Journal of Symbolic Computation 46 (2011) 1–22

Contents lists available at ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

A local construction of the Smith normal form of a matrix polynomial

Jon Wilkening [1], Jia Yu

Department of Mathematics, University of California, Berkeley, CA 94720-3840, USA

ARTICLE INFO

Article history:
Received 8 September 2008
Accepted 28 June 2010
Available online 6 July 2010

Keywords:
Matrix polynomial
Canonical forms
Smith form
Jordan chain
Symbolic computation

ABSTRACT

We present an algorithm for computing a Smith form with multipliers of a regular matrix polynomial over a field. This algorithm differs from previous ones in that it computes a local Smith form for each irreducible factor in the determinant separately and then combines them into a global Smith form, whereas other algorithms apply a sequence of unimodular row and column operations to the original matrix. The performance of the algorithm in exact arithmetic is reported for several test cases.

1. Introduction

Canonical forms are a useful tool for classifying matrices, identifying their key properties, and reducing complicated systems of equations to the de-coupled, scalar case. When working with matrix polynomials over a field K, one fundamental canonical form, the Smith form, is defined. It is a diagonalization

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda) \tag{1}$$

of the given matrix $A(\lambda)$ by unimodular matrices $E(\lambda)$ and $F(\lambda)$ such that the diagonal entries $d_i(\lambda)$ of $D(\lambda)$ are monic polynomials and $d_i(\lambda)$ is divisible by $d_{i-1}(\lambda)$ for $i \geq 2$.

This factorization has various applications. The most common one (Gohberg et al., 1982; Kailath, 1980; Neven and Praagman, 1993) involves solving the system of differential equations

$$A^{(q)}\frac{\mathrm{d}^q x}{\mathrm{d}t^q} + \cdots + A^{(1)}\frac{\mathrm{d}x}{\mathrm{d}t} + A^{(0)}x = f(t), \tag{2}$$

E-mail addresses: wilken@math.berkeley.edu (J. Wilkening), jiay@math.berkeley.edu (J. Yu).
[1] Tel.: +1 510 643 7990; fax: +1 510 486 6199.

0747-7171/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2010.06.025

where $A^{(0)}, \ldots, A^{(q)}$ are $n \times n$ matrices over $\mathbb{C}$. For brevity, we denote this system by $A(\mathrm{d}/\mathrm{d}t)x = f$, where $A(\lambda) = A^{(0)} + A^{(1)}\lambda + \cdots + A^{(q)}\lambda^q$. Assume for simplicity that $A(\lambda)$ is regular, i.e. $\det[A(\lambda)]$ is not identically zero, and that (1) is a Smith form of $A(\lambda)$. The system (2) is then equivalent to

$$\begin{pmatrix} d_1(\frac{\mathrm{d}}{\mathrm{d}t}) & & \\ & \ddots & \\ & & d_n(\frac{\mathrm{d}}{\mathrm{d}t}) \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix},$$

where $y = F(\mathrm{d}/\mathrm{d}t)x(t)$ and $g = E^{-1}(\mathrm{d}/\mathrm{d}t)f(t)$. Note that $E^{-1}(\lambda)$ is a matrix polynomial over $\mathbb{C}$ due to the unimodularity of $E(\lambda)$. This system splits into n independent scalar ordinary differential equations

$$d_i\left(\frac{\mathrm{d}}{\mathrm{d}t}\right) y_i(t) = g_i(t), \quad 1 \le i \le n,$$

and the solution of (2) is then given by $x = F^{-1}(\mathrm{d}/\mathrm{d}t)y$, where $F^{-1}(\lambda)$ is also a matrix polynomial over $\mathbb{C}$.

Another important application of the Smith form concerns the study of the algebraic structural properties of systems in linear control theory (Kailath, 1980; Rosenbrock, 1970; Van Dooren and Dewilde, 1983). For example, the zeros and poles of a multivariable transfer function $H(s)$ are revealed by the Smith–McMillan form of $H(s)$, which is a close variant of the Smith form, but for rational (as opposed to polynomial) matrices. In many applications, one only needs to compute a minimal basis for the kernel of a matrix polynomial. Specialized algorithms (Neven and Praagman, 1993; Zúniga Anaya and Henrion, 2009) have been developed for this sub-problem of the Smith form calculation.

Smith forms of linear matrix polynomials (i.e. matrix pencils) are related to the concept of similarity of matrices. A fundamental theorem in matrix theory (Gantmacher, 1960; Gohberg et al., 1982) states that two square matrices A and B over a field K are similar if and only if their characteristic matrix polynomials $\lambda I - A$ and $\lambda I - B$ have the same Smith form $D(\lambda)$. Other applications of this canonical form include finding the Frobenius form (Villard, 1994, 1997) of a matrix A over a field by computing the invariant factors of the matrix pencil $\lambda I - A$.

Many algorithms have been developed for the computation of canonical forms of matrix polynomials in floating point arithmetic. One common approach involves finding an equivalent linear matrix pencil with the same finite zeros as the original matrix polynomial and a closely related Smith form (Van Dooren and Dewilde, 1983). The Kronecker form (Van Dooren, 1979; Van Dooren and Dewilde, 1983; Demmel, 1997) of the matrix pencil is then computed to determine the eigenstructure of the original polynomial matrix. Another approach centers around computing the local spectral structure of a matrix polynomial at a single complex root, λ_0, of the characteristic determinant (Gohberg et al., 1993; Wilkening, 2007). These methods usually boil down to computing kernels of nested Toeplitz matrices (Wilkening, 2007; Zúniga Anaya and Henrion, 2009). One advantage of this local approach over the global matrix pencil approach is that only a few terms in an expansion of the matrix polynomial in powers of $\lambda - \lambda_0$ are needed to compute the spectral behavior. This can lead to a significant computational savings, and also allows for generalization from matrix polynomials to analytic matrix functions (Gohberg et al., 1993; Wilkening, 2007). Such local canonical forms can be used to efficiently compute successive terms in the Laurent expansion of the inverse of an analytic matrix (Avrachenkov et al., 2001; Wilkening, 2007). Backward stability analysis of the effect of roundoff error may be found in Van Dooren and Dewilde (1983), Wilkening (2007), Zúniga Anaya and Henrion (2009). A geometric approach to the perturbation theory of matrix pencils is discussed in Edelman et al. (1997).

The symbolic computation of Smith forms of matrices over $\mathbb{Q}[\lambda]$ is also a widely studied topic. Kannan (1985) gave a method for computing the Smith form with repeated triangularizations of the matrix polynomial over $\mathbb{Q}$. Kaltofen et al. (1987) gave the first polynomial time algorithm for the Smith form (without multipliers) using the Chinese remainder theorem. A new class of probabilistic algorithms (the Monte Carlo algorithms) was proposed by Kaltofen et al. (1987, 1990). They showed that by multiplying the given matrix polynomial by a randomly generated constant matrix on the right, the Smith form with multipliers may be obtained with high probability by two steps of computation of

the Hermite form. A Las Vegas algorithm given by Storjohann (1994); Storjohann and Labahn (1997) significantly improved the complexity by rapidly checking the correctness of the result of the KKS algorithm. Villard (1993, 1995) established the first deterministic polynomial-time method to obtain the Smith form with multipliers by explicitly computing a good-conditioning matrix that replaces the random constant matrix in the Las Vegas algorithm. Villard also applied the method of Labhalla et al. (1996) to obtain useful complexity bounds for the algorithm.

We propose a new deterministic algorithm for the symbolic computation of Smith forms of matrix polynomials over a field in Section 3. Our approach differs from previous methods in that we begin by constructing local diagonal forms that we later combine to obtain a (global) post-multiplier. Although we do not discuss complexity bounds, we compare the performance of our algorithm to Villard's method with good conditioning in Section 4, and discuss the reasons for the increase in speed. The new algorithm is also easy to parallelize. In the Appendix, we present an algebraic framework that connects this work to Wilkening (2007), and give a variant of the algorithm in which all operations are done in the field K rather than manipulating polynomials directly.

As mentioned above, local canonical forms have been used successfully to study the structure of a matrix polynomial near a single root $\lambda_0 \in \mathbb{C}$ of the characteristic determinant. An important point that has been neglected in the literature is that these roots λ_0 may not be expressible in radicals, or may involve such complicated expressions that current algorithms can only be carried out in floating point arithmetic. A major goal of this paper is to develop a machinery for computing local forms for all the complex roots of a $\mathbb{Q}$-irreducible factor $p(\lambda)$ of the characteristic determinant simultaneously, without having to resort to floating point arithmetic at each root separately. This is done by working over the fields $\mathbb{Q}$ or $\mathbb{Q} + i\mathbb{Q}$ rather than $\mathbb{R}$ or $\mathbb{C}$ when computing local forms.

2. Preliminaries

In this section, we describe the theory of Smith forms of matrix polynomials over a field K, which follows the definition in Gohberg et al. (1982) over $\mathbb{C}$. In practice, K will be $\mathbb{Q}$, $\mathbb{Q} + i\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$, but it is convenient to deal with all these cases simultaneously. We also give a brief review of the theory of Jordan chains as well as Bézout's identity, which play an important role in our algorithm for computing Smith forms of matrix polynomials.

2.1. Smith forms

Suppose $A(\lambda) = \sum_{k=0}^{q} A^{(k)}\lambda^k$ is an $n \times n$ matrix polynomial, where $A^{(k)}$ are $n \times n$ matrices whose entries are in a field K. Assuming that $A(\lambda)$ is *regular*, i.e. the determinant of $A(\lambda)$ is not identically zero, the following theorem is proved (for $K = \mathbb{C}$) in Gohberg et al. (1982).

Theorem 1. *There exist matrix polynomials $E(\lambda)$ and $F(\lambda)$ over K of size $n \times n$, with constant nonzero determinants, such that*

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda), \qquad D(\lambda) = \operatorname{diag}[d_1(\lambda), \ldots, d_n(\lambda)], \tag{3}$$

where $D(\lambda)$ is a diagonal matrix with monic scalar polynomials $d_i(\lambda)$ over K such that $d_i(\lambda)$ is divisible by $d_{i-1}(\lambda)$.

Since $E(\lambda)$ and $F(\lambda)$ have constant nonzero determinants, (3) is equivalent to

$$U(\lambda)A(\lambda)V(\lambda) = D(\lambda), \tag{4}$$

where $U(\lambda) := E(\lambda)^{-1}$ and $V(\lambda) := F(\lambda)^{-1}$ are also matrix polynomials over K.

Definition 2. The representation in (3) or (4), or often $D(\lambda)$ alone, is called a *Smith form* of $A(\lambda)$. The matrices $U(\lambda)$, $V(\lambda)$ are known as *multipliers*. Square matrix polynomials with constant nonzero determinants like $E(\lambda)$ and $F(\lambda)$ are called *unimodular*.

The diagonal matrix $D(\lambda)$ in the Smith form is unique, while the representation (3) is not. Suppose that

$$\Delta(\lambda) := \det[A(\lambda)] \tag{5}$$

can be decomposed into prime elements $p_1(\lambda), \ldots, p_l(\lambda)$ in the principal ideal domain $K[\lambda]$, that is, $\Delta(\lambda) = c\prod_{j=1}^{l} p_j(\lambda)^{\kappa_j}$ where $c \neq 0$ is in the field K, $p_j(\lambda)$ is monic and irreducible, and κ_j are positive integers for $j = 1, \ldots, l$. Then the $d_i(\lambda)$ are given by

$$d_i(\lambda) = \prod_{j=1}^{l} p_j(\lambda)^{\kappa_{ji}}, \quad (1 \le i \le n)$$

for some integers $0 \le \kappa_{j1} \le \cdots \le \kappa_{jn}$ satisfying $\sum_{i=1}^{n} \kappa_{ji} = \kappa_j$ for $j = 1, \ldots, l$.

We now define a *local Smith form* for $A(\lambda)$ at $p(\lambda)$. Let $p(\lambda) = p_j(\lambda)$ be one of the irreducible factors of $\Delta(\lambda)$ and define $\alpha_i = \kappa_{ji}$, $\mu = \kappa_j$. Generalizing the case that $p(\lambda) = \lambda - \lambda_j$, we call μ the algebraic multiplicity of $p(\lambda)$.

Theorem 3. *Suppose $A(\lambda)$ is an $n \times n$ matrix over $K[\lambda]$ and $p(\lambda)$ is an irreducible factor of $\Delta(\lambda)$. There exist $n \times n$ matrix polynomials $E(\lambda)$ and $F(\lambda)$ such that*

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda), \qquad D(\lambda) = \operatorname{diag}[p(\lambda)^{\alpha_1}, \ldots, p(\lambda)^{\alpha_n}], \tag{6}$$

where $0 \le \alpha_1 \le \cdots \le \alpha_n$ are non-negative integers and $p(\lambda)$ does not divide $\det[E(\lambda)]$ *or* $\det[F(\lambda)]$.

$E(\lambda)$ and $F(\lambda)$ are not uniquely determined in a local Smith form. In particular, we can impose the additional requirement that $F(\lambda)$ be unimodular by absorbing the missing parts of $D(\lambda)$ in Theorem 1 into $E(\lambda)$. Then the local Smith form of $A(\lambda)$ at $p(\lambda)$ is given by

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \tag{7}$$

where $V(\lambda) := F(\lambda)^{-1}$ is a matrix polynomial.

2.2. Multiplication and division in R/pR

We define $R = K[\lambda]$ and $M = R^n$. Note that R is a principal ideal domain and M is a free R-module of rank n. Suppose p is a prime element in R. Since p is irreducible, R/pR is a field and M/pM is a vector space over this field.

Multiplication and division in R/pR are easily carried out using the companion matrix of p. If we set $s := \deg p$ and define $\gamma : K^s \to R/pR$ by

$$\gamma(x)(\lambda) = x^{(0)} + \cdots + \lambda^{s-1}x^{(s-1)} + pR, \quad x = \left(x^{(0)}; \ldots; x^{(s-1)}\right) \in K^s, \tag{8}$$

we can pull back the field structure of R/pR to K^s to obtain

$$\begin{aligned} xy = \gamma(x)(S)y &= [x^{(0)}I + x^{(1)}S + \cdots + x^{(s-1)}S^{s-1}]y \\ &= [y, Sy, \ldots, S^{s-1}y]x = [x, Sx, \ldots, S^{s-1}x]y \end{aligned} \tag{9}$$

and $x/y = [y, Sy, \ldots, S^{s-1}y]^{-1}x$, where

$$S = \begin{pmatrix} 0 & \ldots & 0 & -a_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & -a_{s-2} \\ 0 & & 1 & -a_{s-1} \end{pmatrix} \tag{10}$$

is the companion matrix of $p(\lambda) = a_0 + a_1\lambda + \cdots + a_{s-1}\lambda^{s-1} + \lambda^s$. Note that S represents multiplication by λ in R/pR. The matrix $[y, Sy, \ldots, S^{s-1}y]$ is invertible when $y \neq 0$ since a non-trivial vector x in its kernel would lead to non-zero polynomials $\gamma(x), \gamma(y) \in R/pR$ whose product is zero (mod p), which is impossible as p is irreducible.

2.3. Jordan chains

Finding a local Smith form of a matrix polynomial over $\mathbb{C}$ at $p(\lambda) = \lambda - \lambda_0$ is equivalent to finding a canonical system of Jordan chains (Gohberg et al., 1993; Wilkening, 2007) for $A(\lambda)$ at λ_0. We now generalize the notion of Jordan chain to the case of an irreducible polynomial over a field K.

Definition 4. Suppose $A(\lambda)$ is an $n \times n$ matrix polynomial over a field K, $p(\lambda)$ is irreducible in $K[\lambda]$, and $\alpha \geq 1$ is an integer. A vector polynomial $x(\lambda) \in M = K[\lambda]^n$ is called a *root function of order* α for $A(\lambda)$ at $p(\lambda)$ if

$$A(\lambda)x(\lambda) = O(p(\lambda)^\alpha) \tag{11}$$

and $p(\lambda) \nmid x(\lambda)$. The meaning of (11) is that each component of $A(\lambda)x(\lambda)$ is divisible by $p(\lambda)^\alpha$. If the root function $x(\lambda)$ has the form

$$x(\lambda) = x^{(0)}(\lambda) + p(\lambda)x^{(1)}(\lambda) + \cdots + p(\lambda)^{\alpha-1}x^{(\alpha-1)}(\lambda) \tag{12}$$

with $\deg x^{(k)}(\lambda) < s := \deg p(\lambda)$, the coefficients $x^{(k)}(\lambda)$ are said to form a *Jordan chain of length* α for $A(\lambda)$ at $p(\lambda)$. A root function can always be converted to the form (12) by truncating or zero-padding its expansion in powers of $p(\lambda)$. If K can be embedded in $\mathbb{C}$, (11) implies that over $\mathbb{C}$, $x(\lambda)$ is a root function of $A(\lambda)$ of order α at each root λ_j of $p(\lambda)$ simultaneously.

Definition 5. Several vector polynomials $\{x_j(\lambda)\}_{j=1}^\nu$ form a *system of root functions* at $p(\lambda)$ if

$$\begin{aligned} &1.\ A(\lambda)x_j(\lambda) = O(p(\lambda)^{\alpha_j}), \quad (\alpha_j \geq 1,\ 1 \leq j \leq \nu) \\ &2.\ \text{The set } \{\dot{x}_j(\lambda)\}_{j=1}^\nu \text{ is linearly independent in } M/pM \text{ over } R/pR, \\ &\qquad\qquad \text{where } R = K[\lambda],\ M = R^n,\ \dot{x}_j = x_j + pM. \end{aligned} \tag{13}$$

It is called *canonical* if (1) $\nu = \dim\ker \dot{A}$, where $\dot{A}$ is the linear operator on M/pM induced by $A(\lambda)$; (2) $x_1(\lambda)$ is a root function of maximal order α_1; and (3) for $i > 1$, $x_i(\lambda)$ has maximal order α_i among all root functions $x(\lambda) \in M$ such that $\dot{x}$ is linearly independent of $\dot{x}_1, \ldots, \dot{x}_{i-1}$ in M/pM. The integers $\alpha_1 \geq \cdots \geq \alpha_\nu$ are uniquely determined by $A(\lambda)$. We call ν the *geometric multiplicity* of $p(\lambda)$.

Definition 6. An *extended system of root functions* $x_1(\lambda), \ldots, x_n(\lambda)$ is a collection of vector polynomials satisfying (13) with ν replaced by n and α_j allowed to be zero. The extended system is said to be *canonical* if, as before, the orders α_j are chosen to be maximal among root functions not in the span of previous root functions in M/pM. The resulting sequence of numbers $\alpha_1 \geq \cdots \geq \alpha_\nu \geq \alpha_{\nu+1} = \cdots = \alpha_n = 0$ is uniquely determined by $A(\lambda)$.

Given such a system (not necessarily canonical), we define the matrices

$$V(\lambda) = [x_1(\lambda), \ldots, x_n(\lambda)], \tag{14}$$

$$D(\lambda) = \operatorname{diag}[p(\lambda)^{\alpha_1}, \ldots, p(\lambda)^{\alpha_n}], \tag{15}$$

$$E(\lambda) = A(\lambda)V(\lambda)D(\lambda)^{-1}. \tag{16}$$

$E(\lambda)$ is a polynomial since column j of $A(\lambda)V(\lambda)$ is divisible by $p(\lambda)^{\alpha_j}$. The following theorem shows that aside from a reversal of the convention for ordering the α_j, finding a local Smith form is equivalent to finding an extended canonical system of root functions:

Theorem 7. *The following three conditions are equivalent:*

(1) *the columns $x_j(\lambda)$ of $V(\lambda)$ form an extended canonical system of root functions for $A(\lambda)$ at $p(\lambda)$ (up to a permutation of columns).*
(2) $p(\lambda) \nmid \det[E(\lambda)]$.
(3) $\sum_{j=1}^n \alpha_j = \mu$, *where μ is the algebraic multiplicity of $p(\lambda)$ in $\Delta(\lambda)$.*

This theorem is proved e.g. in Gohberg et al. (1993) for the case that $K = \mathbb{C}$. The proof over a general field K is identical, except that the following lemma is used in place of invertibility of $E(\lambda_0)$. This lemma also plays a fundamental role in our construction of Jordan chains and local Smith forms.

Lemma 8. *Suppose K is a field, p is an irreducible polynomial in $R = K[\lambda]$, and $E = [y_1, \ldots, y_n]$ is an $n \times n$ matrix with columns $y_j \in M = R^n$. Then $p \nmid \det(E) \Leftrightarrow \{\dot{y}_1, \ldots, \dot{y}_n\}$ are linearly independent in M/pM over R/pR.*

Proof. The $\dot{y}_j$ are linearly independent iff the determinant of $\dot{E}$ (considered as an $n \times n$ matrix with entries in the field R/pR) is non-zero. But

$$\det \dot{E} = \det E + pR, \tag{17}$$

where $\det E$ is computed over R. The result follows. □

2.4. Bézout's identity

As $K[\lambda]$ is a principal ideal domain, Bézout's Identity holds, which is our main tool for combining local Smith forms into a single global Smith form. We define the notation $\gcd(f_1, \ldots, f_l)$ to be 0 if each f_j is zero, and the monic greatest common divisor (GCD) of $f_1, \ldots, f_l$ over $K[\lambda]$, otherwise.

Theorem 9 (*Bézout's Identity*)**.** *For any two polynomials f_1 and f_2 in $K[\lambda]$, where K is a field, there exist polynomials g_1 and g_2 in $K[\lambda]$ such that*

$$g_1 f_1 + g_2 f_2 = \gcd(f_1, f_2). \tag{18}$$

Bézout's Identity can be extended to combinations of more than two polynomials:

Theorem 10 (*Generalized Bézout's Identity*)**.** *For any scalar polynomials $f_1, \ldots, f_l$ in $K[\lambda]$, there exist polynomials $g_1, \ldots, g_l$ in $K[\lambda]$ such that*

$$\sum_{j=1}^{l} g_j f_j = \gcd(f_1, \ldots, f_l).$$

The polynomials g_j are called the Bézout coefficients *of $\{f_1, \ldots, f_l\}$.*

In particular, suppose we have l distinct prime elements $\{p_1, \ldots, p_l\}$ in $K[\lambda]$, and f_j is given by $f_j = \prod_{k \neq j}^{l} p_k^{\beta_k}$, where $\beta_1, \ldots, \beta_l$ are given positive integers and the notation $\prod_{k \neq j}^{l}$ indicates a product over all indices $k = 1, \ldots, l$ except $k = j$. Then $\gcd(f_1, \ldots, f_l) = 1$, and we can find $g_1, \ldots, g_l$ in $K[\lambda]$ such that

$$\sum_{j=1}^{l} g_j f_j = 1. \tag{19}$$

In this case, the polynomials g_j are uniquely determined by requiring $\deg(g_j) < s_j \beta_j$, where $s_j = \deg(p_j)$. The formula (19) modulo p_k shows that g_k is not divisible by p_k.

The Bézout coefficients are easily computed using the extended Euclidean algorithm (Cormen et al., 2001). In practice, we use MatrixPolynomialAlgebra[HermiteForm] in Maple to find a unimodular matrix Q such that

$$Q \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_l \end{pmatrix} = \begin{pmatrix} r \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{20}$$

where $r = \gcd(f_1, \ldots, f_l) = 1$. The first row of Q is $[g_1, \ldots, g_l]$. One could avoid computing the remaining rows of Q by storing the sequence of elementary unimodular operations required to reduce $[f_1; \ldots; f_l]$ to $[r; 0; \ldots; 0]$ and applying them to the row vector $[1, 0, \ldots, 0]$ from the right to obtain $[g_1, \ldots, g_l]$.

3. An algorithm for computing a (global) Smith form

In this section, we describe an algorithm for computing a Smith form of a regular $n \times n$ matrix polynomial $A(\lambda)$ over a field K. We have in mind the case where $K = \mathbb{C}, \mathbb{R}, \mathbb{Q}$ or $\mathbb{Q} + i\mathbb{Q} \subset \mathbb{C}$, but the construction works for any field. The basic procedure follows several steps, which will be explained further below:

- Step 0. Compute $\Delta(\lambda) = \det[A(\lambda)]$ and decompose it into irreducible monic factors in $K[\lambda]$,
$$\Delta(\lambda) = \text{const} \cdot p_1(\lambda)^{\kappa_1} \dots p_l(\lambda)^{\kappa_l}. \tag{21}$$
- Step 1. Compute a local Smith form
$$A(\lambda)V_j(\lambda) = E_j(\lambda)\,\text{diag}\big[p_j(\lambda)^{\kappa_{j1}}, \dots, p_j(\lambda)^{\kappa_{jn}}\big] \tag{22}$$
for each factor $p_j(\lambda)$ of $\Delta(\lambda)$.
- Step 2. Find a linear combination $B_n(\lambda) = \sum_{j=1}^{l} g_j(\lambda) f_j(\lambda) V_j(\lambda)$ using the Bézout coefficients of $f_j(\lambda) = \prod_{k\neq j}^{l} p_k(\lambda)^{\kappa_{kn}}$ so that the columns of $B_n(\lambda)$ form an extended canonical system of root functions for $A(\lambda)$ with respect to each $p_j(\lambda)$.
- Step 3. Eliminate extraneous zeros from $\det\big[A(\lambda)B_n(\lambda)\big]$ by finding a unimodular matrix $V(\lambda)$ such that $B_1(\lambda) = V(\lambda)^{-1}B_n(\lambda)$ is lower triangular. We will show that $A(\lambda)V(\lambda)$ is then of the form $E(\lambda)D(\lambda)$ with $E(\lambda)$ unimodular and $D(\lambda)$ as in (3).

Remark 11. Once the local Smith forms are known, the diagonal entries of the matrix polynomial $D(\lambda)$ are given by
$$d_i(\lambda) = \prod_{j=1}^{l} p_j(\lambda)^{\kappa_{ji}}, \quad i = 1, \dots, n.$$
This allows us to order the columns once and for all in Step 2.

3.1. A local Smith form algorithm (step 1)

In this section, we show how to generalize the construction in Wilkening (2007) for finding a canonical system of Jordan chains for an analytic matrix function $A(\lambda)$ over $\mathbb{C}$ at $\lambda_0 = 0$ to finding a local Smith form for a matrix polynomial $A(\lambda)$ with respect to an irreducible factor $p(\lambda)$ of $\Delta(\lambda) = \det[A(\lambda)]$. The new algorithm reduces to the "exact arithmetic" version of the previous algorithm when $p(\lambda) = \lambda$. In the Appendix, we present a variant of the algorithm that is easier to implement than the current approach, and is closer in spirit to the construction in Wilkening (2007), but is less efficient by a factor of $s = \deg p$.

Our goal is to find matrices $V(\lambda)$ and $E(\lambda)$ such that $p(\lambda)$ does not divide $\det[V(\lambda)]$ or $\det[E(\lambda)]$, and such that
$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \qquad D(\lambda) = \text{diag}[p(\lambda)^{\alpha_1}, \dots, p(\lambda)^{\alpha_n}], \tag{23}$$
where $0 \leq \alpha_1 \leq \cdots \leq \alpha_n$. In our construction, $V(\lambda)$ will be unimodular, which reduces the work in Step 3 of the high level algorithm, the step in which extraneous zeros are removed from the determinant of the combined local Smith forms.

We start with $V(\lambda) = I_{n\times n}$ and perform a sequence of column operations on $V(\lambda)$ that preserves its determinant (up to a sign) and systematically increases the orders α_i in $D(\lambda)$ in (23) until $\det[E(\lambda)]$ no longer contains a factor of $p(\lambda)$. This can be considered a "breadth first" construction of a canonical system of Jordan chains, in contrast to the "depth first" procedure described in Definition 5 above.

The basic algorithm is presented in Fig. 1. The idea is to run through the columns of V in turn and "accept" columns whenever the leading term of the residual $A(\lambda)x_i(\lambda)$ is linearly independent of its predecessors; otherwise we find a linear combination of previously accepted columns to cancel this leading term and cyclically rotate the column to the end for further processing. Note that for each k, we cycle through each unaccepted column exactly once: after rotating a column to the end, it will not become active again until k has increased by one. At the start of the *while* loop, we have the invariants

Algorithm 1. (Local Smith form, preliminary version)

$k = 0,\ i = 1,\ V = [x_1, \ldots, x_n] = I_{n\times n}$
while $i \le n$
  $r_{k-1} = n + 1 - i$ $r_{k-1} :=$ *dim. of space of J. chains of length* $\ge k$
  for $j = 1, \ldots, r_{k-1}$
    $y_i = \mathrm{rem}(\mathrm{quo}(Ax_i, p^k), p)$ *define* y_i *so* $Ax_i = p^k y_i + O(p^{k+1})$
    if the set $\{\dot y_1, \ldots, \dot y_i\}$ is linearly independent in M/pM over R/pR
      $\alpha_i = k,\ i = i + 1$ *accept* x_i *and* y_i*, define* α_i
    else
      find $\dot a_1, \ldots, \dot a_{i-1} \in R/pR$ so that $\dot y_i - \sum_{m=1}^{i-1} \dot a_m \dot y_m = \dot 0$
    ⋆ $x_i^{(\mathrm{new})} = x_i^{(\mathrm{old})} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m x_m$
      $x_{\mathrm{tmp}} = x_i,\ x_m = x_{m+1},\ (m = i, \ldots, n-1),\ x_n = x_{\mathrm{tmp}}$
    end if
  end for j
  $k = k + 1$
end while
$\beta = k - 1,\ r_\beta = 0$ $\beta := \alpha_n =$ *maximal Jordan chain length*

Fig. 1. Algorithm for computing a local Smith form. Here $\mathrm{quo}(\cdot,\cdot)$ and $\mathrm{rem}(\cdot,\cdot)$ are the quotient and remainder of polynomials: $g = \mathrm{quo}(f, p), r = \mathrm{rem}(f, p) \Leftrightarrow f = gp + r, \deg r < \deg p$.

(1) Ax_m is divisible by p^k, $\quad (i \le m \le n)$.
(2) $Ax_m = p^{\alpha_m} y_m + O(p^{\alpha_m+1}), \quad (1 \le m < i)$.
(3) if $i \ge 2$ then $\{\dot y_m\}_{m=1}^{i-1}$ is linearly independent in M/pM over R/pR.

The third property is guaranteed by the *if* statement, and the second property follows from the first due to the definition of α_i and y_i in the algorithm. The first property is obviously true when $k = 0$; it continues to hold each time k is incremented due to step ⋆, after which $Ax_i^{(\mathrm{new})}$ is divisible by p^{k+1}:

$$Ax_i^{(\mathrm{old})} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m A x_m = p^k y_i + O(p^{k+1}) - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m \Big(p^{\alpha_m} y_m + O(p^{\alpha_m+1}) \Big)$$
$$= p^k \Big(y_i - \sum_{m=1}^{i-1} a_m y_m \Big) + O(p^{k+1}) = O(p^{k+1}).$$

This equation is independent of which polynomials $a_m \in R$ are chosen to represent $\dot a_m \in R/pR$, but different choices will lead to different (equally valid) Smith forms; in practice, we choose the unique representatives such that $\deg a_m < s$, where

$$s = \deg p. \tag{24}$$

This choice of the a_m leads to two additional invariants of the *while* loop, namely

(4) $\deg x_m \le \max(s\alpha_m - 1, 0), \quad (1 \le m < i)$,
(5) $\deg x_m \le \max(sk - 1, 0), \quad (i \le m \le n)$,

which are easily proved inductively by noting that

$$\deg(p^{k-\alpha_m} a_m x_m) \le s(k - \alpha_m) + (s - 1) + \deg(x_m) \le s(k + 1) - 1. \tag{25}$$

The *while* loop eventually terminates, for at the end of each loop (after k has been incremented) we have produced a unimodular matrix $V(\lambda)$ such that

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \qquad D = \mathrm{diag}[p^{\alpha_1}, \ldots, p^{\alpha_{i-1}}, \underbrace{p^k, \ldots, p^k}_{r_{k-1}\ \text{times}}]. \tag{26}$$

Hence, the algorithm must terminate before k exceeds the algebraic multiplicity μ of $p(\lambda)$ in $\Delta(\lambda)$:

$$k \le \left(\sum_{m=1}^{i-1} \alpha_i\right) + (n+1-i)k \le \mu, \qquad \Delta(\lambda) = f(\lambda)p(\lambda)^\mu, \quad p \nmid f. \tag{27}$$

In fact, we can avoid the last iteration of the *while* loop if we change the test to

$$\textbf{while } \left[\left(\sum_{m=1}^{i-1} \alpha_i\right) + (n+1-i)k\right] < \mu$$

and change the last line to

$$\beta = k, \qquad \alpha_m = k, \quad (i \le m \le n), \qquad r_{\beta-1} = n+1-i, \qquad r_\beta = 0.$$

We know the remaining columns of V will be accepted without having to compute the remaining y_i or check them for linear independence. When the algorithm terminates, we will have found a unimodular matrix $V(\lambda)$ satisfying (23) such that the columns of

$$\dot{E}(\lambda) = [\dot{y}_1(\lambda), \ldots, \dot{y}_n(\lambda)]$$

are linearly independent in M/pM over R/pR. By Lemma 8, $p(\lambda) \nmid \det[E(\lambda)]$, as required.

To implement the algorithm, we must find an efficient way to compute y_i, test for linear independence in M/pM, find the coefficients a_m to cancel the leading term of the residual, and update x_i. Motivated by the construction in Wilkening (2007), we interpret the loop over j in Algorithm 1 as a single nullspace calculation.

To this end, we define $R_l = \{a \in R \ : \ \deg a < l\}$ and $M_l = R_l^n$, both viewed as vector spaces over K. Then we have an isomorphism Λ of vector spaces over K

$$\Lambda : (M_s)^k \to M_{sk}, \tag{28}$$
$$\Lambda(x^{(0)}; \ldots; x^{(k-1)}) = x^{(0)} + px^{(1)} + \cdots + p^{k-1}x^{(k-1)}.$$

At times it will be convenient to identify R_{ls} with R/p^lR and M_{ls} with M/p^lM to obtain ring and module structures for these spaces. We also expand

$$A = A^{(0)} + pA^{(1)} + \cdots + p^qA^{(q)}, \tag{29}$$

where $A^{(j)}$ is an $n \times n$ matrix with entries in R_s.

By invariants (4) and (5) of the *while* loop in Algorithm 1, we may write $x_i = \Lambda(x_i^{(0)}; \ldots; x_i^{(\alpha)})$ with $\alpha = \max(k-1, 0)$. Since Ax_i is divisible by p^k, we have

$$y_i = \operatorname{rem}(\operatorname{quo}(Ax_i, p^k), p) = \sum_{j=0}^{k} \operatorname{rem}(A^{(k-j)}x_i^{(j)}, p) + \sum_{j=0}^{k-1} \operatorname{quo}(A^{(k-1-j)}x_i^{(j)}, p). \tag{30}$$

The matrix–vector multiplications $A^{(k-j)}x_i^{(j)}$ are done in the ring R (leading to vector polynomials of degree $\le 2s-2$) before the quotient and remainder are taken. When $k = 0$, the second sum should be omitted, and when $k \ge 1$, the $j = k$ term in the first sum can be dropped since $x_i^{(k)} = 0$ in the algorithm. It is convenient to write (30) in matrix form. If $k = 0$ we have

$$[y_1, \ldots, y_n] = A^{(0)}. \tag{31}$$

If $k \ge 1$, suppose we have already computed the $nk \times r_{k-1}$ matrix X_{k-1} with columns

$$X_{k-1}(:, m+1-i) = \left(x_m^{(0)}; \ldots; x_m^{(k-1)}\right), \qquad i \le m \le n. \tag{32}$$

Note that $\Lambda(X_{k-1})$ (acting column by column) contains the last r_{k-1} columns of $V(\lambda)$ at the start of the *while* loop in Algorithm 1. Then by (30),

$$[y_i, \ldots, y_n] = \operatorname{rem}([A^{(k)}, \ldots, A^{(1)}]X_{k-1}, p) + \operatorname{quo}([A^{(k-1)}, \ldots, A^{(0)}]X_{k-1}, p). \tag{33}$$

As before, the matrix multiplications are done in the ring R before the quotient and remainder are computed. The components of each y_m belong to R_s.

Next we define the auxiliary matrices

$$\mathcal{A}_k = \begin{cases} A^{(0)}, & k = 0, \\ \big[\mathcal{A}_{k-1}\ ,\ [y_i, \ldots, y_n]\big], & 1 \le k \le \beta - 1 \end{cases} \tag{34}$$

and compute the reduced row-echelon form of $\dot{\mathcal{A}}_k$ using Gauss–Jordan elimination over the field R/pR. The reduced row-echelon form of $\dot{\mathcal{A}}_k$ can be interpreted as a tableau telling which columns of $\dot{\mathcal{A}}_k$ are linearly independent of their predecessors (the accepted columns), and also giving the linear combination of previously accepted columns that will annihilate a linearly dependent column. On the first iteration (with $k = 0$), step $\star$ in Algorithm 1 will build up the matrix

$$X_0 = \mathrm{null}(\dot{\mathcal{A}}_0), \tag{35}$$

where null$(\cdot)$ is the standard algorithm for computing a basis for the nullspace of a matrix from the reduced row-echelon form (followed by a truncation to replace elements in R/pR with their representatives in R_s). But rather than rotating these columns to the end as in Algorithm 1, we now *append* the corresponding y_i to the end of $\mathcal{A}_{k-1}$ to form $\mathcal{A}_k$ for $k \ge 1$. The "dead" columns left behind (not accepted, not active) serve only as placeholders, causing the resulting matrices $\mathcal{A}_k$ to be nested. We use rref$(\cdot)$ to denote the reduced row-echelon form of a matrix polynomial. The leading columns of rref$(\dot{\mathcal{A}}_k)$ will then coincide with rref$(\dot{\mathcal{A}}_{k-1})$, and the nullspace matrices will also be nested. We denote the new columns of null$(\dot{\mathcal{A}}_k)$ beyond those of null$(\dot{\mathcal{A}}_{k-1})$ by $[Y_k; U_k]$:

$$\begin{pmatrix} X_0 & Y_1 & \cdots & Y_{k-1} & Y_k \\ 0 & [U_1; 0] & \cdots & [U_{k-1}; 0] & U_k \end{pmatrix} := \mathrm{null}(\dot{\mathcal{A}}_k). \tag{36}$$

Note that $\mathcal{A}_k$ is $n \times (n + R_{k-1})$, where

$$R_{-1} = 0, \qquad R_k = r_0 + \cdots + r_k = \dim \ker \dot{\mathcal{A}}_k, \quad (k \ge 0). \tag{37}$$

We also see that X_0 is $n \times r_0$, Y_k is $n \times r_k$, U_k is $r_{k-1} \times r_k$, and

$$r_k \le r_{k-1}, \quad (k \ge 0). \tag{38}$$

This inequality is due to the fact that the dimension of the kernel cannot increase by more than the number of columns added.

If column i of $\dot{\mathcal{A}}_k$ is linearly dependent on its predecessors, the coefficients a_m used in step $\star$ of Algorithm 1 are precisely the (truncations of the) coefficients that appear in column i of rref$(\dot{\mathcal{A}}_k)$. As shown in Fig. 2, the corresponding null vector (i.e. column of $[Y_k; U_k]$) contains the negatives of these coefficients in the rows corresponding to the previously accepted columns of $\dot{\mathcal{A}}_k$, followed by a 1 in row i. Thus, in step $\star$, if $k \ge 1$ and we write $x_m = \Lambda\big(x_m^{(0)}; \ldots; x_m^{(\alpha)}\big)$ with $\alpha = \max(\alpha_m - 1, 0)$, the update

$$x_i^{(new)} = x_i^{(old)} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m x_m, \qquad a_m x_m = \Lambda\big(z^{(0)}; \ldots; z^{(\alpha_m)}\big),$$

$$z^{(j)} = \begin{cases} \mathrm{rem}(a_m x_m^{(0)}, p), & j = 0, \\ \mathrm{rem}(a_m x_m^{(j)}, p) + \mathrm{quo}(a_m x_m^{(j-1)}, p), & 1 \le j < \alpha_m, \\ \mathrm{quo}(a_m x_m^{(j-1)}, p), & j = \alpha_m \text{ and } \alpha_m > 0, \end{cases}$$

is equivalent to

$$\begin{aligned} X_k \ &= \iota^k(X_{-1}) Y_k + \mathrm{rem}\Big(\big[\iota^{k-1}\rho(X_0), \ldots, \iota^0 \rho(X_{k-1})\big] U_k, p\Big) \\ &\quad + \mathrm{quo}\Big(\big[\iota^k(X_0), \ldots, \iota^1(X_{k-1})\big] U_k, p\Big), \end{aligned} \tag{39}$$

where $\iota, \rho : (M_s)^l \to (M_s)^{l+1}$ act column by column, padding them with zeros:

$$\iota(x) = (0; x), \qquad \rho(x) = (x; 0), \quad x \in (M_s)^l,\ 0 \in M_s. \tag{40}$$

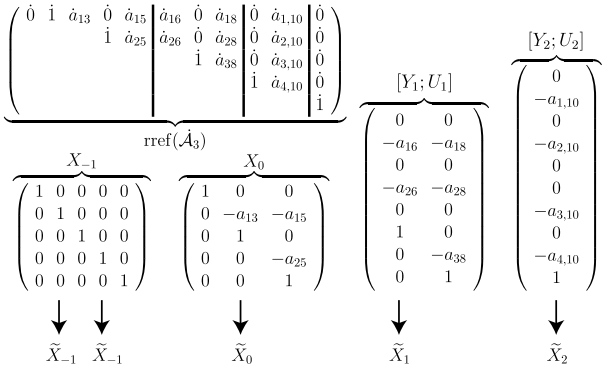

Fig. 2. The reduced row-echelon form of $\dot{\mathcal{A}}_\beta$ contains all the information necessary to construct $V(\lambda) = [\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{s-1})]$. An arrow from a column $[v; u]$ of $[Y_k; U_k]$ indicates that the vector $([\mathrm{rem}(\mathbb{X}_{k-1}u, p); v] + \mathrm{quo}(\iota(\mathbb{X}_{k-1}u), p))$ is a column of $\widetilde{X}_k$.

Here $\Lambda\iota\Lambda^{-1}$ is multiplication by p, which embeds $M_{ls} \cong M/p^lM$ in $M_{(l+1)s} \cong M/p^{l+1}M$ as a module over R, while ρ is an embedding of vector spaces over K (but not an R-module morphism). If we define the matrices $\mathbb{X}_0 = X_0$ and

$$\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k] = \left[\begin{pmatrix} 0_{nk\times r_0} \\ X_0 \end{pmatrix}, \begin{pmatrix} 0_{n(k-1)\times r_1} \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} X_k \end{pmatrix}\right], \quad (k \geq 1), \tag{41}$$

then (39) simply becomes

$$X_k = [\mathrm{rem}(\mathbb{X}_{k-1}U_k, p); Y_k] + \mathrm{quo}(\iota(\mathbb{X}_{k-1}U_k), p). \tag{42}$$

As in (33) above, the matrix multiplications are done in the ring R before the quotient and remainder are computed to obtain X_k. Finally, we line up the columns of X_{k-1} with the last r_{k-1} columns of $\dot{\mathcal{A}}_k$ and extract (i.e. accept) columns of X_{k-1} that correspond to new, linearly independent columns of $\dot{\mathcal{A}}_k$. We denote the matrix of extracted columns by $\widetilde{X}_{k-1}$. At the completion of the algorithm, the unimodular matrix $V(\lambda)$ that puts $A(\lambda)$ in local Smith form is given by

$$V(\lambda) = \left[\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{\beta-1})\right]. \tag{43}$$

The final algorithm is presented in Fig. 3. In the step marked •, we can avoid re-computing the reduced row-echelon form of the first $n+R_{k-2}$ columns of $\dot{\mathcal{A}}_k$ by storing the sequence of Gauss–Jordan transformations (Golub and Van Loan, 1996) that reduced $\dot{\mathcal{A}}_{k-1}$ to row-echelon form. To compute $[Y_k; U_k]$, we need only apply these transformations to the new columns of $\dot{\mathcal{A}}_k$ and then proceed with the row-reduction algorithm on these final columns. Also, if A_0 is large and sparse, rather than reducing to row-echelon form, one could find kernels using an LU factorization designed to handle singular matrices. This would allow the use of graph theory (clique analysis) to choose pivots in the Gaussian elimination procedure to minimize fill-in. We also note that if $\Delta(\lambda)$ contains only one irreducible factor, the local Smith form is a (global) Smith form of $A(\lambda)$.

3.2. From local to global (step 2)

Now that we have a local Smith form (22) for every irreducible factor $p_j(\lambda)$ of $\Delta(\lambda)$, we can use the extended Euclidean algorithm to obtain a family of polynomials $\{g_j(\lambda)\}_{j=1}^l$ with $\deg(g_j(\lambda)) < s_j\kappa_{jn}$, where $s_j = \deg(p_j)$, such that

$$\sum_{j=1}^{l}\left[g_j(\lambda)\prod_{k=1,k\neq j}^{l} p_k(\lambda)^{\kappa_{kn}}\right] = 1, \tag{44}$$

Algorithm 2. (Local Smith form, final version)

$k = 0,\ \ R_{-1} = 0,\ \ \mathcal{A}_0 = A^{(0)}$
$X_0 = \mathbb{X}_0 = \text{null}(\dot{\mathcal{A}}_0)$
$r_0 = R_0 = \text{num\_cols}(\mathbb{X}_0)$ (number of columns)
$\widetilde{X}_{-1} = [e_{j_1}, \ldots, e_{j_{n-r_0}}]$, (columns j_i of $\text{rref}(\dot{\mathcal{A}}_0)$ start new rows)
while $R_k < \mu$ (μ = algebraic multiplicity of p)
  $k = k + 1$
  $\mathcal{A}_k = \left[\mathcal{A}_{k-1}\,,\ \text{rem}(\left[A^{(k)}, \ldots, A^{(1)}\right]X_{k-1}, p) + \text{quo}(\left[A^{(k-1)}, \ldots, A^{(0)}\right]X_{k-1}, p)\ \right]$
- $[Y_k; U_k] =$ new columns of $\text{null}(\dot{\mathcal{A}}_k)$ beyond those of $\text{null}(\dot{\mathcal{A}}_{k-1})$
  $r_k = \text{num\_cols}(U_k)$, ($U_k$ is $R_{k-1} \times r_k$)
  $R_k = R_{k-1} + r_k$
  $X_k = [\text{rem}(\mathbb{X}_{k-1}U_k, p); Y_k] + \text{quo}(\iota(\mathbb{X}_{k-1}U_k), p)$ (X_k is $n(k+1) \times r_k$)
  $\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k]$ ($\mathbb{X}_k$ is $n(k+1) \times R_k$)
  $\widetilde{X}_{k-1} = X_{k-1}(:, [j_1, \ldots, j_{r_{k-1}-r_k}])$, (columns $n + R_{k-2} + j_i$ of $\text{rref}(\dot{\mathcal{A}}_k)$ start new rows)
end while
$\beta = k + 1$ (maximal Jordan chain length)
$\widetilde{X}_{\beta-1} = X_{\beta-1}$
$V(\lambda) = \left[\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{\beta-1})\right]$

Fig. 3. Algorithm for computing a unimodular local Smith form.

where $p_j(\lambda)^{\kappa_{jn}}$ is the last entry in the diagonal matrix of the local Smith form at $p_j(\lambda)$. The integers κ_{jn} are positive. We define a matrix polynomial $B_n(\lambda)$ via

$$B_n(\lambda) = \sum_{j=1}^{l}\left[g_j(\lambda)V_j(\lambda)\prod_{k\neq j}^{l} p_k(\lambda)^{\kappa_{kn}}\right]. \tag{45}$$

The main result of this section is stated as follows.

Proposition 12. *The matrix polynomial $B_n(\lambda)$ in (45) has two key properties:*

(1) *Let $b_{ni}(\lambda)$ be the ith column of $B_n(\lambda)$. Then $A(\lambda)b_{ni}(\lambda)$ is divisible by $d_i(\lambda)$, where $d_i(\lambda) = \prod_{j=1}^{l} p_j(\lambda)^{\kappa_{ji}}$ is the ith diagonal entry in $D(\lambda)$ of the Smith form.*
(2) $\det[B_n(\lambda)]$ *is not divisible by $p_j(\lambda)$ for $j = 1, \ldots, l$.*

Proof. (1) Let $v_{ji}(\lambda)$ be the ith column of $V_j(\lambda)$. Then $A(\lambda)v_{ji}(\lambda)$ is divisible by $p_j(\lambda)^{\kappa_{ji}}$ and

$$b_{ni}(\lambda) = \sum_{j=1}^{l}\left[\prod_{k\neq j}^{l} p_k(\lambda)^{\kappa_{kn}}\right]g_j(\lambda)v_{ji}(\lambda).$$

Since $\kappa_{jn} \geq \kappa_{ji}$ for $1 \leq i \leq n$ and $1 \leq j \leq l$, $A(\lambda)b_{ni}(\lambda)$ is divisible by $d_i(\lambda)$.

(2) The local Smith form construction ensures that $p_j(\lambda) \nmid \det[V_j(\lambda)]$ for each $1 \leq j \leq l$. Eq. (44) modulo $p_j(\lambda)$ shows that $p_j(\lambda) \nmid g_j(\lambda)$. By definition,

$$\begin{aligned}\det[B_n(\lambda)] &= \det\left(\left[b_{n1}(\lambda)\,,\ \ldots\ ,\ b_{nn}(\lambda)\right]\right) = \det\left(\left[b_{ni}(\lambda)\right]_{i=1}^{n}\right)\\ &= \det\left(\left[\sum_{j'=1}^{l}\left(\prod_{k\neq j'}^{l} p_k(\lambda)^{\kappa_{kn}}\right)g_{j'}(\lambda)v_{j'i}(\lambda)\right]_{i=1}^{n}\right).\end{aligned}$$

Each term in the sum is divisible by $p_j(\lambda)$ except $j' = j$. Thus, by multi-linearity,

$$\text{rem}(\det[B_n(\lambda)], p_j(\lambda)) = \text{rem}\left(\left[\prod_{k\neq j}^{l} p_k(\lambda)^{\kappa_{kn}}\right]^n \left[g_j(\lambda)\right]^n \det\left[V_j(\lambda)\right],\ p_j(\lambda)\right) \neq 0,$$

as claimed. □

Remark 13. It is possible for $\det[B_n(\lambda)]$ to be non-constant; however, its irreducible factors will be distinct from $p_1(\lambda), \ldots, p_l(\lambda)$.

Remark 14. Rather than building $B_n(\lambda)$ as a linear combination (45), we may form $B_n(\lambda)$ with columns

$$b_{ni}(\lambda) = \sum_{j=1}^{l} \left[\prod_{k \neq j}^{l} p_k(\lambda)^{\max(\kappa_{ki}, 1)} \right] g_{ij}(\lambda) v_{ji}(\lambda), \quad (1 \leq i \leq n),$$

where $\{g_{ij}\}_{j=1}^{l}$ solves the extended GCD problem

$$\sum_{j=1}^{l} \left[g_{ij}(\lambda) \prod_{k \neq j}^{l} p_k(\lambda)^{\max(\kappa_{ki}, 1)} \right] = 1.$$

The two properties proved above also hold for this definition of $B_n(\lambda)$. This modification can significantly reduce the size of the coefficients in the computation when there is a wide range of Jordan chain lengths. But if κ_{ji} only changes slightly for $1 \leq i \leq n$, this change will not significantly affect the total running time of the algorithm.

3.3. Construction of unimodular multipliers (Step 3)

Given $[f_1(\lambda); \ldots; f_n(\lambda)] \in K[\lambda]^n$, we can compute the Hermite form (20) to obtain a unimodular matrix $Q(\lambda)$ such that, after reversing rows, $Q(\lambda)f(\lambda) = [0; \ldots; 0; r(\lambda)]$, where $r = \gcd(f_1, \ldots, f_n)$. We apply this procedure to the last column of $B_n(\lambda)$ and define $V_n(\lambda) = Q(\lambda)^{-1}$. The resulting matrix

$$B_{n-1}(\lambda) := V_n(\lambda)^{-1} B_n(\lambda)$$

is zero above the main diagonal in column n. We then apply this procedure to the first $n-1$ components of column $n-1$ of $B_{n-1}(\lambda)$ to get a new $Q(\lambda)$, and define

$$V_{n-1}(\lambda) = \left(\begin{array}{ccc:c} & & & 0 \\ & Q(\lambda)^{-1} & & \vdots \\ & & & 0 \\ \hdashline 0 & \cdots & 0 & 1 \end{array} \right). \tag{46}$$

It follows that $B_{n-2}(\lambda) := V_{n-1}(\lambda)^{-1} B_{n-1}(\lambda)$ is zero above the main diagonal in columns $n-1$ and n. Continuing in this fashion, we obtain unimodular matrices $V_n(\lambda), \ldots, V_2(\lambda)$ such that

$$A(\lambda)B_n(\lambda) = A(\lambda) \underbrace{V_n(\lambda) \cdots V_2(\lambda)}_{V(\lambda)} V_2(\lambda)^{-1} \cdots \underbrace{V_n(\lambda)^{-1} B_n(\lambda)}_{B_{n-1}(\lambda)} = A(\lambda)V(\lambda)B_1(\lambda),$$

where $V(\lambda)$ is unimodular, $B_1(\lambda)$ is lower triangular, and

$$\det[B_1(\lambda)] = \text{const} \cdot \det[B_n(\lambda)]. \tag{47}$$

The matrix $V(\lambda)$ puts $A(\lambda)$ in Smith form:

Proposition 15. *There is a unimodular matrix polynomial $E(\lambda)$ such that*

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \tag{48}$$

where $D(\lambda)$ is of the form (3).

Proof. Let $r_{mi}(\lambda)$ denote the entry of $B_1(\lambda)$ in the mth row and ith column. Define $y_i(\lambda)$ and $z_i(\lambda)$ to be the ith columns of $A(\lambda)V(\lambda)$ and $A(\lambda)V(\lambda)B_1(\lambda)$, respectively, so that

$$z_i(\lambda) = y_i(\lambda)r_{ii}(\lambda) + \sum_{m=i+1}^{n} y_m(\lambda)r_{mi}(\lambda), \quad (1 \le i \le n). \tag{49}$$

By Proposition 12, $z_i(\lambda)$ is divisible by $d_i(\lambda)$ for $1 \le i \le n$ and $p_j(\lambda) \nmid \det[B_1(\lambda)]$ for $1 \le j \le l$. It follows that the diagonal entries $r_{ii}(\lambda)$ of $B_1(\lambda)$ are relatively prime to each of the $d_i(\lambda)$. As $d_n(\lambda)$ divides $y_n(\lambda)r_{nn}(\lambda)$ and is relatively prime to $r_{nn}(\lambda)$, it divides $y_n(\lambda)$ alone. Now suppose $1 \le i < n$ and we have shown that $d_m(\lambda)$ divides $y_m(\lambda)$ for $i < m \le n$. Then since $d_i(\lambda)$ divides $d_m(\lambda)$ for $m > i$ and $r_{ii}(\lambda)$ is relatively prime to $d_i(\lambda)$, we conclude from (49) that $d_i(\lambda)$ divides $y_i(\lambda)$. By induction, $d_i(\lambda)$ divides $y_i(\lambda)$ for $1 \le i \le n$. Thus, there is a matrix polynomial $E(\lambda)$ such that (48) holds. Because $V(\lambda)$ is unimodular and $\det[A(\lambda)] = \text{const} \cdot \det[D(\lambda)]$, it follows that $E(\lambda)$ is also unimodular, as claimed. □

Remark 16. $V(\lambda)$ constructed as described above puts $A(\lambda)$ in a global Smith form whether we build $B_n(\lambda)$ as a linear combination (45) or as in Remark 14.

Remark 17. We can stop before reaching $V_2(\lambda)$ by adding a test

while $d_k \neq 1$

to the loop in which $V(\lambda)$ is constructed. When the loop terminates, we have $V(\lambda) = V_n(\lambda)\cdots V_{k+1}(\lambda)$, where k is the largest integer for which

$$d_1(\lambda) = \cdots = d_k(\lambda) = 1.$$

Note that k is known from the local Smith form calculations. The last $n-k$ columns of $V_n(\lambda)\cdots V_{k+1}(\lambda)$ are the same as those of $V_n(\lambda)\cdots V_2(\lambda)$; therefore, either can be used for $V(\lambda)$ as they contain identical Jordan chains.

Remark 18. A slight modification of this procedure can significantly reduce the degree of the polynomials and the size of the coefficients in the computation. In this variant, rather than applying the extended GCD algorithm on $b_{nn}(\lambda)$ to find a unimodular matrix polynomial $Q(\lambda)$ so that $Q(\lambda)b_{nn}(\lambda)$ has the form $[0;\ldots;0;r(\lambda)]$, we compute $Q(\lambda)$ that puts $\text{rem}(b_{nn}(\lambda), d_n(\lambda))$ into this form. That is, we replace the last column of $B_n(\lambda)$ with $\text{rem}(b_{nn}(\lambda), d_n(\lambda))$ before computing $Q(\lambda)$. To distinguish, we denote this new definition of $V_n(\lambda) = Q(\lambda)^{-1}$ by $\widetilde{V}_n(\lambda)$ and the resulting $B_{n-1}(\lambda)$ by $\widetilde{B}_{n-1}(\lambda)$. Continuing in this manner, we find unimodular matrix polynomials $\widetilde{V}_n(\lambda), \ldots, \widetilde{V}_{k+1}(\lambda)$ by applying the procedure on $\text{rem}(\tilde{b}_{ii}(\lambda), d_i(\lambda))$ for $i = n, \ldots, k+1$, where $\tilde{b}_{ii}(\lambda)$ contains the first i components of column i of $\widetilde{B}_i(\lambda)$ and k is defined as in Remark 17. We also define

$$\bar{B}_i = \widetilde{V}_{i+1}(\lambda)^{-1}\cdots\widetilde{V}_n(\lambda)^{-1}B_n(\lambda), \quad (k \le i \le n-1).$$

Note that in general, $\bar{B}_i(\lambda) \neq \widetilde{B}_i(\lambda)$. It remains to show that this definition of $\widetilde{V}(\lambda) = \widetilde{V}_n(\lambda)\ldots\widetilde{V}_{k+1}(\lambda)$, which satisfies

$$A(\lambda)B_n(\lambda) = A(\lambda)\underbrace{\widetilde{V}_n(\lambda)\cdots\widetilde{V}_{k+1}(\lambda)}_{\widetilde{V}(\lambda)}\widetilde{V}_{k+1}(\lambda)^{-1}\cdots\underbrace{\widetilde{V}_n(\lambda)^{-1}B_n(\lambda)}_{\bar{B}_{n-1}(\lambda)} = A(\lambda)\widetilde{V}(\lambda)\bar{B}_k(\lambda),$$

also puts $A(\lambda)$ in Smith form:

Proposition 19. *There is a unimodular matrix polynomial $\widetilde{E}(\lambda)$ such that*

$$A(\lambda)\widetilde{V}(\lambda) = \widetilde{E}(\lambda)D(\lambda), \tag{50}$$

where $D(\lambda)$ is of the form (3).

Proof. Define $\tilde{q}_i(\lambda) = \left[\mathrm{quo}(\tilde{b}_{ii}(\lambda), d_i(\lambda)); 0\right] \in M = R^n$ for $i = n, \ldots, k+1$, where $0 \in R^{n-i}$, $\tilde{b}_{ii}(\lambda)$ was defined above, and $\widetilde{B}_n(\lambda) := B_n(\lambda)$. Then we have

$$\begin{aligned}\widetilde{B}_{n-1}(\lambda) &= \widetilde{V}_n(\lambda)^{-1}\left(B_n(\lambda) - \left[0_{n\times(n-1)} \mid d_n(\lambda)\tilde{q}_n(\lambda)\right]\right)\\ &= \bar{B}_{n-1}(\lambda) - \left[0_{n\times(n-1)} \mid d_n(\lambda)\widetilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\right].\end{aligned}$$

The first $n-1$ columns of $\widetilde{B}_n(\lambda)$ are the same as those of $\bar{B}_n(\lambda)$. Continuing, we have

$$\begin{aligned}\widetilde{B}_{n-2}(\lambda) &= \widetilde{V}_{n-1}(\lambda)^{-1}\left(\widetilde{B}_{n-1}(\lambda) - \left[0_{n\times(n-2)} \mid d_{n-1}(\lambda)\tilde{q}_{n-1}(\lambda) \mid 0_{n\times 1}\right]\right)\\ &= \widetilde{V}_{n-1}(\lambda)^{-1}\left(\bar{B}_{n-1}(\lambda) - \left[0_{n\times(n-2)} \mid d_{n-1}(\lambda)\tilde{q}_{n-1}(\lambda) \mid d_n(\lambda)\widetilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\right]\right)\\ &= \bar{B}_{n-2}(\lambda) - \left[0_{n\times(n-2)} \mid d_{n-1}(\lambda)\widetilde{V}_{n-1}(\lambda)^{-1}\tilde{q}_{n-1}(\lambda) \mid d_n(\lambda)\widetilde{V}_{n-1}(\lambda)^{-1}\widetilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\right].\end{aligned}$$

It follows by induction that

$$\begin{aligned}\widetilde{B}_k(\lambda) &= \widetilde{V}_{k+1}(\lambda)^{-1}\left(\widetilde{B}_{k+1}(\lambda) - \left[0_{n\times k} \mid d_{k+1}(\lambda)\tilde{q}_{k+1}(\lambda) \mid 0_{n\times(n-k-1)}\right]\right) \qquad (51)\\ &= \bar{B}_k(\lambda) - \left[0_{n\times k} \mid d_{k+1}(\lambda)\widetilde{V}_{k+1}(\lambda)^{-1}\tilde{q}_{k+1}(\lambda) \mid \cdots \mid d_n(\lambda)\widetilde{V}_{k+1}(\lambda)^{-1}\cdots\widetilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\right].\end{aligned}$$

$\widetilde{B}_k(\lambda)$ is zero above the main diagonal in columns $k+1$ to n. Define

$$u_i(\lambda) := \widetilde{V}_{k+1}(\lambda)^{-1}\cdots\widetilde{V}_i(\lambda)^{-1}\tilde{q}_i(\lambda), \quad (k+1 \le i \le n).$$

Then the ith column of the difference $\bar{B}_k(\lambda) - \widetilde{B}_k(\lambda)$ is $d_i(\lambda)u_i(\lambda)$ for $k+1 \le i \le n$.

Let $\tilde{r}_{mi}(\lambda)$ denote the entry of $\widetilde{B}_k(\lambda)$ in the mth row and ith column. Define $\tilde{y}_i(\lambda)$ and $z_i(\lambda)$ to be the ith columns of $A(\lambda)\widetilde{V}(\lambda)$ and $A(\lambda)\widetilde{V}(\lambda)\bar{B}_k(\lambda)$, respectively, so that

$$z_i(\lambda) = \left[\tilde{y}_i(\lambda)\tilde{r}_{ii}(\lambda) + \sum_{m=i+1}^{n}\tilde{y}_m(\lambda)\tilde{r}_{mi}(\lambda)\right] + d_i(\lambda)A(\lambda)\widetilde{V}(\lambda)u_i(\lambda), \quad (k+1 \le i \le n).$$

By Proposition 12, $z_i(\lambda)$ is divisible by $d_i(\lambda)$ and $p_j(\lambda) \nmid \det[B_n(\lambda)] = \mathrm{const}\cdot\det[\bar{B}_{i-1}(\lambda)]$ for $k+1 \le i \le n$ and $1 \le j \le l$. As d_i divides d_m for $i \le m \le n$ and since (51) holds with k replaced by $i-1$ for $k+1 \le i \le n$, $\det[\bar{B}_{i-1}(\lambda)] - \det[\widetilde{B}_{i-1}(\lambda)]$ is divisible by $d_i(\lambda)$ due to the multi-linearity of determinants. We also know that $\det[\widetilde{B}_{i-1}(\lambda)]$ is divisible by $\tilde{r}_{ii}(\lambda)$. Proof by contradiction shows that $\tilde{r}_{ii}(\lambda)$ is relatively prime to $d_i(\lambda)$ for $k+1 \le i \le n$. Then we argue by induction as in the proof of Proposition 15 to conclude that $d_i(\lambda)$ divides $\tilde{y}_i(\lambda)$ for $k+1 \le i \le n$. It holds trivially for $1 \le i \le k$ as $d_1 = \cdots = d_k = 1$. Thus, there is a matrix polynomial $\widetilde{E}(\lambda)$ such that (50) holds. Because $\widetilde{V}(\lambda)$ is unimodular and $\det[A(\lambda)] = \mathrm{const}\cdot\det[D(\lambda)]$, it follows that $E(\lambda)$ is also unimodular. □

4. Performance comparison

In this section, we compare our algorithm to Villard's method with good conditioning (Villard, 1995), which is another deterministic sequential method for computing Smith forms with multipliers, and to 'MatrixPolynomialAlgebra[SmithForm]' in Maple. All three algorithms are implemented in exact arithmetic using Maple 13. The maximum number of digits that Maple can use for the numerator and denominator of a rational number (given by 'kernelopts(maxdigits)') is over 38 billion. However, limitations of available memory and running time set the limit on the largest integer number much lower than this. We use the variant of Algorithm 1 given in the Appendix to compute local Smith forms.

To evaluate the performance of these methods, we generate several groups of diagonal matrices $D(\lambda)$ over $\mathbb{Q}$ and multiply them on each side by unimodular matrices of the form $L(\lambda)Z(\lambda)$, where $L(\lambda)$ is unit lower triangular and $Z(\lambda)$ is unit upper triangular, both with off diagonal entries of the form $\lambda - i$ with $i \in \{-10, \ldots, 10\}$ a random integer. As a final step, we apply a row or column permutation to the resulting matrix. We find that row permutation has little effect on the running time of the algorithms while column permutation reduces the performance of Villard's method. We compare the results in two extreme cases: (1) without column permutation and (2) with columns reversed. Each process is repeated five times for each $D(\lambda)$ and the median running time is recorded.

We use several parameters in the comparison, including the size n of the square matrix $A(\lambda)$, the bound d of the polynomial degrees of the entries in $A(\lambda)$, the number l of irreducible factors in $\det[A(\lambda)]$, and the maximal Jordan chain length κ_{jn}.

In Fig. 4, we show the running time of three tests with linear irreducible factors of the form $p_j = \lambda - \lambda_j$. As Villard's method and Maple compute the left and right multipliers $U(\lambda)$ and $V(\lambda)$ while our algorithm instead computes $E(\lambda)$ and $V(\lambda)$, we also report the cost of inverting $E(\lambda)$ to obtain $U(\lambda)$ at the end of our algorithm (using Maple's matrix inverse routine). This step could be made significantly faster by taking advantage of the fact that $E(\lambda)$ is unimodular. For example, one could store the sequence of elementary unimodular operations such that $T(\lambda) = Q_m(\lambda)\cdots Q_1(\lambda)E(\lambda)$ is unit upper triangular. It would not be necessary to actually form the matrices $T(\lambda)^{-1}$ or

$$U(\lambda) = T(\lambda)^{-1}Q_m(\lambda)\cdots Q_1(\lambda) \tag{52}$$

as the right-hand side can be applied directly to any vector polynomial using back substitution to solve $T(\lambda)x(\lambda) = z(\lambda)$ in the last step. The same idea is standard in numerical linear algebra, where the LU-decomposition of a matrix is less expensive to compute than its inverse, and is equally useful. In the first test of Fig. 4, $D_n(\lambda)$ is of the form

$$D_n(\lambda) = \mathrm{diag}[1, \ldots, 1, \lambda, \lambda(\lambda-1), \lambda^2(\lambda-1), \lambda^2(\lambda-1)^2],$$

where the matrix size n increases, starting with $n = 4$. Hence, we have $d = 8$, $l = 2$, and $\kappa_{1n} = \kappa_{2n} = 2$ all fixed. (The unimodular matrices in the construction of $A(\lambda)$ each have degree 2.) For this test, inverting $E(\lambda)$ to obtain $U(\lambda)$ is the most expensive step of our algorithm. Without column permutation of the test matrices, our algorithm (with $U(\lambda)$) and Villard's method have similar running times, both outperforming Maple's built-in function. With column permutation, the performance of Villard's method drops to the level of Maple's routine while our algorithm remains faster. For the second test, we use test matrices $D_l(\lambda)$ of size 9×9, where l is the number of roots of $\det[A(\lambda)]$:

$$D_l(\lambda) = \mathrm{diag}\left[1, \ldots, 1, \prod_{j=1}^{l}(\lambda - j)\right], \qquad (l = 1, 2, \ldots).$$

Thus, $n = 9$, $d = l + 4$ and $\kappa_{jn} = 1$ for $1 \le j \le l$. This time the relative cost of inverting $E(\lambda)$ to obtain $U(\lambda)$ decreases with l in our algorithm, which is significantly faster than the other two methods whether or not we permute columns in the test matrices. In the third test, we use 9×9 test matrices $D_k(\lambda)$ of the form

$$D_k(\lambda) = \mathrm{diag}[1, \ldots, 1, (\lambda-1)^k], \quad (k = 1, 2, \ldots),$$

with $n = 9$, $l = 1$, $\kappa_{1n} = k$ and $d = k + 4$. We did not implement the re-use strategy for computing the reduced row-echelon form of $\mathcal{A}_k$ by storing the Gauss–Jordan transformations used to obtain $\mathrm{rref}(\mathcal{A}_{k-1})$, and then continuing with only the new columns of $\mathcal{A}_k$. This is because the built-in function LinearAlgebra[ReducedRowEchelonForm] is much faster than can be achieved by a user defined Maple code for the same purpose. In a lower level language (or with access to Maple's internal code), this re-use strategy would decrease the running time of local Smith form calculations in this test from $O(k^4)$ to $O(k^3)$. A similar decrease in the cost of computing the left-multiplier $U(\lambda) = E(\lambda)^{-1}$ could be achieved by computing $T(\lambda)$ in (52) instead.

We also evaluate the performance on three test problems (numbered 4–6) with irreducible polynomials of higher degree. The results are given in Fig. 5. In the fourth test, we use matrices $D_n(\lambda)$ similar to those in the first test, but with irreducible polynomials of degree 2 and 4. Specifically, we define

$$D_n(\lambda) = \mathrm{diag}[1, \ldots, 1, p_1, p_1p_2, p_1^2p_2, p_1^2p_2^2], \quad (n = 4, 5, \ldots),$$

where $p_1 = \lambda^2+\lambda+1$, $p_2 = \lambda^4+\lambda^3+\lambda^2+1$, $\kappa_{1n} = 2$, $\kappa_{2n} = 2$, and $d = 16$. When the columns of the test matrices are permuted, our algorithm is faster than the other two methods whether or not $U(\lambda)$ is computed. When the columns are not permuted, computing $U(\lambda)$ causes our method to be slower than Villard's method. In this test, our algorithm would benefit from switching to the R/pR version of Algorithm 2 rather than the version over K described in the Appendix. It would also benefit from computing $T(\lambda)$ in (52) rather than the full inverse $U(\lambda) = E(\lambda)^{-1}$. In the fifth test, we use 9×9 test matrices $D_k(\lambda)$ of the form

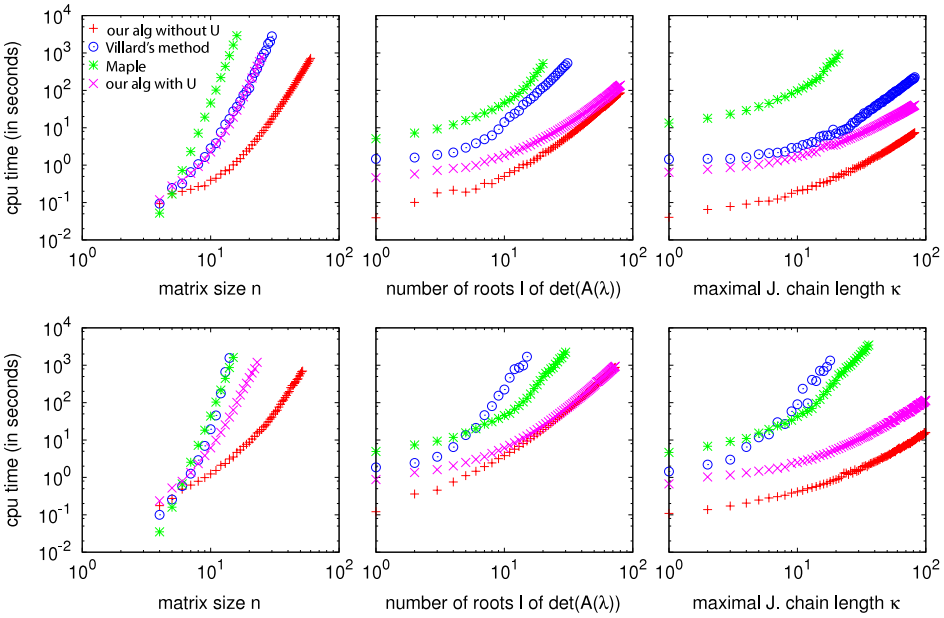

Fig. 4. Comparison of running time of our algorithm (with or without computing $U(\lambda)$) to Villard's method, and to Maple's Smith form routine, on three families of test matrices. (Top row) without column permutation of test matrices. (Bottom row) with column permutation.

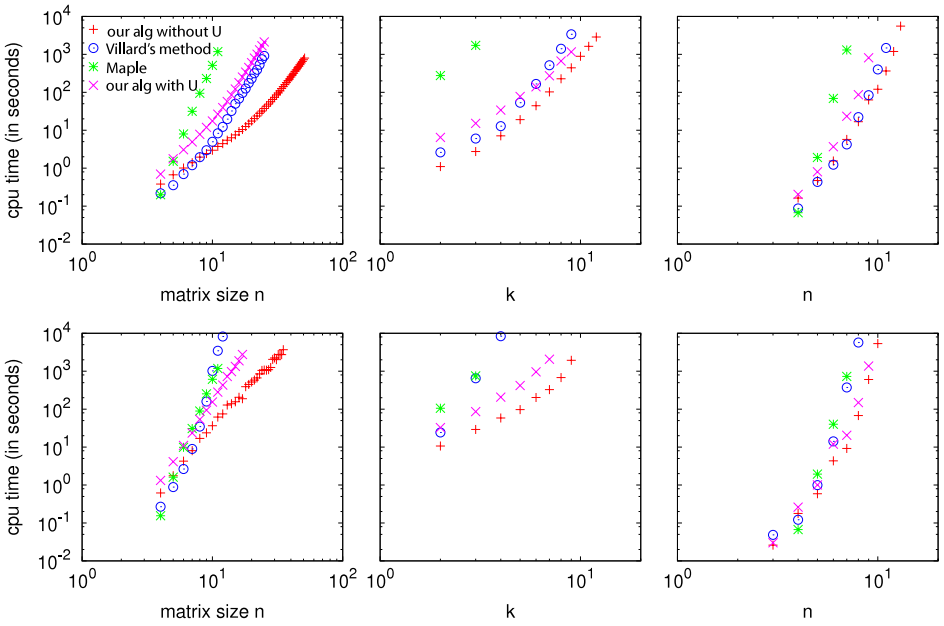

Fig. 5. Comparison of running times of the algorithms for three test problems in which the irreducible factors $p_j(\lambda)$ of the determinant are of degree greater than 1. (Top row) without column permutation of test matrices. (Bottom row) with column permutation.

$$D_k(\lambda) = \operatorname{diag}\left[1, \ldots, 1, \prod_{j=1}^{k}(\lambda^2 + j), \prod_{j=1}^{k}(\lambda^2 + j)^2, \prod_{j=1}^{k}(\lambda^2 + j)^k\right], \quad (k = 2, 3, \ldots),$$

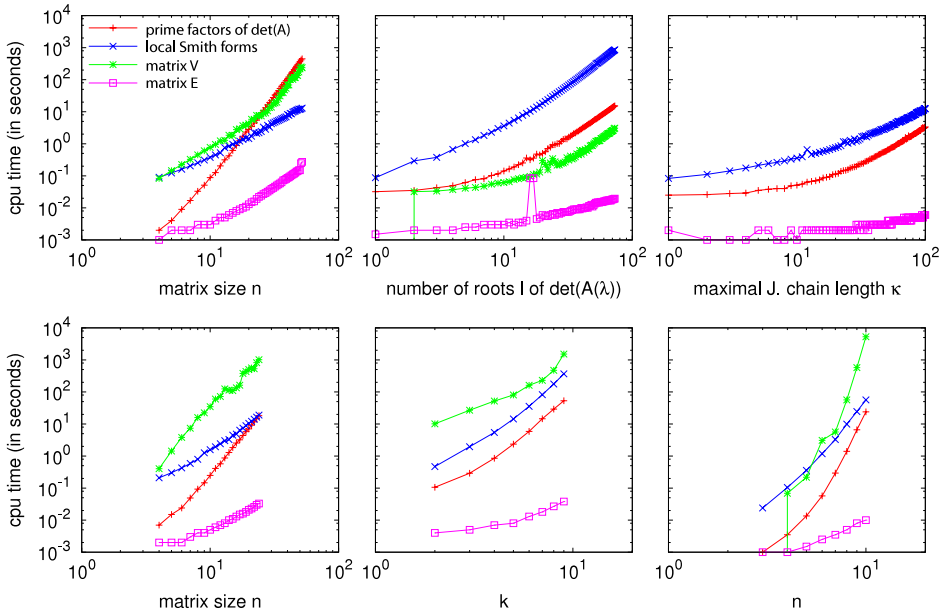

Fig. 6. Running time of each step of our algorithm for the six test problems of Section 4.

with $n = 9, l = k, \kappa_{jn} = k$ and $d = 2k^2+4$. Both the number of factors and maximal Jordan chain length increase with k. Our algorithm performs much better than the others when column permutations are performed on the test matrices. In the final test, we define $n \times n$ matrices

$$D_n(\lambda) = \operatorname{diag}\left[1, 1, (\lambda^2+1), (\lambda^2+1)^2(\lambda^2+2), \ldots, \prod_{j=1}^{n-2}(\lambda^2+j)^{n-1-j}\right], \quad (n = 3, 4, \ldots)$$

so that all the parameters $n, l = n-2, \kappa_{jn} = n-1-j$ and $d = (n-1)(n-2)+4$ increase simultaneously. All three algorithms run very slowly on this last family of test problems.

5. Discussion

The key idea of our algorithm is that it is much less expensive to compute local Smith forms through a sequence of nullspace calculations than it is to compute global Smith forms through a sequence of unimodular row and column operations. This is because (1) row reduction over R/pR in Algorithm 2 (or over K in the Appendix) is less expensive than computing Bézout coefficients over R; (2) the size of the rational numbers that occur in the algorithm remain smaller (as we only deal with the leading terms of A in an expansion in powers of p rather than with all of A); and (3) each column of $V(\lambda)$ in a local Smith form only has to be processed once for each power of p in the corresponding diagonal entry of $D(\lambda)$. Once the local Smith forms are known, we combine them to form a (global) multiplier $V(\lambda)$ for $A(\lambda)$. This last step does involve triangularization of $B_n(\lambda)$ via the extended GCD algorithm, but this is less time consuming in most cases than performing unimodular row and column operations on $A(\lambda)$ to obtain $D(\lambda)$. This is because we only have to apply row operations to $B_n(\lambda)$ (as the columns are already correctly ordered); we keep the degree of polynomials (and therefore the number of terms) in the algorithm small with the operation $\operatorname{rem}(\cdot, d_i)$; and the leading columns of $B_n(\lambda)$ tend to be sparse (as they consist of a superposition of local Smith forms, whose initial columns X_{-1} are a subset of the columns of the identity matrix). Sparsity is not used explicitly in our code, but it does reduce the work required to compute the Bézout coefficients of a column.

A detailed breakdown of the running time of each step of our algorithm is given in Fig. 6. For each test in Section 4, we show only the case where columns of the test matrices are permuted; the

other case is similar. The step labeled "prime factors of $\det(A)$" shows the time of computing the determinant and factoring it into prime factors. The step labeled "local Smith forms" could be made faster in tests 4–6 by working over R/pR (using Algorithm 2 rather than the variant in the Appendix) as the irreducible factors $p_j(\lambda)$ have degree $s_j > 1$ in these tests. Also, although it is not implemented in this paper, this local Smith form construction would be easy to parallelize. The step labeled "matrix V" reports the time of computing $V(\lambda)$ from $B_n(\lambda)$. The cost of this step is zero when there is only one irreducible factor in $\det[A(\lambda)]$ as $B_n(\lambda)$ is already unimodular in that case. This happens when $l = 1$ in the second test, in all cases in the third test, and when $n = 3$ in the last test. Finally, the step labeled "matrix E" reports the time of computing $E(\lambda) = A(\lambda)V(\lambda)D(\lambda)^{-1}$.

The obvious drawback of our algorithm is that we have to compute a local Smith form for each irreducible factor of $\Delta(\lambda)$ separately, while much of the work in deciding whether to accept a column in Algorithm 1 can be done for all the irreducible factors simultaneously by using extended GCDs. In our numerical experiments, it appears that in most cases, the benefit of computing local Smith forms outweighs the fact that there are several of them to compute.

Appendix. Alternative version of Algorithm 2

In this section we present an algebraic framework for local Smith forms of matrix polynomials that shows the connection between Algorithm 2 and the construction of canonical systems of Jordan chains presented in Wilkening (2007). This leads to a variant of the algorithm in which row-reduction is done in the field K rather than in R/pR.

Suppose R is a principal ideal domain and p is a prime in R. M defined via $M = R^n$ is a free R-module with a free basis $\{(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)\}$. Suppose $A : M \to M$ is a R-module morphism. We define submodules

$$N_k = \left\{x \in M : Ax \text{ is divisible by } p^k\right\}, \quad (k \geq 0). \tag{A.1}$$

Then N_k is a free submodule of M by the structure theorem (Hungerford, 1996) for finitely generated modules over a principal ideal domain. (The structure theorem states that if M is a free module over a principal ideal domain R, then every submodule of M is free.) The rank of N_k is also n, as $p^k M \subset N_k \subset M$. Note that $N_0 = M$ and

$$N_{k+1} \subset N_k, \quad (k \geq 0), \tag{A.2}$$

$$N_{k+1} \cap pM = pN_k, \quad (k \geq 0). \tag{A.3}$$

Next we define the spaces W_k via

$$W_k = N_{k+1}/pN_k, \quad (k \geq -1), \tag{A.4}$$

where $N_{-1} := M$ so that $W_{-1} = M/pM$. By (A.3), the action of R/pR on W_k is well-defined, i.e. W_k is a vector space over this field. Let us denote the canonical projection $M \to M/pM$ by π. Note that $\pi(pN_k) = 0$, so π is well-defined from W_k to M/pM for $k \geq -1$. It is also injective as $xp \in N_{k+1} \Rightarrow x \in N_k$, by (A.3). Thus, cosets $\{\dot{x}_1, \ldots, \dot{x}_m\}$ are linearly independent in W_k iff $\{\pi(x_1), \ldots, \pi(x_m)\}$ are linearly independent in M/pM. We define the integers

$$r_k = \text{dimension of } W_k \text{ over } R/pR, \quad (k \geq -1) \tag{A.5}$$

and note that $r_{-1} = n$ and $r_k > 0$ iff there exists $x \in M$ such that $p \nmid x$ and $p^{k+1} \mid Ax$. We also observe that the truncation operator

$$id : W_{k+1} \to W_k : (x + pN_{k+1}) \mapsto (x + pN_k), \quad (k \geq -1) \tag{A.6}$$

is well-defined ($pN_{k+1} \subset pN_k$) and injective ($x \in N_{k+2}$ and $x \in pN_k \Rightarrow x \in pN_{k+1}$, due to (A.3)). We may therefore consider W_{k+1} to be a subspace of W_k for $k \geq -1$, and have the inequalities

$$r_{k+1} \leq r_k, \quad (k \geq -1). \tag{A.7}$$

The case $r_0 = 0$ is not interesting (as $N_k = p^k M$ for $k \geq 0$), so we assume that $r_0 > 0$. Lemma 8 shows that when $R = K[\lambda]$, which we assume from now on, $r_0 > 0$ is equivalent to the condition that $\det[A(\lambda)]$ is divisible by $p(\lambda)$. We also assume that r_k eventually decreases to zero, say

$$r_k = 0 \quad \Leftrightarrow \quad k \geq \beta, \quad \beta := \text{maximal Jordan chain length}. \tag{A.8}$$

This follows from the assumption that $\det[A(\lambda)]$ is not identically zero. It will be useful to define the index sets $I_k = \{i \ : \ n - r_{k-1} + 1 \leq i \leq n - r_k\}$ for $k = 0, \ldots, \beta$.

Any matrix $V = [x_1, \ldots, x_n]$ will yield a local Smith form $AV = ED$ provided that $x_i \in N_k$ for $i \in I_k$ $(0 \leq k \leq \beta)$ and the vectors

$$\{x_i + pN_{k-1}\}_{i \in I_k} \tag{A.9}$$

form a basis for any complement $\widetilde{W}_{k-1}$ of W_k in W_{k-1}. To see that $p \nmid \det E$, we use induction on k to show that the vectors

$$\{\mathrm{quo}(Ax_i, p^{\alpha_i}) + pM\}_{i=1}^{n-r_k} \tag{A.10}$$

are linearly independent in M/pM, where

$$\alpha_i = k, \qquad (i \in I_k). \tag{A.11}$$

Otherwise, a linear combination of the form $\star$ in Algorithm 1 would exist that belongs to $\widetilde{W}_{k-1} \cap W_k$, a contradiction. The result that $p \nmid \det E$ follows from Lemma 8. The *while* loop in Algorithm 1 is a systematic procedure for computing such a collection $\{x_i\}_{i \in I_k}$, and has the added benefit of yielding a unimodular multiplier V.

We now wish to find a convenient representation for these spaces suitable for computation. Since $p^{k+1}M \subset pN_k$, we have the R-module isomorphism

$$N_{k+1}/pN_k \cong (N_{k+1}/p^{k+1}M)/(pN_k/p^{k+1}M), \tag{A.12}$$

i.e.

$$W_k \cong \mathbb{W}_k/p\mathbb{W}_{k-1}, \quad (k \geq 0), \qquad \mathbb{W}_k := N_{k+1}/p^{k+1}M, \quad (k \geq -1). \tag{A.13}$$

Although the quotient $\mathbb{W}_k/p\mathbb{W}_{k-1}$ is a vector space over R/pR, the spaces $\mathbb{W}_k$ and $M/p^{k+1}M$ are not. They are, however, modules over $R/p^{k+1}R$ and vector spaces over K. Note that $A(\lambda)$ induces a linear operator $\mathbb{A}_k$ on $M/p^{k+1}M$ with kernel

$$\mathbb{W}_k = \ker \mathbb{A}_k, \quad (k \geq -1). \tag{A.14}$$

We also define

$$R_k = \frac{\text{dimension of } \mathbb{W}_k \text{ over } K}{s}, \ (k \geq -1, s = \deg p) \tag{A.15}$$

so that $R_{-1} = 0$ and

$$R_k = r_0 + \cdots + r_k, \quad (k \geq 0), \tag{A.16}$$

where we used $\mathbb{W}_0 = W_0$ together with (A.13) and the fact that as a vector space over K, $\dim W_k = sr_k$. By (A.11), $r_{k-1} - r_k = \#\{i \ : \ \alpha_i = k\}$, so

$$\begin{aligned} R_{\beta-1} &= r_0 + \cdots + r_{\beta-1} = (r_{-1} - r_0)0 + (r_0 - r_1)1 + \cdots + (r_{\beta-1} - r_\beta)\beta \\ &= \alpha_1 + \cdots + \alpha_n = \mu = \text{algebraic multiplicity of } p, \end{aligned} \tag{A.17}$$

where we used Theorem 7 in the last step. We also note that $\nu := R_0 = s^{-1} \dim \ker(\mathbb{A}_0)$ can be interpreted as the geometric multiplicity of p.

Equations (A.13) and (A.14) reduce the problem of computing Jordan chains to that of finding kernels of the linear operators $\mathbb{A}_k$ over K. If we represent elements $x \in M/p^{k+1}M$ as lists of coefficients $x^{(j,l,m)} \in K$ such that the components of x involve the terms

$$x^{(j,l,m)} p^j \lambda^m, \qquad 0 \leq j \leq k, \quad 1 \leq l \leq n, \quad 0 \leq m \leq s - 1, \tag{A.18}$$

then multiplication by λ in $M/p^{k+1}M$ becomes the following linear operator on $K^{sn(k+1)}$:

$$\mathbb{S}_k = \begin{pmatrix} I\otimes S & 0 & 0 & 0 \\ I\otimes Z & I\otimes S & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & I\otimes Z & I\otimes S \end{pmatrix}, \qquad S \text{ as in (10)}, \qquad Z = \begin{pmatrix} 0 & 0 & 1 \\ & \ddots & 0 \\ 0 & & 0 \end{pmatrix}. \tag{A.19}$$

Here $\mathbb{S}_k$ is a $(k+1)\times(k+1)$ block matrix, $I\otimes S$ is a Kronecker product of matrices, S and Z are $s\times s$ matrices, and I is $n\times n$. Multiplication by λ^m is represented by $\mathbb{S}_k^m$, which has a similar block-Toeplitz structure to $\mathbb{S}_k$ for $2\le m\le s-1$, but with S replaced by S^m and Z replaced by

$$Z_m = \begin{cases} 0 & m = 0 \\ \sum_{l=0}^{m-1} S^l Z S^{m-1-l}, & 1\le m\le s-1. \end{cases} \tag{A.20}$$

The matrix $p(\mathbb{S}_k)^j$ is a shift operator with identity blocks $I_n\otimes I_s$ on the jth sub-diagonal. If we expand

$$A(\lambda) = A^{(0)} + pA^{(1)} + \cdots + p^q A^{(q)}, \qquad A^{(j)} = A^{(j,0)} + \cdots + \lambda^{s-1}A^{(j,s-1)}, \tag{A.21}$$

the matrix $\mathbb{A}_k$ representing $A(\lambda)$ is given by

$$\mathbb{A}_k = \begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_k & A_{k-1} & \cdots & A_0 \end{pmatrix}, \tag{A.22}$$

where

$$A_j = \begin{cases} \sum_{m=0}^{s-1} A^{(0,m)}\otimes S^m, & j = 0, \\ \sum_{m=0}^{s-1}\left[A^{(j,m)}\otimes S^m + A^{(j-1,m)}\otimes Z_m\right], & j\ge 1. \end{cases} \tag{A.23}$$

This formula may be derived by observing that the matrix representation of the action of $p(\lambda)^j\lambda^m A^{(j,m)}$ on $M/p^{k+1}M$ is block Toeplitz with $A^{(j,m)}\otimes S^m$ on the jth sub-diagonal and $A^{(j,m)}\otimes Z_m$ on the $(j+1)$st. Defining A_j this way avoids the need to compute remainders and quotients in subsequent steps (such as occur in Algorithm 2).

Next we seek an efficient method of computing a basis matrix $\mathbb{X}_k$ for the nullspace $\mathbb{W}_k = \ker\mathbb{A}_k$. Suppose $k\ge 1$ and we have computed $\mathbb{X}_{k-1}$. The first k blocks of equations in $\mathbb{A}_k\mathbb{X}_k = 0$ imply there are matrices $\mathbb{U}_k$ and $\mathbb{Y}_k$ such that $\mathbb{X}_k = [\mathbb{X}_{k-1}\mathbb{U}_k; \mathbb{Y}_k]$, while the last block of equations is

$$\overbrace{\begin{pmatrix} A_k & \cdots & A_0 \end{pmatrix}\underbrace{\begin{pmatrix} 0 & \mathbb{X}_{k-1} \\ I_{sn\times sn} & 0 \end{pmatrix}}_{\mathbb{X}_k}\begin{pmatrix} \mathbb{Y}_k \\ \mathbb{U}_k \end{pmatrix}}^{\mathscr{A}_k} = \begin{pmatrix} 0_{sn\times sR_k} \end{pmatrix}. \tag{A.24}$$

The matrices $\mathbb{X}_k$ can be built up recursively by setting $X_0 = \mathbb{X}_0 = \mathbb{Y}_0 = \mathrm{null}(A_0)$, defining $\mathbb{U}_0$ to be an empty matrix (with zero rows and sR_0 columns), and computing

$$\begin{aligned} &\mathscr{A}_k = \begin{pmatrix} \mathscr{A}_{k-1}, [A_k, \ldots, A_1]X_{k-1} \end{pmatrix}, \quad (k\ge 1) \\ &[Y_k; U_k] = \text{new columns of } \mathrm{null}(\mathscr{A}_k) \text{ beyond those of } \mathrm{null}(\mathscr{A}_{k-1}), \\ &[\mathbb{Y}_k; \mathbb{U}_k] = \begin{pmatrix} \mathbb{Y}_{k-1} & Y_k \\ [\mathbb{U}_{k-1}; 0] & U_k \end{pmatrix}, \qquad \begin{aligned} X_k &= [\mathbb{X}_{k-1}U_k; Y_k], \\ \mathbb{X}_k &= [\iota(\mathbb{X}_{k-1}), X_k]. \end{aligned} \end{aligned}$$

Here $\iota : K^{snl}\to K^{sn(l+1)}$ represents multiplication by p from M/p^lM to $M/p^{l+1}M$:

$$\iota([x^{(0)}; \ldots; x^{(l-1)}]) = [0; x^{(0)}; \ldots; x^{(l-1)}], \quad x^{(j)}, 0\in K^{sn}. \tag{A.25}$$

By construction, $\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k]$ is a basis for $\mathbb{W}_k$ when $k\ge 1$; it follows that $X_k + \iota(\mathbb{W}_{k-1})$ is a basis for W_k when W_k is viewed as a vector space over K. We define $X_0 = \mathbb{X}_0$ and $X_{-1} = I_{sn\times sn}$ to obtain bases for W_0 and W_{-1} as well.

But we actually want a basis for W_k viewed as a vector space over R/pR rather than K. Fortunately, all the matrices in this construction are manipulated $s \times s$ blocks, and the desired basis over R/pR may be obtained by selecting the first column from each supercolumn (group of s columns) of X_k. Indeed, if $[x_1, \ldots, x_s]$ is a supercolumn of X_k, we are able to prove that $x_j - \mathbb{S}_k x_{j-1} \in \iota(\mathbb{W}_{k-1})$ for $2 \leq j \leq s$. Since $\mathbb{S}_k$ represents multiplication by λ, these columns are all equivalent over R/pR. We are also able to prove that constructing X_k in this way (using the first column of each supercolumn) is equivalent to Algorithm 2, i.e. it yields the same unimodular matrix $V(\lambda)$ that puts $A(\lambda)$ in local Smith form. We omit the proof as it is long and technical, involving a careful comparison of nullspace calculations via row-reduction in the two algorithms.

In practice, this version of the algorithm (over K) is easier to implement, but the other version (over R/pR) should be about s times faster as the cost of multiplying two elements of R/pR is $O(s^2)$ while the cost of multiplying two $s \times s$ matrices is $O(s^3)$. The results in Section 4 were computed as described in this appendix (over $K = \mathbb{Q}$).

Acknowledgements

The authors were supported in part by the Director, Office of Science, Computational and Technology Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and by the National Science Foundation through grant DMS-0955078.

References

Avrachenkov, K.E., Haviv, M., Howlett, P.G., 2001. Inversion of analytic matrix functions that are singular at the origin. SIAM J. Matrix Anal. Appl. 22 (4), 1175–1189.
Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2001. Introduction to Algorithms. MIT Press, Cambridge, MA.
Demmel, James W., 1997. Applied Numerical Linear Algebra. SIAM, Philadelphia.
Edelman, A., Elmroth, E., Kågström, B., 1997. A geometric approach to perturbation theory of matrices and matrix pencils, part I: versal deformations. SIAM J. Matrix Anal. Appl. 18 (3), 653–692.
Gantmacher, F.R., 1960. Matrix Theory, vol. 1. Chelsea Publishing Company.
Gohberg, I., Kaashoek, M.A., van Schagen, F., 1993. On the local theory of regular analytic matrix functions. Linear Algebra Appl. 182, 9–25.
Gohberg, I., Lancaster, P., Rodman, L., 1982. Matrix Polynomials. Academic Press, New York.
Golub, Gene H., Van Loan, Charles F., 1996. Matrix Computations. John Hopkins University Press, Baltimore.
Hungerford, Thomas W., 1996. Algebra. Springer, New York.
Kailath, T., 1980. Linear Systems. Englewood Cliffs, Prentice Hall.
Kaltofen, E., Krishnamoorthy, M.S., Saunders, B.D., 1987. Fast parallel computation of Hermite and Smith forms of polynomial matrices. SIAM J. Alg. Disc. Meth. 8 (4), 683–690.
Kaltofen, E., Krishnamoorthy, M.S., Saunders, B.D., 1990. Parallel algorithms for matrix normal forms. Linear Algebra Appl. 136, 189–208.
Kannan, R., 1985. Solving systems of linear equations over polynomials. Theoret. Comput. Sci. 39, 69–88.
Labhalla, S., Lombardi, H., Marlin, R., 1996. Algorithmes de calcul de la rduction de hermite d'une matrice coefficients polynomiaux. Theoret. Comput. Sci. 161, 69–92.
Neven, W.H., Praagman, C., 1993. Column reduction of polynomial matrices. Linear Algebra Appl. 188, 569–589.
Rosenbrock, H.H., 1970. State-space and Multivariable Theory. Wiley, New York.
Storjohann, A., 1994. Computation of Hermite and Smith normal forms of matrices. Master's Thesis, Dept. of Computer Science, Univ. of Waterloo, Canada.
Storjohann, A., Labahn, G., 1997. A fast las vegas algorithm for computing the Smith normal form of a polynomial matrix. Linear Algebra Appl. 253, 155–173.
Van Dooren, P., 1979. The computation of Kronecker's canonical form of a singular pencil. Linear Algebra Appl. 27, 103–140.
Van Dooren, P., Dewilde, P., 1983. The eigenstructure of an arbitrary polynomial matrix: computational aspects. Linear Algebra Appl. 50, 545–579.
Villard, G., 1993. Computation of the Smith normal form of polynomial matrices. In: International Symposium on Symbolic and Algebraic Computation, Kiev, Ukraine. ACM Press, pp. 209–217.
Villard, G., 1994. Fast parallel computation of the Smith normal form of polynomial matrices. In: International Symposium on Symbolic and Algebraic Computation, Oxford, UK. ACM Press, pp. 312–317.
Villard, G., 1995. Generalized subresultants for computing the Smith normal form of polynomial matrices. J. Symbolic Comput. 20, 269–286.
Villard, G., 1997. Fast parallel algorithms for matrix reduction to normal forms. Appl. Alg. Eng. Comm. Comp. 8 (6), 511–538.
Wilkening, J., 2007. An algorithm for computing Jordan chains and inverting analytic matrix functions. Linear Algebra Appl. 427/1, 6–25.
Zúñiga Anaya, J.C., Henrion, D., 2009. An improved Toeplitz algorithm for polynomial matrix null-space computation. Appl. Math. Comput. 207, 256–272.