Output-sensitive Modular Algorithms for Polynomial Matrix Normal Forms

Howard Cheng [a], George Labahn [b]

[a] *Department of Mathematics and Computer Science*
University of Lethbridge
Lethbridge, Canada

[b] *Symbolic Computation Group*
School of Computer Science
University of Waterloo
Waterloo, Canada

Abstract

We give modular algorithms to compute row-reduced forms, weak Popov forms, and Popov forms of polynomial matrices, as well as the corresponding unimodular transformation matrices. Our algorithms improve on existing fraction-free algorithms. In each case we define lucky homomorphisms, determine the appropriate normalization, as well as bound the number of homomorphic images required. The algorithms have the advantage that they are output-sensitive, that is, 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. Our algorithms can be used to compute normalized one-sided greatest common divisors and least common multiples of polynomial matrices along with irreducible matrix-fraction descriptions of matrix rational functions. When our algorithm is used to compute polynomial greatest common divisors, we obtain a new output-sensitive modular algorithm.

Key words: Polynomial matrices, row-reduced form, weak Popov form, Popov form, modular algorithm

Email addresses: `cheng@cs.uleth.ca` (Howard Cheng), `glabahn@scg.math.uwaterloo.ca` (George Labahn).

1 Introduction

Row-reduced forms, weak (or quasi) Popov forms, and Popov forms [Kailath, 1980, Mulders and Storjohann, 2003] of polynomial matrices, as well as the associated unimodular transformation matrices, appear in many application areas in computer science and engineering (for example in control theory). Informally, a polynomial matrix is in row-reduced form if its leading row coefficient matrix has maximal rank, and it is in weak Popov form if its leading row coefficient is in upper echelon form (up to row permutation). A polynomial matrix is in Popov form if it is in weak Popov form and the leading column coefficient matrix is diagonal. Normal forms for polynomial matrices are used in such computations as one-sided greatest common divisor or least common multiple of two polynomial matrices [Kailath, 1980].

These normal forms are traditionally computed by performing elementary row operations to eliminate unwanted coefficients [Beckermann and Labahn, 1997, Kailath, 1980, Mulders and Storjohann, 2003]. By reversing the coefficients of a polynomial matrix, the problem of row-reduction is equivalent to one of elimination of trailing coefficients. The latter can be solved by finding an *order basis* of a polynomial matrix [Beckermann and Labahn, 1997]. In the common cases of integer or polynomial coefficients, such row operations can introduce significant intermediate expression swell which must be carefully controlled. The Fast Fraction-free Gaussian (FFFG) elimination algorithm of Beckermann and Labahn [2000b] can be used to compute one-sided GCDs for polynomial matrices while controlling coefficient growth. The procedure was later extended to compute row-reduced forms [Beckermann and Labahn, 2001] and weak Popov forms [Cheng, 2003, Sec. 4.4] of polynomial matrices. The FFFG elimination algorithm has also been used for computing Popov forms [Beckermann et al., 1999, **?**] by reducing the problem to one of computing minimal bases. The FFFG elimination algorithm, which includes the subresultant algorithm as a special case [Beckermann and Labahn, 2000a], predicts common factors by examining the linear systems of equations associated with the order basis problem. For an input $m \times n$ polynomial matrix of degree N, the FFFG algorithm has a worst case bit complexity of $O((m+n)\alpha^3 \mathsf{M}(\alpha(\log \alpha + K)))$ when the coefficients are integers of size (number of bits) bounded by K, $\alpha = m \min(m,n)N$, and $\mathsf{M}(K)$ is the complexity of multiplying two integers of length K. The worst case bit complexity is $O((m+n)\alpha^4 K^2)$ when the coefficients are polynomials in $\mathbb{Z}[x]$ with degrees bounded by K and quadratic multiplication is assumed [Cheng, 2003, Sec. 3.6].

In this paper we are interested in the modular computation of row-reduced, weak Popov, and Popov forms for polynomial matrices, as well as 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 ensure a unique answer. Although these issues are well understood in the case of polynomial GCD computations, they are nontrivial in our case. The problem of computing these normal forms and their transformation matrices can be viewed as one of determining a basis for a particular module. The difficulties in devising a modular algorithm to compute a basis for a module or a vector space are well known. The lack of uniqueness, definitions of lucky homomorphisms, and appropriate normalizations are all far from obvious in any basis computation. We overcome these difficulties by reducing the problem to a linear algebra problem.

While modular algorithms for systems of linear equations are well known [Geddes et al., 1992], the design of our modular algorithms is complicated by the fact that the corresponding fraction-free algorithm computes solutions to a sequence of systems of linear equations. The sequence computed under each homomorphism may be different, and it is not clear a priori which linear systems are the desired ones. Additional insights are required to detect unlucky homomorphisms. In particular, we must ensure that the linear systems solved under different homomorphisms are the same. Our approach is to study the linear systems examined by the FFFG elimination algorithm and compare them against those examined by the modular algorithm. The key to our approach lies in examining the pivot choices (or a "computational path") during the execution of the algorithm.

Our algorithms can be used to solve the extended one-sided GCD problem for polynomial matrices. This allows us to compute one-sided least common multiples of polynomial matrices, as well as irreducible matrix-fraction descriptions of matrix rational functions. We also obtain an efficient test for coprimeness of two row-reduced polynomial matrices. In the special case of polynomial GCD, we obtain a new modular algorithm for the extended GCD problem similar to that of Brown [1971] except that no trial division or multiplication is required to verify the result after early termination.

By following an approach of Cabay [1971] we obtain modular algorithms with early termination, leading to output-sensitive algorithms. Unlike the polynomial GCD case, it is not easy to verify the result by "trial division" of the input by the candidate normal form. The elimination of the verification step is important. While Kaltofen and Monagan [1999] also suggested lifting the "cofactors" and checking that the sum of the degrees is correct for the polynomial GCD problem, the degree check is insufficient for the matrix case and polynomial matrix multiplication is needed to verify the result. Our approach is different in that when the results have not changed after a few additional homomorphic images, the results are proven to be correct without additional verification. The worst case complexity of the new modular algorithm is $O((m+n)\alpha\mathsf{M}(\alpha(\log\alpha + K))(\alpha+\log(\alpha(\log\alpha+K))))$ when the coefficients are in $\mathbb{Z}$, and $O((m+n)\alpha^3 K^2)$ when the coefficients are in $\mathbb{Z}[x]$. This is a significant improvement over the fraction-free algorithm. Furthermore, the algorithm can be significantly faster when the size of output is small. While our analyses and proofs assume that the polynomial matrices have integer or univariate polynomial (over a field) coefficients, they can be adapted to other Euclidean domains such as the Gaussian integers.

The paper is organized as follows. In Section 2, we review the relevant definitions of polynomial matrices. In Section 3, we review the FFFG elimination algorithm and give examples illustrating the difficulties in obtaining a modular algorithm. In the next two sections we develop the ingredients of the modular algorithms: computation of the homomorphic images of the normal form, normalization, and definition of lucky homomorphisms. In Section 6, we study early termination for our algorithms, and in Section 7 we examine a termination strategy that is sensitive to both input and output. In Section 8 we study the complexity of the algorithms presented. Section 9 explains how the algorithms can be used to compute a one-sided GCD of polynomial matrices, and provides a fast test for coprimeness of two polynomial matrices. Section 10 gives some experimental results. Concluding remarks and future research directions are discussed in the closing section.

2 Polynomial Matrices: Definitions

Let $\mathbb{D}$ be an integral domain with quotient field $\mathbb{Q}_{\mathbb{D}}$. We shall denote the ring of integers $\mathbb{Z}$, the field of rational numbers $\mathbb{Q}$, and the finite field of p elements $\mathbb{Z}_p$ for a prime p. In this paper, we will only examine the cases where $\mathbb{D} = \mathbb{Z}$ and $\mathbb{D} = \mathbb{Z}[x]$. Other choices of $\mathbb{D}$ (such as $\mathbb{D} = \mathbb{K}[x]$ for any field $\mathbb{K}$) can also be used.

Let $\mathbb{D}[z]^{m\times n}$ and $\mathbb{Q}_{\mathbb{D}}[z]^{m\times n}$ be the rings of $m \times n$ polynomial matrices over $\mathbb{D}$ and $\mathbb{Q}_{\mathbb{D}}$, respectively. Let $\mathbf{F}(z) \in \mathbb{Q}_{\mathbb{D}}[z]^{m\times n}$ and $N = \deg \mathbf{F}(z)$, and write

$$\mathbf{F}(z) = \sum_{j=0}^{N} F^{(j)} z^j, \text{ with } F^{(j)} \in \mathbb{Q}_{\mathbb{D}}^{m\times n}.$$

We also write $c_j(\mathbf{F}(z)) = F^{(j)}$ to denote the j-th coefficient matrix. We denote the elements of $\mathbf{F}(z)$ by $\mathbf{F}(z)_{k,\ell}$, the elements of $F^{(j)}$ by $F^{(j)}_{k,\ell}$, the i-th row of $\mathbf{F}(z)$ by $\mathbf{F}(z)_{i,*}$, and the j-th column by $\mathbf{F}(z)_{*,j}$. For any sets of row and column indices I and J, $\mathbf{F}(z)_{I,*}$ is the submatrix of $\mathbf{F}(z)$ consisting of the rows indexed by I, $\mathbf{F}(z)_{*,J}$ is the submatrix of $\mathbf{F}(z)$ consisting of the columns indexed by J, and $\mathbf{F}(z)_{I,J}$ is the submatrix of $\mathbf{F}(z)$ consisting of the rows and columns indexed by I and J.

For any vector of non-negative 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. Additionally, two vectors can be compared in lexicographical order. We say that $\vec{v} \leq_{lex} \vec{w}$ 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.

A 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 $\text{LC}_{row}(\mathbf{F}(z))$, is the matrix whose i-th row are the coefficients of z^{ν_i} of the corresponding elements of $\mathbf{F}(z)$. A polynomial matrix $\mathbf{F}(z)$ is *row-reduced* if $\text{LC}_{row}(\mathbf{F}(z))$ has maximal row rank. $\mathbf{F}(z)$ is in *weak Popov form* if $\text{LC}_{row}(\mathbf{F}(z))$ is in upper echelon form (up to row permutation). In other words, if we define the *pivot index* of row i, denoted Π_i to be

$$\Pi_i = \begin{cases} \min_{1 \leq j \leq n} \{ j : \deg \mathbf{F}(z)_{i,j} = \deg \mathbf{F}(z)_{i,*} \} & \text{if } \mathbf{F}(z)_{i,*} \neq \mathbf{0} \\ 0 & \text{if } \mathbf{F}(z)_{i,*} = \mathbf{0} \end{cases},$$

then $\Pi_i \neq \Pi_j$ whenever $i \neq j$, and $\mathbf{F}(z)_{i,*}$ and $\mathbf{F}(z)_{j,*}$ are both nonzero. Finally, a polynomial matrix $\mathbf{F}(z)$ is in *Popov form* if it is in weak Popov form, and if i is such that $\mathbf{F}(z)_{i,*}$ is nonzero, then

(1) $\mathbf{F}(z)_{i,\Pi_i}$ is monic;
(2) $\deg \mathbf{F}(z)_{j,\Pi_i} < \deg \mathbf{F}(z)_{i,\Pi_i}$ for all $j \neq i$.

3 The FFFG Elimination Algorithm

In this section we give a brief description of the FFFG elimination algorithm [Beckermann and Labahn, 2000b] which forms much of the basis of our work. Instead of eliminating unwanted high-order coefficients in the input polynomial matrix $\mathbf{A}(z) \in \mathbb{D}[z]^{m \times n}$ of degree N, we reverse the coefficients to get its reciprocal $\mathbf{F}(z) = \mathbf{A}(1/z) \cdot z^N$ and eliminate the low-order coefficients instead.

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

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

for some *residual* $\mathbf{R}(z) \in \mathbb{D}[z]^{m\times n}$. It is easy to see that the set of all polynomial vectors of a particular order $\vec{\omega}$ forms a $\mathbb{Q}_{\mathbb{D}}[z]$-module. The FFFG algorithm computes a basis $\mathbf{M}(z) \in \mathbb{D}[z]^{m\times m}$ 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 $1 \le i \le m$;
(2) the rows of $\mathbf{M}(z)$ form a basis of the module of all vectors of order $\vec{\omega}$. That is, every $\mathbf{P}(z) \in \mathbb{Q}_{\mathbb{D}}[z]^{1\times m}$ of order $\vec{\omega}$ can be written as $\mathbf{P}(z) = \mathbf{Q}(z)\cdot\mathbf{M}(z)$ for some $\mathbf{Q}(z) \in \mathbb{Q}_{\mathbb{D}}[z]^{1\times m}$;
(3) the leading column coefficient is normalized. That is, there exists a nonzero $d \in \mathbb{Q}_{\mathbb{D}}$ such that
$$\mathbf{M}(z) = d \cdot z^{\vec{\mu}} + \mathbf{L}(z)$$
where $\deg \mathbf{L}(z)_{k,l} \le \mu_l - 1$.

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 3.4). An order basis of a particular order and degree, if it exists, is unique up to a scalar constant [Beckermann and Labahn, 2000b]. In the FFFG 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 (or computational path) to be described later.

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

$$\mathbf{M}(z) \cdot \mathbf{F}(z) = \mathbf{R}(z) \cdot z^{\vec{\omega}} \tag{2}$$

with $\vec{\omega} = \sigma \cdot \vec{e}$. It has been shown [**?**, Section 9] that if $\sigma = mN + 1$, then $|\vec{\mu}| \le \sigma \min(m, n)$ and the trailing coefficient of $\mathbf{R}(z)$ has rank $\mathbf{F}(z)$ nonzero rows. Furthermore, one can choose the pivot rows used in the last n steps of the algorithm to construct $\hat{\mathbf{M}}(z)$ and $\hat{\mathbf{R}}(z)$ satisfying (2) with the trailing coefficient of $\hat{\mathbf{R}}(z)$ in upper echelon form [Cheng, 2003, Sec. 4.4]. Reversing the coefficients of $\mathbf{M}(z)$ and $\mathbf{R}(z)$ (or $\hat{\mathbf{M}}(z)$ and $\hat{\mathbf{R}}(z)$) gives $\mathbf{U}(z) = z^{\vec{\mu}} \cdot \mathbf{M}(1/z)$ and $\mathbf{T}(z) = z^{\vec{\mu}} \cdot \mathbf{R}(1/z) \cdot z^{N\cdot\vec{e}-\vec{\omega}}$, such that

$$\mathbf{U}(z) \cdot \mathbf{A}(z) = \mathbf{T}(z) \tag{3}$$

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

with $\mathbf{U}(z)$ unimodular and $\mathbf{T}(z)$ in row-reduced form (weak Popov form).

The FFFG algorithm can also be used to compute the Popov form of a polynomial matrix [Beckermann et al., 1999, **?**]. In particular, we compute an order basis $\mathbf{M}(z)$ of the input matrix $\begin{bmatrix} \mathbf{A}(z) \\ -\mathbf{I}_n \end{bmatrix}$ of a sufficiently large order. Then, a basis $\mathbf{M}'(z)$ of the nullspace of the input matrix can be obtained by choosing the rows of $\mathbf{M}(z)$ corresponding to the zero rows of the residual. If we use a particular pivoting scheme then

$$\mathbf{M}'(z) = \begin{bmatrix} \mathbf{U}(z) & \mathbf{T}(z) \end{bmatrix}$$

where $\mathbf{U}(z)$ is unimodular and $\mathbf{T}(z)$ is a constant multiple of a matrix in Popov form.

Since we are interested in matrix normal forms, we will only consider the order basis problem of (2) with $\vec{\omega} = \sigma \cdot \vec{e}$ for the remainder of this paper.

3.2 Linear Algebra Formulation

In (2), the coefficients of $\mathbf{M}(z)_{i,*}$ can be viewed as a solution to the linear system of equations:

$$(\mathbf{M}_{i,*})_{\vec{\mu}-\vec{e}+\vec{e}_i} \cdot \mathbf{K}(\vec{\mu} - \vec{e} + \vec{e}_i, \vec{\omega}) = \mathbf{0}, \tag{4}$$

where for any $\mathbf{P}(z) \in \mathbb{Q}_{\mathbb{D}}[z]^{m\times n}$ (recall that $\vec{\omega} = \sigma \cdot \vec{e}$)

$$\mathbf{P}_{\vec{v}} = \begin{bmatrix} P^{(0)}_{*,1} & \cdots & P^{(v_1)}_{*,1} | & \cdots & | \, P^{(0)}_{*,n} & \cdots & P^{(v_n)}_{*,n} \end{bmatrix} \tag{5}$$

$$\mathbf{K}(\vec{\mu}, \vec{\omega}) = \begin{bmatrix} c_0(\quad \mathbf{F}(z)_{1,*}) & \cdots & c_{\sigma-1}(\quad \mathbf{F}(z)_{1,*}) \\ \vdots & & \vdots \\ c_0(z^{\mu_1} \cdot \mathbf{F}(z)_{1,*}) & \cdots & c_{\sigma-1}(z^{\mu_1} \cdot \mathbf{F}(z)_{1,*}) \\ \vdots & & \vdots \\ c_0(\quad \mathbf{F}(z)_{m,*}) & \cdots & c_{\sigma-1}(\quad \mathbf{F}(z)_{m,*}) \\ \vdots & & \vdots \\ c_0(z^{\mu_m} \cdot \mathbf{F}(z)_{m,*}) & \cdots & c_{\sigma-1}(z^{\mu_m} \cdot \mathbf{F}(z)_{m,*}) \end{bmatrix}. \tag{6}$$

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

Example 3.1 *Let* $\mathbf{A}(z) = \left[a_2 z^2 + a_1 z + a_0 \ , \ b_2 z^2 + b_1 + b_0\right]^T$*, and* $\mathbf{F}(z) = \mathbf{A}(1/z) \cdot z^2$*. Then*

$$\mathbf{K}((1,1),(4)) = \begin{bmatrix} a_2 & a_1 & a_0 & 0 \\ 0 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 \\ 0 & b_2 & b_1 & b_0 \end{bmatrix}.$$

□

Example 3.2 *Let* $\mathbf{A}(z)$ *be the* 4×2 *polynomial matrix*

$$\begin{bmatrix} 3\,z^4 + 3\,z^3 + 4\,z^2 - 2\,z - 4 & 3\,z^4 + 3\,z^2 + 14\,z + 8 \\ z^4 + 5\,z^3 + 3\,z^2 + 3\,z + 1 & z^4 + 7\,z^3 + 6\,z^2 + z + 1 \\ z^3 + 9\,z^2 + 5\,z + 1 & z^3 + 15\,z^2 + 19\,z + 5 \\ z^5 + z^4 + 2\,z^3 + 3\,z^2 + 2\,z + 1 & z^5 + z^3 + 7\,z^2 + 6\,z + 1 \end{bmatrix}.$$

If we reverse the coefficients and let $\mathbf{F}(z) = \mathbf{A}(1/z) \cdot z^5$*, we get, for example,*

$$\mathbf{K}((1,1,1,1),(6,6)) = \left[\begin{array}{cc|cc|cc|cc|cc|cc} 0 & 0 & 3 & 3 & 3 & 0 & 4 & 3 & -2 & 14 & -4 & 8 \\ 0 & 0 & 0 & 0 & 3 & 3 & 3 & 0 & 4 & 3 & -2 & 14 \\ \hline 0 & 0 & 1 & 1 & 5 & 7 & 3 & 6 & 3 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 5 & 7 & 3 & 6 & 3 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 9 & 15 & 5 & 19 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 9 & 15 & 5 & 19 \\ \hline 1 & 1 & 1 & 0 & 2 & 1 & 3 & 7 & 2 & 6 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 2 & 1 & 3 & 7 & 2 & 6 \end{array}\right].$$

□

We also define the matrix $\mathbf{K}^*(\vec{\mu},\vec{\omega}) = \mathbf{K}(\vec{\mu},\vec{\omega})_{*,J}$ where J is the lexicographically smallest set of column indices such that $\mathbf{K}^*(\sigma \cdot \vec{e},\vec{\omega})_{*,J}$ has full column rank (this is called the rank profile by Storjohann [2000]). An order basis of degree $\vec{\mu}$ and order $\vec{\omega}$ exists if $\mathbf{K}^*(\vec{\mu} - \vec{e},\vec{\omega})$ is nonsingular [Beckermann and Labahn, 2000b], and in that case, (4) has a solution space of dimension one. The system can be transformed into the following system with a unique solution:

$$(\mathbf{M}_{i,*})_{\vec{\mu}-\vec{e}} \cdot \mathbf{K}^*(\vec{\mu} - \vec{e},\vec{\omega}) = d \cdot \left[c_0\left(z^{\mu_i} \cdot \mathbf{F}(z)_{i,*}\right) \ \ \cdots \ \ c_{\sigma-1}\left(z^{\mu_i} \cdot \mathbf{F}(z)_{i,*}\right)\right] \tag{7}$$

where $d = \pm \det \mathbf{K}^*(\vec{\mu} - \vec{e}, \vec{\omega})$. In other words, we are interested in the Cramer solution of (7).

3.3 Fraction-free Recursion

Starting from $\mathbf{M}(z) = \mathbf{I}_m$ and $\vec{\omega} = \vec{0}$, the FFFG algorithm computes order bases for increasing $\vec{\omega}$ until the desired order is reached. It performs a fraction-free elimination of the matrix $\mathbf{K}(\vec{\mu}, \vec{\omega})$ efficiently by taking advantage of the inherent structure in the matrix $\mathbf{K}(\vec{\mu}, \vec{\omega})$. Indeed the algorithm has the effect of maintaining only one row in each of the m stripes. 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 [Beckermann and Labahn, 2000b, Theorem 6.1]:

Theorem 3.3 *Let $\mathbf{M}(z)$ be an order basis of order $\vec{\omega}$ and degree $\vec{\mu}$, and $r_\ell = c_{\omega_j}\left((\mathbf{M}(z) \cdot \mathbf{F}(z))_{\ell,j}\right)$. 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 $r_\pi \neq 0$, and let $p_\ell = c_{\mu_\ell - 1 + \delta_{\pi,\ell}}(\mathbf{M}(z)_{\pi,\ell})$. 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*

$$p_\pi \cdot \widetilde{\mathbf{M}}(z)_{\ell,*} = r_\pi \cdot \mathbf{M}(z)_{\ell,*} - r_\ell \cdot \mathbf{M}(z)_{\pi,*} \quad \textit{for } \ell \neq \pi; \tag{8}$$

$$p_\pi \cdot \widetilde{\mathbf{M}}(z)_{\pi,*} = z \cdot r_\pi \cdot \mathbf{M}(z)_{\pi,*} - \sum_{\ell \neq \pi} p_\ell \cdot \mathbf{M}(z)_{\ell,*}. \tag{9}$$

□

In the case of row-reduced and weak Popov form, the pivot π is chosen by the formula

$$\pi = \min_{1 \le \ell \le m} \left\{ \ell : r_\ell \neq 0, \mu_\ell = \min_{1 \le j \le m} \{\mu_j : r_j \neq 0\} \right\}. \tag{10}$$

The formula for Popov form is induced by an "offdiagonal path." We refer the reader to the references for more details [**?**].

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 [1968]. The elements of $\mathbf{M}(z)$ are Cramer solutions to (7). The order basis $\mathbf{M}(z)$ can be viewed as the transformation matrix representing the row operations performed during Gaussian elimination. We also note that the degree of the order basis (and hence the corresponding linear systems of equations) depends on the pivot choices in (10) and cannot be predicted in advance.

Example 3.4 *Let $\mathbf{A}(z)$ be defined as in Example 3.2, so that we are performing Gaussian elimination on $\mathbf{K}(\cdot, \cdot)$. In the first step of FFFG, we choose $\pi = 4$ to eliminate the first column so that the row considered in the fourth stripe is advanced to the second row of the stripe. In the second step, the entries in the second column are zero (except those rows that have already been* *used) and so no elimination is performed. In the third step, we choose $\pi = 1$ to eliminate the third column, but in the fourth step we see that all entries in the fourth column are zero and so no elimination is performed. This gives us an order basis of order $\vec{\omega} = (2, 2)$. The row degrees of the order bases computed are $(0, 0, 0, 1)$, $(0, 0, 0, 1)$, $(1, 0, 0, 1)$, and $(1, 0, 0, 1)$. The first two elements of J are 1 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.

If we perform the same computations in $\mathbb{Z}_3$, we see that the pivot chosen in the third step will be different and we have an order basis of order $(0, 1, 0, 1)$ instead. It is also possible that $r_\ell = 0$ for all ℓ at a particular step when the computation is done in $\mathbb{Z}_p$, but some r_ℓ is nonzero at the same step when the computation is done in $\mathbb{Z}$. This means that the set of column indices J is different in $\mathbb{Z}_p$. Note that the latter cannot occur in the computation of polynomial GCD (with $n = 1$) provided that the leading coefficients of the input polynomials do not vanish in $\mathbb{Z}_p$. $\square$

4 Computing Homomorphic Images and Normalization

Let $I \subseteq \mathbb{D}$ be an ideal, and $\phi : \mathbb{D} \to \mathbb{D}/I$ be the natural homomorphism defined by $\phi(a) = a + I$ for all $a \in \mathbb{D}$. If $\mathbb{D} = \mathbb{Z}$ we choose $I = p\mathbb{Z}$ for some prime $p \in \mathbb{Z}$, while if $\mathbb{D} = \mathbb{Z}[x]$ we choose $I = (x - \alpha)\mathbb{Z}[x]$ for some $\alpha \in \mathbb{Z}$. Thus, ϕ corresponds to the "modulo p" or "evaluation at α" operation. In a modular algorithm, a number of distinct primes p_i or evaluation points α_i is chosen and the computation is performed in the domain $(\mathbb{D}/I_i)[z]$ where I_i is the ideal defined by the i-th prime or evaluation point. The images of the results are combined via Chinese remaindering to obtain the desired result in $\mathbb{D}[z]$.

Let ϕ_i be the natural homomorphism defined by I_i. We will compute an order basis $\mathbf{M}_i(z)$ of degree $\vec{\mu}_i$ and order $\vec{\omega} = \sigma \cdot \vec{e}$ of $\phi_i(\mathbf{F}(z))$ with residual $\mathbf{R}_i(z)$. The desired results $\mathbf{M}(z)$ and $\mathbf{R}(z)$ are reconstructed from $\mathbf{M}_i(z)$ and $\mathbf{R}_i(z)$ via Chinese remaindering.

Since order bases are unique up to multiplication of a constant, we need to ensure that the images computed under each homomorphism correspond to the same result in $\mathbb{D}[z]$. This is ensured by normalizing the order basis such that

$\mathrm{LC}_{row}(\mathbf{M}(z))$ has $d = \det \mathbf{K}^*(\vec{\mu} - \vec{e}, \vec{\omega})$ on the diagonal (recall that $\mathbf{M}(z)$ is square) as in [Beckermann and Labahn, 2000b]. The normalization is done by applying the same operations specified in the formulas (8) and (9) in $(\mathbb{D}/I_i)[z]$. However, the sequence of pivots chosen may be different, so the result is correct up to sign (e.g. consider Example 3.2 in $\mathbb{Z}_3[z]$) which can be computed by the following formula:

$$\epsilon_0 = 1 \tag{11}$$

$$\epsilon_{k+1} = \begin{cases} \epsilon_k & \text{if } r_\ell = 0 \text{ for all } \ell, \\ \epsilon_k \cdot (-1)^{\sum_{i=\pi_k+1}^{m} (\nu_k)_i} & \text{otherwise,} \end{cases} \tag{12}$$

where π_k and $\vec{\nu}_k$ are the values of π and $\vec{\mu}$ at the k-th step of the algorithm performed in $(\mathbb{D}/I_i)[z]$ [Beckermann and Labahn, 2000b, Lemma 5.1(c)]. As long as the final degrees of the order bases $\vec{\mu}_i$ are identical, the order bases $\mathbf{M}_i(z)$ and the residuals $\mathbf{R}_i(z)$ correspond to the same result in $\mathbb{D}[z]$ with the normalization above. We also compute the vector $\vec{J}_i$ recording the steps in which formulas (8) and (9) are applied. Then $\vec{J}_i$ has $|\vec{\mu}_i|$ components and it gives the set of column indices J in the definition of $\mathbf{K}^*(\vec{\mu}_i - \vec{e}, \vec{\omega})$ in $\mathbb{D}/I_i$.

5 Lucky Homomorphisms

Although order bases are unique (up to a constant) for a given degree and order, FFFG may not always arrive at the same final degree for the order basis even when the order is the same. In terms of the systems of linear equations (7), this implies that the systems solved under distinct homomorphisms may be different. In this section, we discuss the definition of lucky homomorphisms and how they are detected.

As we have seen in Example 3.4, the sequence (or path) of row degrees of the order bases constructed during the execution of FFFG represents the choice of pivots π_k. We define the path $w = \{\vec{w}_k\}_{k=0,1,2,\ldots}$ by $\vec{w}_0 = \vec{0}$ and $\vec{w}_{k+1} = \vec{w}_k + \vec{e}_{k \bmod m+1}$. The path w is the sequence of degrees followed by FFFG for row-reduced and weak Popov forms, if $r_\ell \neq 0$ for all ℓ at every step. The path w for computing Popov forms can similarly be defined [Beckermann et al., 1999, **?**].

In either case, it was shown [Beckermann and Labahn, 2000b, Theorem 7.3] that the final degree $\vec{\mu}_i$ is the unique closest normal point[2] to w. That is, if $\mathbf{K}^*(\vec{v} - \vec{e}, \vec{\omega})$ is nonsingular for some $\vec{v}$ such that $|\vec{v}| = |\vec{\mu}_i|$, then

$$|\max(\vec{0}, \vec{w}_k - \vec{\mu}_i)| \leq |\max(\vec{0}, \vec{w}_k - \vec{v})| \text{ for all } k \geq 0. \tag{13}$$

[2] $\vec{\mu}$ is a normal point if $\mathbf{K}^*(\vec{\mu} - \vec{e}, \vec{\omega})$ is nonsingular.

To facilitate the presentation, we let $\mathbf{K}_i^*(\vec{\mu}-\vec{e},\vec{\omega})$ be the submatrix of the corresponding striped Krylov matrix $\mathbf{K}_i(\vec{\mu}-\vec{e},\vec{\omega})$ over $\mathbb{D}/I_i$, with the column indices J given by $\vec{J}_i$. We will assume that $\mathbf{M}(z)$, $\mathbf{R}(z)$, $\vec{\mu}$, and $\vec{J}$ are the output of FFFG over $\mathbb{D}$, and d is the diagonal element of $\mathrm{LC}_{row}(\mathbf{M}(z))$. The subscript i is used to indicate the corresponding quantities computed over $\mathbb{D}/I_i$.

We define lucky homomorphisms for row-reduced form computation as follows.

Definition 5.1 *The homomorphism ϕ_i is* lucky *if $\phi_i(d) \neq 0$ and $|\vec{\mu}| = |\vec{\mu}_i|$. Otherwise, it is* unlucky.

We remark that if the degree of $\mathbf{A}(z)$ drops under ϕ_i, then $\phi_i(\mathbf{F}(0)) = \mathbf{0}$. It follows that ϕ_i is unlucky because the first column of $\mathbf{K}^*(\vec{\mu}-\vec{e},\vec{\omega})$ is zero.

We now prove Lemma 5.2 and Theorem 5.3 which will be used for detecting whether a homomorphism is lucky.

Lemma 5.2 *Suppose $\phi_i(\mathbf{F}(0)) \neq \mathbf{0}$ and $|\vec{\mu}_i| = |\vec{\mu}|$. Then $\vec{J} \leq_{lex} \vec{J}_i$. Moreover, if $\vec{J} = \vec{J}_i$, then $\vec{\mu}$ is at least as close to w as $\vec{\mu}_i$, as defined in (13).*

Proof. The columns indexed by $\vec{J}_i$ in $\mathbf{K}_i(\vec{\mu}_i-\vec{e},\vec{\omega})$ are linearly independent over $\mathbb{D}/I_i$. Therefore, the same columns in $\mathbf{K}(\vec{\mu}_i-\vec{e},\vec{\omega})$ are also linearly independent over $\mathbb{Q}_{\mathbb{D}}$. By the definition of $\mathbf{K}^*(\vec{\mu}-\vec{e},\vec{\omega})$, it follows that $\vec{J} \leq_{lex} \vec{J}_i$.

If $\vec{J} = \vec{J}_i$, then $\phi_i\left(\det \mathbf{K}^*(\vec{\mu}_i-\vec{e},\vec{\omega})\right) = \det \mathbf{K}_i^*(\vec{\mu}_i-\vec{e},\vec{\omega}) = d_i \neq 0$. It follows that $\mathbf{K}^*(\vec{\mu}_i-\vec{e},\vec{\omega})$ is nonsingular over $\mathbb{Q}_{\mathbb{D}}$. The second part now follows [Beckermann and Labahn, 2000b, Theorem 7.3]. □

Theorem 5.3 *Suppose $\phi_i(\mathbf{F}(0)) \neq \mathbf{0}$. Then ϕ_i is lucky if and only if $\vec{\mu}_i = \vec{\mu}$ and $\vec{J}_i = \vec{J}$.*

Proof. Suppose ϕ_i is lucky. Since $\phi_i(d) \neq 0$, $\phi_i(\mathbf{K}^*(\vec{\mu}-\vec{e},\vec{\omega}))$ is nonsingular over $\mathbb{D}/I_i$. Thus, the columns of $\mathbf{K}(\vec{\mu}-\vec{e},\vec{\omega})$ indexed by $\vec{J}$ are linearly independent over $\mathbb{D}/I_i$, so that $\vec{J}_i \leq_{lex} \vec{J}$. Hence $\vec{J}_i = \vec{J}$ by Lemma 5.2. Moreover, $\vec{\mu}$ is at least as close to w as $\vec{\mu}_i$ by Lemma 5.2. On the other hand, $\vec{\mu}_i$ is the unique closest point to w, so that $\vec{\mu}_i = \vec{\mu}$.

Conversely, assume that $\vec{\mu}_i = \vec{\mu}$ and $\vec{J}_i = \vec{J}$. Then $|\vec{\mu}| = |\vec{\mu}_i|$ and $\phi_i(d) = d_i \neq 0$, so ϕ_i is lucky. □

Theorem 5.4 *If ϕ_i is lucky, then $\phi_i(\mathbf{M}(z)) = \mathbf{M}_i(z)$ and $\phi_i(\mathbf{R}(z)) = \mathbf{R}_i(z)$.*

Proof. Suppose that ϕ_i is lucky, so that $\vec{\mu} = \vec{\mu}_i$ and $\vec{J} = \vec{J}_i$ by Theorem 5.3. Then $\phi_i(\mathbf{K}^*(\vec{\mu}-\vec{e}+\vec{e}_j,\vec{\omega})) = \mathbf{K}_i^*(\vec{\mu}_i-\vec{e}+\vec{e}_j,\vec{\omega})$ for all j. It follows from (7) that $\phi_i(\mathbf{M}(z)) = \mathbf{M}_i(z)$. Finally, over $\mathbb{D}/I_i$ we have $\phi_i(\mathbf{R}(z)) = \phi_i(\mathbf{M}(z)\cdot\mathbf{F}(z)) =$

$\phi_i(\mathbf{M}(z)) \cdot \phi_i(\mathbf{F}(z)) = \mathbf{M}_i(z) \cdot \phi_i(\mathbf{F}(z)) = \mathbf{R}_i(z).$ □

This shows that Definition 5.1 is sufficient. Since $\vec{\mu}$ and $\vec{J}$ are not known a priori, we need criteria to compare the results computed under two homomorphisms and determine if one of them is unlucky.

Theorem 5.5 *Suppose $\phi_i(\mathbf{F}(0)) \neq \mathbf{0}$ and $\phi_j(\mathbf{F}(0)) \neq \mathbf{0}$. Then ϕ_i is unlucky if one of the following holds:*

(1) $|\vec{\mu}_i| = |\vec{\mu}_j|$ and $\vec{J}_i >_{lex} \vec{J}_j$;
(2) $|\vec{\mu}_i| = |\vec{\mu}_j|$, $\vec{J}_i = \vec{J}_j$, and $\vec{\mu}_j$ is closer to w than $\vec{\mu}_i$;
(3) $|\vec{\mu}_i| < |\vec{\mu}_j|$.

Proof. Conditions (1) and (2) follow from Lemma 5.2. For (3), recall that $|\vec{\mu}|$ is the maximum rank of $\mathbf{K}(\vec{\nu} - \vec{e}, \vec{\omega})$ over all $\vec{\nu}$ [Beckermann and Labahn, 2000b]. Since the rank of such matrices over $\mathbb{D}/I_i$ cannot increase, we have $|\vec{\mu}_i| \leq |\vec{\mu}|$. □

Example 5.6 *Let*

$$\mathbf{F}(z) = \begin{bmatrix} 2 + 3z + 4z^2 & z + 4z^2 \\ 1 - 2z - 2z^2 & 3 + z + z^2 \end{bmatrix}.$$

The striped Krylov matrix $\mathbf{K}((1,1),(4,4))$ is

$$\mathbf{K}((1,1),(4,4)) = \left[\begin{array}{cc|cc|cc|cc} 2 & 0 & 3 & 1 & 4 & 4 & 0 & 0 \\ 0 & 0 & 2 & 0 & 3 & 1 & 4 & 4 \\ \hline 1 & 3 & -2 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 & -2 & 1 & -2 & 1 \end{array}\right].$$

We see that over $\mathbb{Z}$, $\vec{J} = (1, 2, 3, \ldots)$. However, over $\mathbb{Z}_3$, it is $\vec{J}_3 = (1, 3, \ldots)$. Since $\vec{J}_3 >_{lex} \vec{J}$, it follows that $p = 3$ is unlucky. □

For the computation of weak Popov form, the definition of lucky homomorphisms is given as follows.

Definition 5.7 *The homomorphism ϕ_i is* lucky *if it is lucky by Definition 5.1 and the trailing coefficients of $\mathbf{R}_i(z)$ and $\mathbf{R}(z)$ have leading nonzero entries at the same position in each row. Equivalently, the sequences of pivot rows used in the last n steps of FFFG are the same over $\mathbb{D}$ and $\mathbb{D}/I_i$.*

In this case, we can also compare the results computed under two homomorphisms to detect if one of them is unlucky, by detecting which sequence of pivot rows is less desirable using the pivoting strategy (10). Equivalently, we can compare the sequences of pivot indices in the computed normal forms. The one that is lexicographically larger must correspond to an unlucky homomorphism.

6 Termination

As all coefficients in $\mathbf{M}(z)$ are Cramer solutions of the linear systems (7), they can be written as determinants of the coefficient matrices [Beckermann and Labahn, 2000b]. Note that the coefficient matrix in (7) has at most $\sigma \min(m,n)$ rows with $\sigma = mN+1$ (in each block of n columns, the rank, and hence the number of elimination steps, is at most $\min(m,n)$). By Hadamard's inequality, the size of the coefficients is $O(\alpha(\log\alpha + K))$ where $\alpha = m\min(m,n)N$ and K bounds the bit length of the coefficients in $\mathbf{F}(z) \in \mathbb{Z}[z]^{m\times n}$. When $\mathbf{F}(z) \in (\mathbb{Z}[x])[z]^{m\times n}$, the bound is $O(\alpha K)$ where K is a bound on the size of each coefficient (the product of degree in x and bit length of coefficients in x). We may terminate the modular algorithm when the size of $\prod_i p_i$ or $\prod_i(x-\alpha_i)$ exceeds the Hadamard bound.

Unfortunately, the Hadamard bound can be extremely pessimistic. A common approach is to reconstruct the results incrementally, 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 [Cabay, 1971] was first used for solving systems of linear equations with a modular algorithm without the need for verification. Once the reconstructed result of a system of linear equations have not changed for τ additional homomorphisms, it can be proven that the reconstructed result is in fact correct. The quantity τ is determined a priori from the coefficient matrix. We note that the desired transformation matrix (order basis) consists of solutions to systems of linear equations (7). The coefficient matrices have size $|\vec{\mu}| \le (mN+1)\min(m,n)$. Therefore, we have the following result.

Theorem 6.1 *Suppose $\mathbb{D} = \mathbb{Z}$ and the primes are ordered such that $p_1 < p_2 < \cdots$, and that*

$$\sum_{i=1}^{m}\sum_{k=0}^{N}\left|F_{i,j}^{(k)}\right| \le p_1\cdots p_\tau$$

for all $j = 1,\ldots,n$. Suppose that $\widetilde{\mathbf{M}}(z)$ 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 (2).

If $\mathbb{D} = \mathbb{Z}[x]$, then $\widetilde{\mathbf{M}}(z)$ and $\widetilde{\mathbf{R}}(z)$ give a solution to (2) if $\deg_x \mathbf{F}(z)_{i,j} \le \tau$ for all i,j.

Proof. The fact that $\widetilde{\mathbf{M}}(z)$ is correct is a straightforward application of the result of Cabay [1971]. To see that $\widetilde{\mathbf{R}}(z)$ is correct, note that (2) holds for $\widetilde{\mathbf{M}}(z)$ and $\widetilde{\mathbf{R}}(z)$ modulo $p_1 \cdots p_{k+\tau}$ for some $k \geq 1$ in the case of $\mathbb{D} = \mathbb{Z}$. Since $\widetilde{\mathbf{M}}(z)$ and $\widetilde{\mathbf{R}}(z)$ both have coefficients bounded by $p_1 \cdots p_k$, we see that the coefficients of $\widetilde{\mathbf{M}}(z) \cdot \mathbf{F}(z) - \widetilde{\mathbf{R}}(z) \cdot z^{\vec{\omega}}$ must have magnitudes bounded by $p_1 \cdots p_{k+\tau}$ and hence they must be zero in $\mathbb{Z}$. The case $\mathbb{D} = \mathbb{Z}[x]$ is similar. $\square$

Theorem 6.1 shows that if the reconstructed results are unchanged for a sufficient number of steps, we indeed have an order basis $\widetilde{\mathbf{M}}(z)$ of degree $\vec{\mu}'$. However, it is not sufficient to guarantee that the reconstructed result is correct. Indeed, it is possible that $\vec{\mu} \neq \vec{\mu}'$, so that $\widetilde{\mathbf{R}}(z) \neq \mathbf{R}(z)$. This happens when the matrix $\mathbf{K}^*(\vec{\mu} - \vec{e}, \vec{\omega})$ in (7) is singular under all homomorphisms so far, but is in fact nonsingular over $\mathbb{D}$. In spite of this, the computed results $\widetilde{\mathbf{M}}(z)$ and $\widetilde{\mathbf{R}}(z)$ can be used to obtain the desired normal form and the transformation matrix.

Theorem 6.2 *If the coefficients of* $\widetilde{\mathbf{R}}(z)$ *and* $\widetilde{\mathbf{M}}(z)$ *in Theorem 6.1 are reversed, we obtain a row-reduced form (or weak Popov form) of the input polynomial matrix, as well as the corresponding transformation matrix.*

Proof. Theorem 6.1 shows that $\widetilde{\mathbf{M}}(z)$ is an order basis of order $\vec{\omega} = (mN + 1) \cdot \vec{e}$ with $\widetilde{\mathbf{R}}(z)$ the corresponding residual. Since each $\mathbf{R}_i(z)$ has the correct number of zero rows (as well as the same echelon structure in the trailing coefficient for weak Popov form computation), it follows that $\widetilde{\mathbf{R}}(z)$ has the same property as well. $\square$

The early termination criteria is most useful if τ is small. In particular, if $\tau = 1$ then the proposed criteria is clearly an improvement. This is often true in practical cases. For example, when $\mathbb{D} = \mathbb{Z}$ it is often true that $m(N + 1) \max_{i,j,k} \left| F_{i,j}^{(k)} \right| \leq p_1$ (e.g. using 31-bit primes). This performs especially well if the size of the output is small, because the Hadamard bound is based on a matrix of size $O(\alpha)$, and it is much larger than the bound given in Theorem 6.1.

We also note that Theorems 6.1 and 6.2 still hold if we are only interested in the transformation matrix and only perform the reconstruction of $\mathbf{M}(z)$. This is useful in the case of computing the Popov form, and as well as other cases in which the row-reduced or weak Popov form itself is not needed (see Section 9). Furthermore, the results can be adapted to other Euclidean domains, such as the Gaussian integers, by applying the given norm and defining τ appropriately. Finally, the early termination criteria is possible only because we choose to compute the fraction-free results. If we apply a different normalization (e.g. by requiring $d = 1$ in the order basis), the images of rational numbers modulo increasing moduli may change even after the results of rational reconstruction have stabilized.

7 Alternate FFFG Termination Strategy

In the proof of Theorem 6.2, we used the fact that the order bases computed under different homomorphisms have the same order $\vec{\omega} = (mN+1) \cdot \vec{e}$. Alternatively, the FFFG elimination algorithm can be terminated as soon as the number of nonzero rows of $\mathbf{R}(z)$ is the same as the rank of the trailing coefficient of $\mathbf{R}(z)$ [Beckermann et al., 2002]. This has the advantage that the final order $\vec{\omega}$ and hence the number of elimination steps performed are typically smaller, leading to less coefficient growth and a faster execution time. We may view this as an input-sensitive strategy, so that less work is done if the input matrix is already close to being in normal form.

With this termination strategy, the final order $\vec{\omega}_i$ reached under different homomorphisms may be less than the desired order $\vec{\omega}$ reached over $\mathbb{D}$, so that we are examining solutions to systems of equations which have different numbers of unknowns. As a result, it is more difficult to detect unlucky homomorphisms. In particular, if $|\vec{\mu}_i| \neq |\vec{\mu}_j|$ we cannot determine which homomorphism is unlucky. One solution is to simply reject both homomorphisms. Otherwise, if two homomorphisms give two different orders, the one giving a smaller order must be unlucky. We also have to ensure that the zero rows in the residual $\mathbf{R}_i(z)$ occur at the same indices. These are minor drawbacks as unlucky homomorphisms are not often encountered in practice.

More importantly, Theorem 6.2 still holds. Thus, we obtain an algorithm that is sensitive to both input and output. We also note that in the worst case the final order may still be $(mN+1) \cdot \vec{e}$, so that the worst case complexity is not improved.

Example 7.1 *Let $\mathbf{A}(z)$ be the defined as in Example 3.2. If we apply FFFG over $\mathbb{Z}$ with the alternate termination strategy, we get*

$$
\begin{aligned}
d &= 2480256 = 2^7 \cdot 3^2 \cdot 2153, \\
\vec{\mu} &= (5,4,3,2), \\
\vec{J} &= (1,3,5,6,7,8,9,10,11,12,13,14,15,16), \\
\mathbf{T}(z) &= 1240128 \cdot \begin{bmatrix} 2z^2-4z-8 & 12z^2+28z+16 \\ 3z & 6z+6 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.
\end{aligned}
$$

When $p_1 = 2$, we have $\vec{\mu}_1 = (3,3,2,2)$, and when $p_2 = 3$, we get $\vec{\mu}_2 = (3,2,2,2)$. These two primes are unlucky not only because they both divide

d, but also because $\vec{J}_1, \vec{J}_2 <_{lex} \vec{J}$. Since $|\vec{\mu}_1| \neq |\vec{\mu}_2|$, we simply assume that $\vec{\mu}_1$ is unlucky and the corresponding results are discarded. The prime $p_3 = 5$ is lucky and the previous results are discarded. However, for $p_4 = 7$ we get $\vec{\mu}_4 = (4, 3, 3, 2)$. Notice that although $d \not\equiv 0 \bmod 7$, $p_4 = 7$ is an unlucky prime because $|\vec{\mu}_4| \neq |\vec{\mu}|$. This means that when FFFG is applied over $\mathbb{Z}$, in one of the steps all residuals are $0 \bmod 7$. Thus, the linear system solved by FFFG over $\mathbb{Z}_7$ is different from the one over $\mathbb{Z}$.

Since $|\vec{\mu}_3| \neq |\vec{\mu}_4|$, the previous results are also discarded. The primes $p_i = 11, 13, \ldots, 37$ are all lucky, and we can terminate the algorithm because the reconstructed results satisfy (2). If we used $p_1, p_2, \ldots = 2, 3, 5, \ldots$, then $\tau = 4$ and we need the reconstructed results to remain unchanged for 4 additional primes. On the other hand, the Hadamard bound implies that the size of the coefficients is bounded by 2^{308} and is extremely pessimistic. In practice, we are more likely to use 31-bit primes and in that case we have $\tau = 1$. □

8 Complexity Analysis

In the worst case, the number of homomorphisms required is implied by the Hadamard bound. In the case of $\mathbb{D} = \mathbb{Z}$, if we assume that the primes chosen have size $O(\log \alpha + K)$, then the number of lucky primes needed is $O(\alpha)$ where $\alpha = m \min(m, n) N$. The same reasoning shows that we need $O(\alpha K)$ lucky evaluation points when $\mathbb{D} = \mathbb{Z}[x]$.

To bound the number of unlucky homomorphisms, we see that if ϕ_i is unlucky, then either $\phi_i(d) = 0$ or $|\vec{\mu}| \neq |\vec{\mu}_i|$ by Definition 5.1. In the first case, d is the determinant of $\mathbf{K}^*(\vec{\mu}, \vec{\omega})$ of order at most $\min(m, n)(mN + 1)$. In the latter case, some column of $\mathbf{K}^*(\vec{\mu}, \vec{\omega})$ is linearly dependent on the others, so again $\phi_i(d) = 0$. If $\mathbb{D} = \mathbb{Z}$, the product of all unlucky primes must divide d, so there are at most $O(\alpha)$ unlucky primes. Similarly, there are at most $O(\alpha K)$ unlucky evaluation points if $\mathbb{D} = \mathbb{Z}[x]$. Since the number of unlucky homomorphisms is of the same order as the lucky ones, we may ignore the presence of unlucky homomorphisms in our analysis. In practice, unlucky homomorphisms are rarely encountered.

Theorem 8.1 *Let $\mathbb{D} = \mathbb{Z}$ and K be a bound on the size of the coefficients appearing in $\mathbf{F}(z)$. The worst case complexity of the modular algorithm is $O((m+n)\alpha \mathsf{M}(\alpha(\log \alpha + K))(\alpha + \log(\alpha(\log \alpha + K))))$ bit operations.*

Proof. Using the same analysis as for the FFFG elimination algorithm [Beckermann and Labahn, 2000b, Theorem 6.3] while assuming all arithmetic operations can be done in $\mathsf{M}(K)$ time, we see that the computation of $\mathbf{M}_i(z)$ and $\mathbf{R}_i(z)$ can be done in $O((m+n)\alpha^2 \mathsf{M}(K))$ operations. Moreover, we can compute $\phi_{p_i}(\mathbf{F}(z))$ in $O(mnN\mathsf{M}(K))$ bit operations. Since we need $O(\alpha)$ primes, all $\mathbf{M}_i(z)$ and $\mathbf{R}_i(z)$ can be computed in $O((m+n)\alpha^3\mathsf{M}(K))$ bit operations, or $O((m+n)\alpha^2\mathsf{M}(\alpha(\log\alpha+K)))$ bit operations since $\alpha\mathsf{M}(K)\in O(\mathsf{M}(\alpha K))$.

Finally, each coefficient in $\mathbf{M}(z)$ and $\mathbf{R}(z)$ can be reconstructed in $O(\mathsf{M}(\alpha(\log\alpha+K))\log(\alpha(\log\alpha+K)))$ by Chinese remaindering [von zur Gathen and Gerhard, 2002]. There are potentially $O(m^2\min(m,n)N)$ nonzero coefficients in $\mathbf{M}(z)$ and $O(mnN)$ nonzero coefficients in $\mathbf{R}(z)$. Combining the two parts gives the desired complexity. $\square$

When $\mathbb{D}=\mathbb{Z}[x]$, a similar analysis gives the following complexity result.

Theorem 8.2 *Let $\mathbb{D}=\mathbb{Z}[x]$ and K be a bound on the degree of the coefficients appearing in $\mathbf{F}(z)$. The worst case complexity is $O((m+n)\alpha^3K^2)$ operations in $\mathbb{Q}$.*

This shows that the modular algorithm is an order of magnitude faster than the FFFG elimination algorithm in the parameter $\alpha = m\min(m,n)N$.

9 Greatest Common Divisors

Let $\mathbf{A_1}(z)\in\mathbb{Q}_{\mathbb{D}}[z]^{m_1\times n}$ and $\mathbf{A_2}(z)\in\mathbb{Q}_{\mathbb{D}}[z]^{m_2\times n}$. Without loss of generality, we may assume that both $\mathbf{A_1}(z)$ and $\mathbf{A_2}(z)$ have coefficients in $\mathbb{D}$. We will make the standard assumption that the matrix $[\mathbf{A_1}(z)^T\ \mathbf{A_2}(z)^T]^T$ has full column rank [Kailath, 1980, page 378]. Then any greatest common right divisor (GCRD) of $\mathbf{A_1}(z)$ and $\mathbf{A_2}(z)$ is nonsingular, and it is unique up to multiplication on the left by a unimodular matrix. A GCRD of $\mathbf{A_1}(z)$ and $\mathbf{A_2}(z)$ can be obtained by computing a row-reduced or a weak Popov form of $[\mathbf{A_1}(z)^T\ \mathbf{A_2}(z)^T]^T$:

$$\mathbf{U}(z)\cdot\begin{bmatrix}\mathbf{A_1}(z)\\ \mathbf{A_2}(z)\end{bmatrix}=\begin{bmatrix}\mathbf{G}(z)\\ \mathbf{0}\end{bmatrix}, \tag{14}$$

where the GCRD $\mathbf{G}(z)$ is in row-reduced or weak Popov form. The unimodular matrix $\mathbf{U}(z)$ can be partitioned into submatrices of appropriate dimensions to obtain a solution to the *extended matrix GCRD problem*:

$$\begin{aligned}\mathbf{S_1}(z)\cdot\mathbf{A_1}(z)+\mathbf{T_1}(z)\cdot\mathbf{A_2}(z)&=\mathbf{G}(z),\\ \mathbf{S_2}(z)\cdot\mathbf{A_1}(z)+\mathbf{T_2}(z)\cdot\mathbf{A_2}(z)&=\mathbf{0}.\end{aligned} \tag{15}$$

A least common left multiple (LCLM) of $\mathbf{A_1}(z)$ and $\mathbf{A_2}(z)$ is given by $\mathbf{S_2}(z)\cdot\mathbf{A_1}(z)$, and an irreducible left matrix-fraction description of $\mathbf{A_1}(z)\cdot\mathbf{A_2}(z)^{-1}$ is given by $\mathbf{S_2}(z)^{-1}\cdot\mathbf{T_2}(z)$ [Kailath, 1980]. Since these can be obtained from the transformation matrix $\mathbf{U}(z)$ alone, we do not need to reconstruct the matrix

$\mathbf{R}(z)$ over $\mathbb{D}$ for these problems. Greatest common left divisors, least common right multiples, and irreducible right matrix-fraction descriptions can be computed by considering the transposes of the input matrices. Note that the computation of least common multiples requires a polynomial matrix multiplication on matrices smaller than the entire transformation matrix $\mathbf{U}(z)$, so that it is still worthwhile to use the output-sensitive algorithm.

We remark that when $m_1 = m_2 = n = 1$, we obtain an output-sensitive modular extended polynomial GCD algorithm with complexity $O(N^3K^2)$. When the size of the coefficients is also $O(N)$, this gives a complexity of $O(N^5)$ which is as good as the "small primes modular extended Euclidean Algorithm, single row" as described by von zur Gathen and Gerhard [2002, Table 6.5] (where multiplication with quadratic complexity is assumed).

Example 9.1 *Let $\mathbf{A}(z)$ be defined in Example 3.2, $\mathbf{A_1}(z) = \mathbf{A}(z)_{\{1,2\},*}$, and $\mathbf{A_2}(z) = \mathbf{A}(z)_{\{3,4\},*}$. Then $\mathbf{T}(z)_{\{1,2\},*}$ in Example 7.1 is a GCRD of $\mathbf{A_1}(z)$ and $\mathbf{A_2}(z)$.* □

It is well known that if $\deg \phi(a(z)) = \deg a(z)$, $\deg \phi(b(z)) = \deg b(z)$, and $\phi(a(z))$ and $\phi(b(z))$ are relatively prime in $(\mathbb{D}/I)[z]$, then $a(z)$ and $b(z)$ are relatively prime in $\mathbb{D}[z]$ [Geddes et al., 1992]. We now prove an analogous result for polynomial matrices. For simplicity we assume that the input matrices are square and nonsingular.

Theorem 9.2 *Let $\mathbf{A_1}(z), \mathbf{A_2}(z) \in \mathbb{D}[z]^{m\times m}$, $\mathbf{G}(z) \in \mathbb{D}[z]^{m\times m}$ be a GCRD of $\mathbf{A_1}(z)$ and $\mathbf{A_2}(z)$ computed by FFFG over $\mathbb{D}$, and $\mathbf{G}'(z) \in (\mathbb{D}/I)[z]^{m\times m}$ be a GCRD of $\phi(\mathbf{A_1}(z))$ and $\phi(\mathbf{A_2}(z))$ computed by FFFG over $\mathbb{D}/I$. If $\deg \phi(\det \mathbf{A_1}(z)) = \deg \det \mathbf{A_1}(z)$ and $\deg \phi(\det \mathbf{A_2}(z)) = \deg \det \mathbf{A_2}(z)$ then*

$$\deg \det \mathbf{G}(z) \le \deg \det \mathbf{G}'(z).$$

Furthermore, if $\deg \det \mathbf{G}'(z) = 0$ then $\mathbf{A_1}(z)$ and $\mathbf{A_2}(z)$ are right coprime over $\mathbb{Q}_{\mathbb{D}}$ (i.e. the GCRD is unimodular).

Proof. Let $\mathbf{Q}_1(z)$ and $\mathbf{Q}_2(z)$ be the polynomial matrices such that $\mathbf{A_1}(z) = \mathbf{Q}_1(z) \cdot \mathbf{G}(z)$ and $\mathbf{A_2}(z) = \mathbf{Q}_2(z) \cdot \mathbf{G}(z)$. Note that $\mathbf{Q}_1(z)$ and $\mathbf{Q}_2(z)$ are submatrices of the unimodular matrix $\mathbf{U}(z)$ computed by FFFG and have entries in $\mathbb{D}[z]$, and they cannot vanish in $\mathbb{D}/I$ because $\phi(\mathbf{A_1}(z))$ and $\phi(\mathbf{A_2}(z))$ are nonzero. Examining these equations over $\mathbb{D}/I$ shows that $\phi(\mathbf{G}(z))$ is a common right divisor of $\phi(\mathbf{A_1}(z))$ and $\phi(\mathbf{A_2}(z))$, so $\phi(\mathbf{G}(z))$ is a right divisor of $\mathbf{G}'(z)$. Taking determinants, we also see that $\det \mathbf{G}(z)$ is a common divisor of $\det \mathbf{A_1}(z)$ and $\det \mathbf{A_2}(z)$, so that $\deg \det \mathbf{G}(z) = \deg \phi(\det(\mathbf{G}(z)))$ by the degree assumptions on $\deg \det \mathbf{A_1}(z)$ and $\deg \det \mathbf{A_2}(z)$. Thus,

$$\deg \det \mathbf{G}(z) = \deg \phi(\det \mathbf{G}(z)) \le \deg \det \mathbf{G}'(z).$$

The second part of the theorem follows. □

Note that the theorem above can only be applied easily if the input polynomial matrices are row-reduced. In this case, the condition $\deg \phi(\det \mathbf{A}(z)) = \deg \det \mathbf{A}(z)$ can be tested quickly by checking whether $\phi(\mathrm{LC}_{row}(\mathbf{A}(z)))$ is nonsingular over $\mathbb{D}/I$. A similar test exists to determine if $\mathbf{A}(z)$ is row-reduced [Cheng, 2003, Theorem 5.16]:

Theorem 9.3 *Let $\mathbf{T}(z)$ be a row-reduced form of $\phi(\mathbf{A}(z))$ over $\mathbb{D}/I$. If $|rdeg\ \mathbf{A}(z)| = |rdeg\ \mathbf{T}(z)|$, then $\mathbf{A}(z)$ is row-reduced over $\mathbb{Q}_{\mathbb{D}}$.* □

10 Experimental Results

In this section, we show some experimental results comparing the performance of the modular algorithm and the fraction-free algorithm. Both termination strategies described in Sections 6 and 7 were implemented. All experiments were performed on the Maple 9.5 computer algebra system using a Pentium M 1.6GHz processor with 1 Gb of RAM. For $\mathbb{D} = \mathbb{Z}$, we used the `modp1` representation for polynomials for the efficient implementation of the modular algorithm.

All our experiments were performed on the computation of GCRDs of two $n \times n$ polynomial matrices. That is, the dimensions of the input polynomial matrix are $2n \times n$. In the following, κ is a bound on the coefficients in the input matrices. In the case of $\mathbb{D} = \mathbb{Z}[x]$, κ is a bound on the degree (in x) of the coefficients in the input matrices. In this case, the integer coefficients have size $O(1)$.

In the first set of experiments, we chose various values of n, N, and κ, and generated the input matrices randomly. In these cases, the input matrices are usually right coprime so that coefficient growth is the greatest. The experimental results are presented in Tables 1 and 2.

We see from the results that in the case of $\mathbb{D} = \mathbb{Z}$, the implementation of integer arithmetic in Maple 9.5 (using gmp) is very efficient but one can see that the modular algorithm becomes better than the fraction-free algorithm as the coefficient growth increases[3]. The main reason that the modular algorithm is slower for small inputs is that the overhead in the bookkeeping (e.g. loop control, extracting coefficients, etc.) in the FFFG algorithm is repeated for

[3] The same experiments ran on Maple 7 (before gmp was used in Maple) showed that the modular algorithm was superior even for smaller inputs. The fraction-free algorithm was 4–6 times slower in Maple 7, while the modular algorithm was slower by only 10–20%. We also note that gmp was used only for sufficiently large integers in Maple 9.5.

n	N	Size	FFFG (s)	Modular (s)	Ratio
1	25	386	0.261	1.174	4.480
1	50	445	0.504	1.934	3.830
1	100	1325	13.844	18.718	1.360
1	200	2765	127.747	91.407	0.714
1	400	4572	1136.155	433.241	0.380
2	25	708	3.326	15.024	4.500
2	50	1457	31.978	86.637	2.710
2	100	2692	230.528	433.523	1.880
2	200	5477	3629.549	3753.344	1.030
3	25	1194	22.279	122.893	5.520
3	50	2336	198.885	776.743	3.900
3	100	4576	2253.885	5628.713	2.500
3	200	9015	28213.511	50963.573	1.810

Table 1
Comparison of fraction-free and modular algorithms for various values of n and N for $\mathbb{D} = \mathbb{Z}$ ($\kappa = 10^4$). Also shown is the size (in number of decimal digits) of the largest coefficient in the result.

each of the computations under the $O(\alpha)$ primes. Over all primes the bookkeeping requires $O(\alpha^3)$ bit operations. This is significant for smaller inputs given that all homomorphic images can be computed in $O((m+n)\alpha^3\mathsf{M}(K))$ bit operations (see the proof of Theorem 8.1). On the other hand, the $O(\alpha^2)$ bookkeeping cost is only incurred once in the fraction-free algorithm. After some simple optimizations in the bookkeeping operations, we were able to reduce the cost of the modular algorithm by as much as 40% for smaller input matrices. As expected, the same optimizations on the fraction-free algorithm did not provide significant improvement.

In the case of $\mathbb{D} = \mathbb{Z}[x]$ the modular algorithm is already superior for small input matrices. The modular algorithm is better when coefficient growth is significant, or when arithmetic operations in $\mathbb{D}$ are not very efficient. Again, our complexity analysis is confirmed. Here, the overhead in bookkeeping is not as significant as coefficient arithmetic has a much higher cost.

In the second set of experiments, we generated random input matrices over $\mathbb{D} = \mathbb{Z}[x]$ with a known greatest common divisor of degree d. The results are shown in Table 3. As expected, the advantage of the modular algorithm is more significant when the degree of the GCRD is small because coefficient growth is the greatest in this case.

n	N	κ	Size	FFFG (s)	Modular (s)	Ratio
1	5	1	24	0.440	0.860	1.950
1	10	1	42	4.250	4.101	0.965
1	15	1	60	16.130	10.451	0.652
1	20	1	74	63.089	20.451	0.325
2	5	1	46	15.160	11.839	0.776
2	10	1	86	445.129	94.381	0.212
2	15	1	122	2361.239	355.330	0.150
2	20	1	152	5476.321	930.549	0.170
3	5	1	68	282.409	84.991	0.301
3	10	1	128	4922.710	882.271	0.179
3	15	1	182	28952.810	4631.179	0.160
5	5	1	112	2688.831	777.028	0.289
10	5	1	222	> 172800.000	52993.325	< 0.306

Table 2
Comparison of fraction-free and modular algorithms for various values of n, N, and κ for $\mathbb{D} = \mathbb{Z}[x]$. Also shown is the size (in degree in x) of the largest coefficient in the result.

n	d	Size	FFFG (s)	Modular (s)	Ratio
1	1	46	5.900	5.150	0.873
1	5	42	4.109	3.950	0.961
1	10	22	0.631	0.930	1.470
2	1	110	1271.039	204.560	0.161
2	5	86	283.510	112.641	0.398
2	10	48	35.690	21.719	0.608
3	1	176	10636.089	2635.260	0.249
3	5	127	8053.540	1012.950	0.125
3	10	76	580.631	181.110	0.312

Table 3
Comparison of fraction-free and modular algorithms for various values of n and d with $N = 15$ and $\kappa = 1$.

In the case of $\mathbb{D} = \mathbb{Z}$, we have also performed the experiments with a version of the modular algorithm that terminates only when the Hadamard bound is reached. Even when only the normal form is computed (without the transformation matrix), the algorithm is 2–3 times slower for small inputs and much slower for larger inputs compared to the modular algorithm with early termination.

11 Conclusions and Future Work

In this paper, we presented modular algorithms for computing a row-reduced form, a weak Popov form, and a Popov form of a polynomial matrix, along with the associated unimodular transformation matrix. By formulating the problem as a sequence of related linear algebra problems, we were able to define lucky homomorphisms. Normalization and bounds on the number of homomorphisms needed were also determined by examining the corresponding linear systems. The worst case complexity of the algorithm is significantly improved over the fraction-free algorithm. We also gave an early termination criteria which eliminates the need for verification of the reconstructed results. In many practical cases, the criteria provides an improvement over the commonly used approach of early termination with verification.

A limitation of our algorithms is that the properties of the unimodular transformation matrix are used in many aspects of the algorithm, including the definition of lucky homomorphisms, normalization, and early termination. We currently do not have modular algorithms in which only the normal forms are computed.

It would be interesting to see if the techniques used by Giorgi et al. [2003] can be applied to incorporate matrix multiplication into our fraction-free and modular algorithms. We are also interested in devising a modular version for the corresponding fraction-free algorithm for Ore polynomial matrix [**?**Beckermann et al., 2002]. For example, we showed that the FFFG algorithm for Ore polynomial matrices can be faster than a non-fraction-free approach [Abramov and Bronstein, 2001, 2002] when coefficient growth is significant [**?**]. We expect that techniques similar to those of Li and Nemes [1997] can be combined with those proposed in this paper.

References

S. Abramov and M. Bronstein. On solutions of linear functional systems. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, pages 1–6. ACM, 2001.

S. Abramov and M. Bronstein. Linear algebra for skew-polynomial matrices. Technical Report RR-4420, INRIA, 2002.

E. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comp.*, 22:565–578, 1968.

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.

B. Beckermann and G. Labahn. Recursiveness in matrix rational interpolation problems. *J. Comput. Appl. Math.*, 77:5–34, 1997.

B. Beckermann and G. Labahn. Effective computation of rational approximants and interpolants. *Reliable Computing*, 6(365–390), 2000a.

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, 2000b.

B. Beckermann and G. Labahn. On the fraction-free computation of column-reduced matrix polynomials via FFFG. Technical Report ANO436, Laboratoire ANO, University of Lille, 2001. Available at `http://ano.univ-lille1.fr/pub/2001/ano436.ps.Z`.

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

W. S. Brown. On Euclid's algorithm and the computation of polynomial greatest common divisors. *Journal of the ACM*, 18(4):478–504, October 1971.

S. Cabay. Exact solution of linear equations. In *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, pages 392–398, 1971.

H. Cheng. *Algorithms for Normal Forms for Matrices of Polynomials and Ore Polynomials*. PhD thesis, University of Waterloo, 2003.

K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1992.

P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, pages 135–142, 2003.

T. Kailath. *Linear Systems*. Prentice-Hall, 1980.

E. Kaltofen and M. Monagan. On the genericity of the modular polynomial GCD algorithm. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*, pages 59–66. ACM, 1999.

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.

T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, 35(4):377–401, 2003.

A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, Department of Computer Science, Swiss Federal Institute of Technology—ETH, 2000.

J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, second edition, 2002.