & \\ \hline \cdots & Z^0 \cdot F_{k,\cdot}(Z) & \cdots \\ & \vdots & \\ \cdots & Z^{\mu_k-1+\delta_{1,k}} \cdot F_{k,\cdot}(Z) & \cdots \\ \hline & \vdots & \end{array}\right] \end{array} = \mathbf{0}. \tag{2}$$

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

$$\mathbf{P}_{\vec{v}} = \left[P_{*,1}^{(0)} \quad \cdots \quad P_{*,1}^{(v_1)} | \cdots | P_{*,n}^{(0)} \quad \cdots \quad P_{*,n}^{(v_n)}\right]. \tag{3}$$

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

$$K(\vec{\mu},\vec{\omega}) = \begin{bmatrix} c_0(\mathbf{F}(Z)_{1,*}) & \cdots & c_{\sigma-1}(\mathbf{F}(Z)_{1,*}) \\ \vdots & & \vdots \\ c_0(Z^{\mu_1}\cdot \mathbf{F}(Z)_{1,*}) & \cdots & c_{\sigma-1}(Z^{\mu_1}\cdot \mathbf{F}(Z)_{1,*}) \\ \vdots & & \vdots \\ c_0(\mathbf{F}(Z)_{m,*}) & \cdots & c_{\sigma-1}(\mathbf{F}(Z)_{m,*}) \\ \vdots & & \vdots \\ c_0(Z^{\mu_m}\cdot \mathbf{F}(Z)_{m,*}) & \cdots & c_{\sigma-1}(Z^{\mu_m}\cdot \mathbf{F}(Z)_{m,*}) \end{bmatrix}. \tag{4}$$

Then the i-th row of the order basis satisfies

$$(\mathbf{M}_{i,*})_{\vec{\mu}-\vec{e}+\vec{e}_i} \cdot K(\vec{\mu}-\vec{e}+\vec{e}_i,\vec{\omega}) = \mathbf{0}. \tag{5}$$

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 $n = 1$. We also define $K^*(\vec{\mu},\vec{\omega})$ to be the matrix $K(\vec{\mu},\vec{\omega})$ with linearly dependent columns removed.

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

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

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

$$K(\vec{\mu}, \vec{\omega}) = \left[\begin{array}{cc|cc|cc} 6t^2 & 2 & 3t & -1 & 2 & 1 \\ 12t & 0 & 6t^2+3 & 2 & 3t & -1 \\ 12 & 0 & 24t & 0 & 6t^2+6 & 2 \\ \hline 3t^3 & t & t-1 & 3t & 0 & 0 \\ 9t^2 & 1 & 3t^3+1 & t+3 & t-1 & 3t \\ 18t & 0 & 18t^2 & 2 & 3t^3+2 & t+6 \end{array}\right]. \tag{7}$$

One way to obtain an order basis of degree $\vec{\mu}$ and order $\vec{\omega}$ is to perform Gaussian elimination on $K(\vec{\mu}, \vec{\omega})$ so that the first $\vec{\omega}$ columns are eliminated. The rows in the submatrix $K(\vec{\mu} - \vec{e}, \vec{\omega})$ are used for pivots in the elimination process, and the remaining rows give the residual $\mathbf{R}(Z)$. The order basis can be recovered from the transformation matrix corresponding to these rows.

Example 3.2 *Continuing from Example 3.1, we perform Gaussian elimination on $K((2,2),(3,3))$ using the first two rows of each stripe as pivots. After removing some common factors in each row to reduce the results, the resulting matrix is*

$$\left[\begin{array}{cccccc} 6t^2 & 2 & 3t & -1 & 2 & 1 \\ 0 & -4 & 6t^3-3t & 2t+2 & 3t^2-4 & -t-2 \\ 0 & 0 & 0 & 0 & -252t^5+270t^4-234t^3-22t^2-16t+16 & 882t^4-104t^2-56t-10 \\ 0 & 0 & -3t^2+2t-2 & 7t & -2t & -t \\ 0 & 0 & 0 & 21t^2-14 & 9t^4-12t^3+10t^2-8t+8 & -21t^3+11t^2-14t+2 \\ 0 & 0 & 0 & 0 & -126t^6+135t^5-180t^4-11t^3+118t^2-20t & 441t^5-52t^3-28t^2-103t \end{array}\right] \tag{8}$$

with the corresponding transformation matrix

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & t & 0 & 0 & 0 & 0 \\ -6t^2+4 & -126t^4+6t^3-4t^2+4t+14 & 21t^3-14t & -252t^2+24t+34 & 252t^3-12t^2-34t-8 & 0 \\ -t & 0 & 0 & 2 & 0 & 0 \\ -3t^2+2 & 3t^3-2t^2+2t & 0 & 12t-4 & -6t^2+4t-4 & 0 \\ -3t^3+2t & -63t^5+3t^4-2t^3+2t^2+21t & 0 & -126t^3+12t^2+59t & 126t^4-6t^3-59t^2-4t & 21t^3-14t \end{array}\right]. \tag{9}$$

The order basis $\mathbf{M}(Z)$ of degree $\vec{\mu} = (2,2)$ and order $\vec{\omega} = (3,3)$ can be easily extracted. The rows of $\mathbf{M}(Z)$ are:

$$\left[(21t^3-14t)Z^2 + (-126t^4+6t^3-4t^2+4t+14)Z - 6t^2 + 4 \quad (252t^3-12t^2-34t-8)Z - 252t^2 + 24t + 34\right]$$

and

$$\left[(-63t^4+3t^3-2t^2+2t+21)Z - 3t^2 + 2 \quad (21t^3-14t)Z^2 + (126t^3-6t^2-59t-4)Z - 126t^2 + 12t + 59\right].$$

Unfortunately, the row degree $\vec{\mu}$ of the order basis $\mathbf{M}(Z)$ of order $\vec{\omega}$ is not known *a priori*. In practice, one starts with $\vec{\mu}_0 = \vec{0}$ and performs elimination on $K(\vec{\mu}_0, \vec{\omega})$. For any $i \geq 0$, $\vec{\mu}_{i+1}$ is determined by the pivoting needed to reduce $K(\vec{\mu}_i, \vec{\omega})$ by one more column. Thus, each step in the algorithm involves performing Gaussian elimination of one column followed by adding one row to the matrix. Unlucky homomorphisms occur when the determinant of $K^*(\vec{\mu}, \vec{\omega})$ vanishes under the evaluation $t \leftarrow \alpha$. In such case, the pivoting that occurs during the elimination is different. Unlucky homomorphisms can be detected by comparing the different row degrees of the final order basis computed under each evaluation homomorphism.

The `LinearAlgebra:-Modular` package in Maple was used to perform efficient computations over $\mathbb{Z}_p$. The use of Gaussian elimination for solving the system of linear equations instead of working on the Ore polynomial matrices directly means that the modular algorithm is no longer exploiting the structure present in the Krylov matrix. On the other hand, coefficient arithmetic over $\mathbb{Z}[t]$ can be replaced by simpler coefficient arithmetic over $\mathbb{Z}_p$. For larger problems, the gain in simpler coefficient arithmetic more than offsets the loss in efficiency by not exploiting the structure. The algorithm outperforms the fraction-free algorithm [2] for very large problems even though the fraction-free algorithm exploits the structure of the Krylov matrix. However, the modular algorithm is not competitive for small input [3].

4 Improved Implementation

The implementation of the modular algorithm described in [3] has two drawbacks. First, the interleaving between matrix construction and row elimination means that routines such as Gaussian elimination (on an entire matrix) or block matrix multiplication cannot be applied to speed up the computation further. The implementation would have to switch between high-level Maple code and the faster, low-level compiled code in the `LinearAlgebra:-Modular` package. Second, the extra work and bookkeeping required for incremental matrix construction reduce the advantage of the modular algorithm. We would like to make use of low-level compiled linear algebra routines as much as possible without switching to Maple code.

In order to improve the modular algorithm, we note the unpredictability of the final row degree is mostly due to the presence of unlucky homomorphisms, but they occur rarely in practice. Therefore, the incremental elimination algorithm given previously [3] is used on one evaluation point in $\mathbb{Z}_p$. Assuming that the evaluation point (and the prime p) is not unlucky, the order basis computed has the correct degree $\vec{\mu}$. If $\vec{\mu}$ turns out to be incorrect, it will be detected when combined with the results from other primes. In that case, we perform extra computations in $\mathbb{Z}_p$ that are wasted. However, it does not occur often in practice.

When the correct degree $\vec{\mu}$ of the order basis is known (as assumed), it is relatively straightforward to compute the order basis and the residual:

1. construct $\mathbf{A} = [K(\vec{\mu}, \vec{\omega} + (N+1)\cdot\vec{e}) \mid \mathbf{I}]$;
2. perform Gaussian elimination on $\mathbf{A}$ to compute a reduced row echelon form to eliminate the first $|\vec{\omega}|$ columns, using only rows in $K(\vec{\mu}-\vec{e}, \vec{\omega})$ as pivots;
3. record the linearly dependent columns J as well as $d = \det K^*(\vec{\mu}-\vec{e}, \vec{\omega})$ which is (up to sign) the product of the pivots used;
4. construct $\mathbf{B}$ from $-\mathbf{A}_{*,J}$ after removing the pivot rows and inserting the $m \times m$ identity matrix into the columns corresponding to those rows.
5. compute $\mathbf{C} = (-1)^{\sum_{i=2}^{m}\mu_i} \cdot d \cdot \mathbf{B} \cdot \mathbf{A}$;
6. if $\mathbf{C}$ is not zero in the first $|\vec{\omega}|$ columns, then the homomorphism is unlucky. Otherwise, extract $\mathbf{R}(Z)$ from the left part and $\mathbf{M}(Z)$ from the right part of $\mathbf{C}$.

The Gaussian elimination in Step 2 can be performed, for example, by calling the `RowReduce` routine in the `LinearAlgebra:-Modular` package of Maple on the entire matrix $\mathbf{A}$. The matrix multiplication in Step 5 can be performed by the `Multiply` routine. As a result, the new implementation can fully take advantage of good low-level implementation of block Gaussian elimination and multiplication (e.g. [5]). Since there is no need to perform incremental matrix construction, both memory management and bookkeeping are reduced. In addition, the control of the program can stay inside the low-level `LinearAlgebra:-Modular` subroutines instead of switching back and forth between them and Maple code.

Example 4.1 *We apply this method to Example 3.1. We perform our calculations in* $\mathbb{Z}_{31}$ *and perform the evaluation* $t \leftarrow 7$*. To conserve space, we only show* $\mathbf{A}' = [K(\vec{\mu}, \vec{\omega}) \mid \mathbf{I}]$ *and compute only* $\mathbf{M}(Z)$*. Initially,*

$$\mathbf{A}' = \begin{bmatrix} 15 & 2 & 21 & 30 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 22 & 0 & 18 & 2 & 21 & 30 & 0 & 1 & 0 & 0 & 0 & 0 \\ 12 & 0 & 13 & 0 & 21 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 6 & 7 & 6 & 21 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 7 & 1 & 7 & 10 & 6 & 21 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 14 & 2 & 8 & 13 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

Performing Gaussian elimination on rows 1, 2, 4, and 5, we obtain:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 14 & 14 & 29 & 5 & 0 & 27 & 1 & 0 \\ 0 & 1 & 0 & 0 & 25 & 9 & 11 & 1 & 0 & 24 & 28 & 0 \\ 12 & 0 & 13 & 0 & 21 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 14 & 25 & 0 & 23 & 0 & 6 & 20 & 0 \\ 0 & 0 & 0 & 1 & 25 & 8 & 22 & 2 & 0 & 21 & 26 & 0 \\ 2 & 0 & 14 & 2 & 8 & 13 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

Here, $J = \{1, 2, 3, 4\}$ *and* $d = 26$*. Thus,*

$$\begin{aligned} \mathbf{C} &= 26 \cdot \begin{bmatrix} -12 & 0 & \mathbf{1} & -13 & 0 & \mathbf{0} \\ -2 & 0 & \mathbf{0} & -14 & -2 & \mathbf{1} \end{bmatrix} \cdot \mathbf{A}' \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 6 & 4 & 28 & 26 & 26 & 27 & 0 \\ 0 & 0 & 0 & 0 & 28 & 14 & 14 & 6 & 0 & 1 & 27 & 26 \end{bmatrix}, \end{aligned}$$

where the highlighted entries are the identity matrix inserted to form $\mathbf{B}$*. Therefore, the image of the order basis computed is*

$$\mathbf{M}(Z) = 5 \cdot \begin{bmatrix} 6Z^2 + 16Z + 20 & 11Z + 6 \\ 30Z + 8 & 6Z^2 + 11Z + 5 \end{bmatrix}. \tag{12}$$

One may easily verify that this is a scalar multiple of the image of the order basis computed in Example 3.2 under the evaluation $t \leftarrow 7$ *in* $\mathbb{Z}_{31}$*.*

The introduction of the scalar multiple is due to the removal of content in Example 3.2. The implementation given here in fact computes exactly the same result (including the scalar multiple) as the previous fraction-free and modular implementations for the order basis problem [2, 3].

5 Experimental Results

Experiments were performed on Ore polynomial matrices in differential case. The results of these experiments are shown in Tables 1 and 2. The application of block linear algebra routines reduces the running time of the modular algorithm in all cases. The improvement is more significant for smaller problems, where the original modular algorithm is not competitive against the fraction-free problems.

Table 1: Comparison of fraction-free, modular, and the new modular algorithm on random $m \times n$ matrices with $\deg_t = 1$ and integer coefficients having magnitude ≤ 5.

m, n	N	FFreduce (s)	Modular (s)	New Modular (s)	Improvement
2	1	0.023	0.115	0.069	40%
2	2	0.107	0.242	0.192	21%
2	4	1.689	2.984	2.301	23%
2	8	15.047	28.499	23.232	18%
2	16	278.883	279.041	232.877	17%
2	32	5447.542	4669.801	3992.689	15%
3	1	0.472	1.060	0.723	32%
3	2	3.808	6.268	5.416	20%
3	4	41.549	51.498	44.253	14%
3	8	667.682	599.466	521.571	13%
4	2	41.348	47.841	41.765	13%
4	4	663.707	554.060	487.000	12%
4	6	3850.143	2561.281	2303.989	10%
5	2	293.122	260.021	227.314	13%
5	4	6179.169	3845.945	3362.376	13%
8	1	1660.258	1088.609	998.659	8%
10	1	16179.879	8019.137	7524.240	6%

Table 2: Comparison of fraction-free, modular, and the new modular algorithm on random $m \times n$ matrices with $\deg_t = 2$ and integer coefficients having magnitude ≤ 5.

m, n	N	FFreduce (s)	Modular (s)	New Modular (s)	Improvement
2	2	0.470	1.647	1.378	16%
2	4	5.920	11.611	10.004	14%
2	8	86.214	128.528	109.911	14%
2	16	1237.410	1437.492	1300.804	10%
3	2	14.718	25.101	21.954	13%
3	4	216.214	267.295	238.497	11%
3	6	1157.705	1220.524	1114.106	9%
3	8	3933.234	3994.955	3735.837	6%
4	2	170.561	193.981	174.399	10%
4	4	2397.460	2270.272	2096.580	8%

For the larger problems, however, the improvement is less significant. For larger problems, the size of the coefficients in the output becomes larger as well. More time is spent on the other parts of the algorithm such as reconstruction by Chinese remaindering and memory management, and the amount of time spent on actual elimination is proportionally smaller. Since the new implementation improves mainly the elimination process, the improvement is less significant for larger problems. On the other hand, we see that the improved implementation given in this paper increases the advantage of the modular algorithm over the fraction-free algorithm, and allows the modular algorithm to be used beneficially for smaller problems.

References and Notes

[1] 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.

[2] 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.

[3] H. Cheng and G. Labahn. Modular computation for matrices of Ore polynomials. In *Computer Algebra 2006: Latest Advances in Symbolic Algorithms*, pages 43–66, 2007.

[4] P. Davies, H. Cheng, and G. Labahn. Computing Popov form of general Ore polynomial matrices. In *Milestones in Computer Algebra (MICA) 2008*, pages 149–156, 2008.

[5] J.-G. Dumas, P. Giorgi, and C. Pernet. FFPACK: finite field linear algebra package. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, pages 119–126. ACM, 2004.

[6] M. Halas, U. Kotta, Z. Li, H. Wang, and C. Yuan. Submersive rational difference systems and formal accessibility. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, pages 175–182. ACM, 2009.

[7] 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.

[8] 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.

[9] 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.

[10] R. Loos. Generalized polynomial remainder sequences. In *Computer Algebra: Symbolic and Algebraic Computation*, pages 115–137. Springer-Verlag, 1982.