Journal of Symbolic Computation 35 (2003) 377–401

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

On lattice reduction for polynomial matrices

T. Mulders[a], A. Storjohann[b,*]

[a]COMIT AG, CH-8004 Zurich, Switzerland
[b]School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Received 27 December 2000; accepted 31 October 2002

Abstract

A simple algorithm for lattice reduction of polynomial matrices is described and analysed. The algorithm is adapted and applied to various tasks, including rank profile and determinant computation, transformation to Hermite and Popov canonical form, polynomial linear system solving and short vector computation. © 2003 Elsevier Science Ltd. All rights reserved.

Keywords: Polynomial matrices; Lattice reduction; Rank profile; Determinant; Hermite form; Popov form

1. Introduction

Let A be a matrix over $F[x]$, F a field. By applying a sequence of elementary row operations we can transform A to a matrix R which is in weak Popov form. An example is given in Fig. 1.

$$\overset{A}{\begin{bmatrix} 4x^2+3x+5 & 4x^2+3x+4 & 6x^2+1 \\ 3x+6 & 3x+5 & 3+x \\ 6x^2+4x+2 & 6x^2 & 2x^2+x \end{bmatrix}} \longrightarrow \overset{R}{\begin{bmatrix} 1 & 6x+3 & 6 \\ 0 & 0 & 0 \\ 2 & 5 & 3 \end{bmatrix}}$$

Fig. 1. Transformation of a 3×3 rank 2 matrix to weak Popov form, $F = \mathbb{Z}/(7)$.

We defer until Section 2 to define the form precisely. For now, we note two key properties of the weak Popov form:

- the number of non-zero rows of R is equal to the rank of A, and
- the sum of the degrees of the non-zero rows of R is minimal among all matrices which can be obtained from A by applying elementary row transformations.

* Corresponding author.
E-mail addresses: thom.mulders@comit.ch (T. Mulders), astorjoh@scg.math.uwaterloo.ca (A. Storjohann).

0014-5793/03/$ - see front matter
doi:10.1016/S0747-7171(02)00139-6

Thus, transformation to weak Popov form is essentially lattice reduction for polynomial matrices. The weak Popov form is a simplified, non-canonical version of the well-known Popov canonical form from linear control theory.

This paper gives a simple algorithm for transforming an input matrix over $F[x]$ to weak Popov form. We adapt and apply the algorithm to get solutions to various other problems involving polynomial matrices, see Table 1.

Table 1
Some polynomial matrix computations

Section 2	Transformation to weak Popov form.
Section 3	Computation of rank profile.
Section 4	Computation of determinant.
Section 5	Transformation of full column rank matrix to Hermite form.
Section 6	Polynomial linear system solving.
Section 7	Transformation to canonical Popov form.

The algorithms we present are designed to handle efficiently the case of input matrices which may be rectangular and/or rank deficient. Consider the well understood case of matrices over a field. Let $A \in F^{n\times m}$ have rank r. Problems involving A like linear system solving and rank profile computation can be solved with $O(nmr)$ field operations using Gaussian elimination. This paper gives analogous results for matrices over $F[x]$. Let $A \in F[x]^{n\times m}$ have rank r and degree bounded by d, where the degree of a polynomial matrix is defined as being the maximum of the degree of its entries. We show that all the problems listed in Table 1 can be solved with $O(nmrd^2)$ field operations. Note that when r and d appear in a big-O bounds they should be taken as upper bounds, that is, $r > 0$ and $d > 0$.

An algorithm to compute a reduced basis very similar to the weak Popov form has been given by von zur Gathen (1984) and applied to the problem of computing short vectors. In Section 8 we indicate the relationship between the Popov form and *reduced basis* as defined there. This results in a substantially faster algorithm for the reduced basis and short vectors problem.

In Section 9 we extend the notion of weak Popov form to the setting of discrete valuation rings. Analogous results as in the polynomial setting hold. In Section 10 we end the paper with a short summary, some remarks on implementation issues and some suggestions for further research.

1.1. Cost model

We assume we have primitives for polynomial arithmetic which support the following cost bounds. Let $a, b \in F[x]$ be non-zero. Then $a+b$ and $a-b$ can be computed with $O(1+\max(\deg(a), \deg(b)))$ field operations, ab can be computed with $O((1+\deg a)(1+\deg b))$ field operations, and if $\deg a \geq \deg b$, then the unique $q, r \in F[x]$ with $a = bq + r$ and $\deg r < \deg b$ can be computed with $O((1 + \deg a - \deg b)(1 + \deg b))$ field operations. The algorithms in this paper are deterministic. Allowing randomization, asymptotically faster algorithms are known in some cases. For each problem we mention the currently best known complexity bound. Some of these randomized algorithms allow use of asymptotically fast matrix or polynomial multiplication. Let θ ($2 < \theta \leq 3$) be such that two $n \times n$ matrices over a field can be multiplied together with $O(n^\theta)$ field operations. Let ϵ ($0 < \epsilon \leq 1$) be such that two degree d polynomials can be multiplied together with $O(d^{1+\epsilon})$ field operations.

2. The weak Popov form

A well-known notion in systems theory is the Popov form (Popov, 1969) of a rectangular matrix with polynomial entries. A non-canonical but still useful version of the Popov form is the quasi Popov form (Kailath, 1980). In this section we define the weak Popov form—a form with even less conditions than the quasi Popov form.

Let F be a field and $M = (m_{i,j}) \in F[x]^{n\times m}$. In what follows we use M to define general notions for matrices. We use calligraphic characters to refer to specific variables used in the various algorithms.

Definition. For $1 \leq i \leq n$ we define the ith *pivot index* I_i^M of M as follows: if $m_{i,j} = 0$ for $1 \leq j \leq m$, then $I_i^M = 0$; otherwise

1. $\deg(m_{i,j}) \leq \deg(m_{i,I_i^M})$ for $1 \leq j < I_i^M$;
2. $\deg(m_{i,j}) < \deg(m_{i,I_i^M})$ for $I_i^M < j \leq m$.

When $I_i^M \neq 0$, the element m_{i,I_i^M} is called the ith *pivot element* of M and is denoted by P_i^M. The degree of P_i^M is called the ith *pivot degree* of M and is denoted by D_i^M. When $I_i^M = 0$ we put $D_i^M = -1$.

A pivot element is the rightmost element with maximal degree in its row.

Definition. The *carrier set* C^M of M is defined as $C^M = \{1 \leq i \leq n \mid I_i^M \neq 0\}$.

Definition. M is said to be in weak Popov form if the positive pivot indices of M are all different, i.e. if

$$k, l \in C^M, \qquad k \neq l \quad \Rightarrow \quad I_k^M \neq I_l^M.$$

By applying unimodular row-transformations, we want to transform a given matrix to weak Popov form. We now define a particularly simple kind of unimodular transformation.

Definition. If $k \in C^M$, $l \neq k$ and $\deg(m_{l,I_k^M}) \geq D_k^M$, there are unique $c \in F$ and $e \in \mathbb{N}$ such that

$$\deg(m_{l,I_k^M} - cx^e P_k^M) < \deg(m_{l,I_k^M}).$$

In that case we call subtracting cx^e times row k from row l the *simple transformation* of row k on row l. If $I_l^M = I_k^M$, the transformation is called of the *first kind*, otherwise it is called of the *second kind*.

algorithm WeakPopovForm
input: $\mathcal{M} \in F[x]^{n\times m}$.
output: $\mathcal{N}$ in weak Popov form, obtained by applying simple transformations of the first kind on $\mathcal{M}$.
$\mathcal{A} := \mathrm{copy}(\mathcal{M})$;
while $\mathcal{A}$ is not in weak Popov form **do**
    Apply a simple transformation of the first kind on $\mathcal{A}$
od;
$\mathcal{N} := \mathrm{copy}(\mathcal{A})$;
return $\mathcal{N}$

Fig. 2. Algorithm *WeakPopovForm*.

Sometimes we want to apply a simple transformation on M and simultaneously apply the same transformation on a vector or matrix A. We then say that we apply the transformation on $[\,M \mid A\,]$. Note that we only consider M when we determine the pivot element of a row.

Definition. When $[\,N \mid B\,]$ is the result after applying a number of simple transformations on $[\,M \mid A\,]$, we write $[\,M \mid A\,] \to [\,N \mid B\,]$. Note that in that case $[\,N \mid B\,]$ is left equivalent to $[\,M \mid A\,]$, i.e. $[\,N \mid B\,] = U[\,M \mid A\,]$ where U is unimodular and even $\det(U) = 1$.

Example 1. Let

$$M = \begin{bmatrix} 1 & x^2 & x \\ 3x & x + 2x^3 & x^3 \end{bmatrix}, \qquad A = \begin{bmatrix} x^4 \\ x^2 \end{bmatrix}$$

and

$$N = \begin{bmatrix} 1 & x^2 & x \\ x & x & x^3 - 2x^2 \end{bmatrix}, \qquad B = \begin{bmatrix} x^4 \\ x^2 - 2x^5 \end{bmatrix}.$$

Then $I_1^M = 2$ and by applying the simple transformation of the first row on the second row of $[\,M \mid A\,]$, we see that $[\,M \mid A\,] \to [\,N \mid B\,]$.

Algorithm *WeakPopovForm*, shown in Fig. 2, transforms a matrix by applying simple transformations of the first kind. The algorithm is based on the following trivial lemma.

Lemma 2.1. *M is not in weak Popov form if and only if we can apply a simple transformation of the first kind on M, that is, not all non-zero pivot indices of M are different.*

We remark that the copying of matrices is done only in order to be able to reason about the algorithm. Correctness of the algorithms output follows from Lemma 2.1. That the algorithm always terminates will follow as a corollary of our cost analysis.

The next lemma notes how the pivot indices and pivot degrees may change when we apply a simple transformation.

Lemma 2.2. *Let N be the matrix we get after applying the simple transformation of row k on row l of M. If the simple transformation is of the first kind, then either $D_l^N < D_l^M$ or $(D_l^N = D_l^M$ and $I_l^N < I_l^M)$. If the simple transformation is of the second kind, then $I_l^N = I_l^M$ and $D_l^N = D_l^M$.*

Now we bound the cost of algorithm *WeakPopovForm*. For this, the following corollary of Lemma 2.2 is important.

Corollary 2.1. *If d is a bound on the degree of $\mathcal{M}$, then the degree of $\mathcal{A}$ is always bounded by d.*

Now we describe the possible values that a pair $(D_l^{\mathcal{A}}, I_l^{\mathcal{A}})$ can assume during the course of algorithm *WeakPopovForm*.

Definition. The set $I^M = \{I_i^M \mid i \in C^M\}$ of non-zero pivot indices of M is called the *index set* of M.

The next two lemmas follow from Lemma 2.2 and the definitions of a simple transformation of the first and second kind.

Lemma 2.3. *If N is the matrix we get after applying a simple transformation on M, then $I^M \subseteq I^N$.*

Lemma 2.4. *For $1 \leq l \leq n$, the values that the pair $(D_l^{\mathcal{A}}, I_l^{\mathcal{A}})$ can assume during the course of algorithm WeakPopovForm are all in the set $\{D_l^{\mathcal{N}}, D_l^{\mathcal{N}} + 1, \ldots, D_l^{\mathcal{M}}\} \times (I^{\mathcal{N}} \cup \{0\})$.*

Lemma 2.5. *If the pivot indices of all rows of M are positive and different, then the rows of M are independent over $F(x)$.*

Proof. Let N be the matrix we get by multiplying, for $1 \leq i \leq n$, row i by $x^{-D_i^M}$. Then $N = N_0 + \hat{N}$, where $\hat{N} \in x^{-1}F[x^{-1}]^{n\times m}$ and $N_0 \in F^{n\times m}$ has independent rows. Consider $F(x) \subset F((x^{-1}))$. It is clear that the rows of N are independent over $F((x^{-1}))$ and thus are also independent over $F(x)$. □

Corollary 2.2. $\mathrm{Rank}(M) \geq \#I^M$.

Theorem 2.1. *Algorithm WeakPopovForm is correct. The cost of the algorithm is bounded by $O(nmrd^2)$ field operations, where r is the rank of $\mathcal{M}$ and d is a bound on the degree of $\mathcal{M}$.*

Proof. From Lemma 2.4 it follows that, during the course of the algorithm, the pair $(D_l^{\mathcal{A}}, I_l^{\mathcal{A}})$ can assume at most $(D_l^{\mathcal{M}} + 2)(\#I^{\mathcal{N}} + 1)$ values. Since $\mathrm{rank}(\mathcal{N}) = \mathrm{rank}(\mathcal{M})$, it follows from Corollary 2.2 that $\#I^{\mathcal{N}} = r$. By Lemma 2.2, every simple transformation of the first kind decreases, for one l, the pair $(D_l^{\mathcal{A}}, I_l^{\mathcal{A}})$ in the lexicographic order. It follows that the number of simple transformations applied during the course of the algorithm is

$O(nrd)$. By Corollary 2.1 the cost of one simple transformation is bounded by $O(md)$ field operations. □

To be able to compute the amortized cost of some algorithms we have to specify in more detail the number of simple transformations applied by algorithm *WeakPopovForm*.

Definition. The *state* S^M of M is defined by

$$S^M = \sum_{i \in C^M} (D_i^M m + I_i^M).$$

Lemma 2.6. $S^M \geq 0$. *Moreover, when N is the matrix we get after applying a simple transformation of the first kind on M, then $S^N < S^M$.*

So the state of M is a bound on the number of simple transformations of the first kind it will take to transform M into weak Popov form.

Definition. If $M \to N$, the *state drop* $S^{M,N}$ from M to N is defined by $S^{M,N} = S^M - S^N$.

The next result follows immediately from the definition of the state drop.

Theorem 2.2. *The number of simple transformations applied by algorithm WeakPopov-Form is at most $S^{\mathcal{M},\mathcal{N}}$.*

In fact S^M can also be defined with m replaced by $r = \text{rank}(M)$ and Theorem 2.2 then still holds. Since the proof is more involved, and we do not need this result in what follows, we restrict ourselves to the current definition.

3. The rank profile

In this section we show how algorithm *WeakPopovForm* can be adjusted to compute the rank profile of a matrix $A \in F[x]^{n\times m}$. Recall that the *column rank profile* of A is the lexicographically smallest list of row indices $[i_1, i_2, \ldots, i_r]$ such that these rows of A are linearly independent, where r is the rank of A. The column rank profile is thus named because it describes the echelon structure of the column echelon form of A. The row rank profile is defined analogously, and is equal to the column rank profile of the transpose.

The rank profile over $F[x]$ can be recovered with high probability by computing the rank profile modulo a small degree and randomly chosen irreducible polynomial. This Monte Carlo algorithm requires about $O(nmr^{\theta-2} + nmd)$ field operations. The cost estimate might increase by a poly-logarithmic factor in the case of small fields.

Algorithm *RankProfile*, shown in Fig. 3, computes the rank profile deterministically. We get the following as a corollary of Theorem 2.1.

Theorem 3.1. *Algorithm RankProfile is correct. The cost of the algorithm is bounded by $O(nmrd^2)$ field operations, where r is the rank of $\mathcal{M}$ and d is a bound on the degree of $\mathcal{M}$.*

algorithm RankProfile
input: $\mathcal{M} \in F[x]^{n\times m}$.
output: the column rank profile of $\mathcal{M}$.
$r := 0$;
$\mathcal{A} :=$ the $0 \times m$ matrix;
for i **to** n **do**
    Augment $\mathcal{A}$ with row i of $\mathcal{M}$;
    $\mathcal{A} :=$ WeakPopovForm($\mathcal{A}$);
    if rank($\mathcal{A}$) $= r + 1$ **then**
        $r := r + 1$;
        $i_r := i$
    fi
od;
return $[i_1, i_2, \ldots, i_r]$

Fig. 3. Algorithm *RankProfile*.

algorithm ExtendedWeakPopovForm
input: $\mathcal{M} \in F[x]^{n\times m}, \mathcal{V} \in F[x]^n$.
output: $\left[\ \mathcal{N} \mid \mathcal{W}\ \right]$ with $\mathcal{N}$ in weak Popov form, obtained by applying simple transformations of the first kind on $\left[\ \mathcal{M} \mid \mathcal{V}\ \right]$.
$(\mathcal{A},\mathcal{U}) :=$ copy($\mathcal{M}, \mathcal{V}$);
while $\mathcal{A}$ is not in weak Popov form **do**
    Apply a simple transformation of the first kind on $\left[\ \mathcal{A} \mid \mathcal{U}\ \right]$
od;
$(\mathcal{N}, \mathcal{W}) :=$ copy($\mathcal{A},\mathcal{U}$);
return $\left[\ \mathcal{N} \mid \mathcal{W}\ \right]$

Fig. 4. Algorithm *ExtendedWeakPopovForm*.

4. The determinant

In this section we show how algorithm *WeakPopovForm* can be adjusted to compute the determinant of a matrix $A \in F[x]^{n\times n}$. The determinant will have degree bounded by nd, where d is a bound on the degree of A. The algorithm we propose here computes $\det(A)$ with $O(n^3d^2)$ field operations.

Using randomization and a completely different approach, Storjohann (2002) gives a Las Vegas probabilistic algorithm that requires an expected number of $O(n^\theta(\log n)^2 d^{1+\epsilon})$ field operations. The cost estimate might increase by a poly-logarithmic factor in the case of small fields. Also, the $O((\log n)^2)$ factor is present even in the case $\theta = 3$.

Algorithm *ExtendedWeakPopovForm*, shown in Fig. 4, applies simple transformations on $\mathcal{M}$ to obtain the weak Popov form $\mathcal{N}$ and applies the same transformations on the vector $\mathcal{V}$, obtaining $\mathcal{W}$. To estimate the cost of algorithm *ExtendedWeakPopovForm* we have to bound the degree of $\mathcal{U}$.

Definition. The *degree sum* D^M of M is defined by

$$D^M = \sum_{i=1}^{n} D_i^M.$$

Lemma 4.1. *If N is the matrix we get after applying a simple transformation on M, then $D^N \le D^M$.*

Proof. This follows immediately from Lemma 2.2. $\square$

Definition. If $M \to N$, the *degree drop* $D^{M,N}$ is defined by $D^{M,N} = D^M - D^N$.

Lemma 4.2. *Let $v \in F[x]^n$ and assume that $[\, M \mid v \,] \to [\, N \mid w \,]$. If $c \in \mathbb{Z}$ is such that $\deg(v_i) \le D_i^M + c$ for all i, then $\deg(w_i) \le D_i^N + c + D^{M,N}$ for all i.*

Proof. Since degree drop is additive, we only have to prove the lemma when applying one simple transformation. Suppose we apply the simple transformation of row k on row l. For $i \ne l$ we have $\deg(w_i) = \deg(v_i) \le D_i^M + c = D_i^N + c \le D_i^N + c + D^{M,N}$, since $D^{M,N} \ge 0$ by Lemma 4.1. Let $j = I_k^M$ and $M = (m_{i,j})$. Then

$$\begin{aligned} \deg(w_l) &\le \max(\deg(v_l), \deg(m_{l,j}) - \deg(m_{k,j}) + \deg(v_k)) \\ &\le \max(D_l^M + c, D_l^M - D_k^M + D_k^M + c) \\ &= D_l^M + c. \end{aligned}$$

Since $D^{M,N} = D_l^M - D_l^N$ we have $D_l^M + c = D_l^N + c + D^{M,N}$. $\square$

Theorem 4.1. *The cost of algorithm ExtendedWeakPopovForm is bounded by $O((m+n)dS^{\mathcal{M},\mathcal{N}})$ field operations, where d is a bound on the degree of $\mathcal{M}$ and $\mathcal{V}$.*

Proof. By Theorem 2.2 at most $S^{\mathcal{M},\mathcal{N}}$ simple transformations are applied. By Corollary 2.1 the degree of $\mathcal{A}$ is always bounded by d. Since $\deg(\mathcal{V}_i) \le D_i^{\mathcal{M}} + d + 1$ for all i and always $D^{\mathcal{M},\mathcal{A}} \le n(d+1)$, it follows from Lemma 4.2 that the degree of $\mathcal{U}$ is always bounded by $d + (d+1) + n(d+1) = O(nd)$. From this the theorem follows. $\square$

Let $T \in F[x]^{n\times n}$. Write $T = [\, M \mid V \,]$, where M consists of the first $n-1$ rows of T and V is the last column of T. Apply algorithm *ExtendedWeakPopovForm* on the pair (M, V) yielding $[\, N \mid W \,]$. Since N is in weak Popov form and $\operatorname{rank}(N) = \operatorname{rank}(M) \le n-1$, it follows from Corollary 2.2 that N will contain at least one zero row. So up to a row permutation we have

$$[\, N \mid W \,] = \left[\begin{array}{c|c} \bar{T} & * \\ 0 & t \end{array}\right],$$

where $\bar{T} \in F[x]^{(n-1)\times(n-1)}$ and $t \in F[x]$. Thus, up to sign we have $\det(T) = \det(\bar{T})\, t$. This leads to algorithm *Determinant* shown in Fig. 5. Fig. 6 is (up to row permutation) a pictorial representation of the flow of algorithm *Determinant*. Here, the dark gray areas represent $\mathcal{M}$ and $\mathcal{N}$, the middle gray areas represent $\mathcal{V}$ and $\mathcal{W}$, the light gray areas are

algorithm Determinant
input: $\mathcal{T} \in F[x]^{n\times n}$.
output: $\det(\mathcal{T})$.
$\bar{\mathcal{T}} := \mathrm{copy}(\mathcal{T})$;
$\det := 1$;
for i **from** $n-1$ **by** -1 **to** 1 **do**
  $\mathcal{M} :=$ first i columns of $\bar{\mathcal{T}}$;
  $\mathcal{V} :=$ last column of $\bar{\mathcal{T}}$;
  $[\ \mathcal{N} \mid \mathcal{W}\ ] := \mathrm{ExtendedWeakPopovForm}(\mathcal{M}, \mathcal{V})$;
  Let k be such that the kth row of $\mathcal{N}$ is zero;
  $\bar{\mathcal{T}} := \mathcal{N}$ with row k deleted;
  $t := k$th entry of $\mathcal{W}$;
  $\det := (-1)^{k+i+1}\, t \det$
od;
return $\bar{\mathcal{T}}_{1,1} \det$

Fig. 5. Algorithm *Determinant*.

ignored during the computation and the white areas represent zero entries. The determinant of the matrix is (up to sign) the product of the black entries.

Theorem 4.2. *The cost of algorithm Determinant is bounded $O(n^3d^2)$ field operations, where d is a bound on the degree of $\mathcal{T}$.*

Proof. By Corollary 2.1 the degrees of $\bar{\mathcal{T}}, \mathcal{M}, \mathcal{V}$ and $\mathcal{N}$ are always bounded by d. Let $\mathcal{M}_{n-1}, \mathcal{M}_{n-2}, \ldots, \mathcal{M}_1$ be the consecutive values of $\mathcal{M}$ and $\mathcal{N}_{n-1}, \mathcal{N}_{n-2}, \ldots, \mathcal{N}_1$ the consecutive values of $\mathcal{N}$ during the course of the algorithm. By Theorem 4.1 the cost is then bounded by

$$O\left(nd \sum_{i=1}^{n-1} S^{\mathcal{M}_i,\mathcal{N}_i}\right).$$

If $i \notin I^{\mathcal{N}_i}$, then $D_l^{\mathcal{M}_{i-1}} = D_l^{\mathcal{N}_i}$, $I_l^{\mathcal{M}_{i-1}} = I_l^{\mathcal{N}_i}$ for all i and thus $S^{\mathcal{M}_{i-1}} = S^{\mathcal{N}_i}$. If k is such that $I_k^{\mathcal{N}_i} = i$, then $D_l^{\mathcal{M}_{i-1}} = D_l^{\mathcal{N}_i}$, $I_l^{\mathcal{M}_{i-1}} = I_l^{\mathcal{N}_i}$ for $l \neq k$, $D_k^{\mathcal{M}_{i-1}} \leq D_k^{\mathcal{N}_i}$ and $I_k^{\mathcal{M}_{i-1}} < I_k^{\mathcal{N}_i}$ and thus $S^{\mathcal{M}_{i-1}} < S^{\mathcal{N}_i}$. So

$$\sum_{i=1}^{n-1} S^{\mathcal{M}_i,\mathcal{N}_i} = S^{\mathcal{M}_{n-1}} - \sum_{i=2}^{n-1}(S^{\mathcal{N}_i} - S^{\mathcal{M}_{i-1}}) - S^{\mathcal{N}_1} \leq S^{\mathcal{M}_{n-1}}.$$

Since $S^{\mathcal{M}_{n-1}} = O(n^2d)$, the theorem follows. □

5. The Hermite form

Let A over $F[x]$ have full column rank. The Hermite form H of A is the unique upper triangular matrix which is left equivalent to A, has diagonal entries monic, and off-diagonal

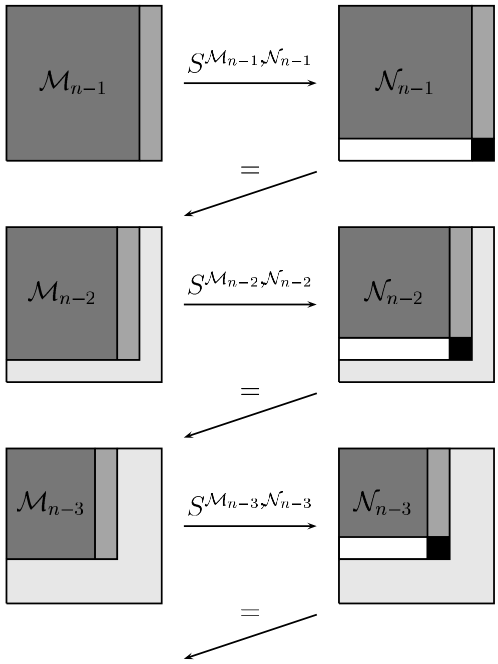

Fig. 6. Flow of algorithm *Determinant*.

entries of degree less than the diagonal entry in the same column, see MacDuffee (1956) or Newman (1972). In this section we show how algorithm *Determinant* can be adjusted to compute the Hermite form of a non-singular input matrix $A \in F[x]^{m\times m}$. The cost of the algorithm is $O(m^3d^2)$ field operations, where d is a bound on the degree of A. The algorithm extends immediately to rectangular input matrices of full column rank by first computing the weak Popov form and restricting to the non-zero rows.

Different approaches to computing the Hermite form have been given. Domich et al. (1987) work modulo the determinant of the input matrix to avoid intermediate expression swell. Labhalla et al. (1996) transform the original problem over $F[x]$ to that of triangularizing a larger matrix over F. Villard (1996) deduces the Hermite form from the Popov form, computed via a matrix gcd using a block Hankel construction, see Section 7. The $O(m^3d^2)$ field operations algorithm we give here, based on lattice reduction, is the first with a complexity bound that is cubic in the matrix dimension. For comparison, the approach of Domich et al. (1987) has cost $O(m^3(md)^{1+\epsilon})$ field operations.

In algorithm *Determinant* we ignored the last columns of the matrix when applying transformations, see Fig. 6. That algorithm recovered the diagonal entries of the Hermite form but not the off-diagonal entries. If instead we apply all transformations to the whole matrix, we would be left with a triangularization. One could finally use the diagonal entries to lower the degree of the off-diagonal entries, yielding a matrix in Hermite form. The problem with this approach is that the degrees of the off-diagonal entries may become too

algorithm HermiteForm
input: $\mathcal{T} \in F[x]^{n\times m}$ with full column rank.
output: $\mathcal{H}$ in Hermite form, left equivalent to $\mathcal{T}$.
$\bar{\mathcal{T}}$:=WeakPopovForm($\mathcal{T}$);
$\mathcal{M} :=$ first $n-1$ columns of $\bar{\mathcal{T}}$;
$\mathcal{V} :=$ last column of $\bar{\mathcal{T}}$;
$\mathcal{A} :=$ empty matrix;
for i **from** $n-1$ **by** -1 **to** 1 **do**
  while $\mathcal{M}$ is not in weak Popov form **do**
    Apply a simple transformation of the first kind on $\left[\begin{array}{c|cc}\mathcal{M} & \mathcal{V} & \mathcal{A}\end{array}\right]$, say from row k
    on row l;
    (1) Use $\bullet$–entries to lower degrees of entries in lth row of $\mathcal{A}$
  od;
  (2) Use $\clubsuit$–entry to lower degrees in $\mathcal{V}$;
  Let l such that $I_l^{\mathcal{M}} = i$;
  $\mathcal{M} :=$ first $i-1$ columns of $\left[\begin{array}{ccc}\mathcal{M} & \mathcal{V} & \mathcal{A}\end{array}\right]$;
  $\mathcal{V} :=$ ith column of $\left[\begin{array}{ccc}\mathcal{M} & \mathcal{V} & \mathcal{A}\end{array}\right]$;
  $\mathcal{A} :=$ last $n-i$ columns of $\left[\begin{array}{ccc}\mathcal{M} & \mathcal{V} & \mathcal{A}\end{array}\right]$;
  (3) Use $\bullet$ entries to lower degrees of entries in lth row of $\mathcal{A}$
od;
$\mathcal{H} := \left[\begin{array}{cc}\mathcal{V} & \mathcal{A}\end{array}\right]$ with rows permuted to make it upper triangular;
Multiply rows of $\mathcal{H}$ with constants to make diagonal entries monic;
return $\mathcal{H}$

Fig. 7. Algorithm *HermiteForm*.

high, thus leading to a bad complexity. In order to avoid these high degrees we will apply, during the course of the algorithm, extra elementary transformations.

Fig. 7 gives a description of algorithm *HermiteForm* to transform a full column rank matrix into Hermite form. The details of steps (1), (2) and (3) will be explained shortly. Fig. 8 is a pictorial representation of algorithm *HermiteForm*. Here the dark gray columns represent $\mathcal{M}$, the middle gray columns represent $\mathcal{V}$ and the light gray columns represent $\mathcal{A}$.

Fig. 9 represents (up to row permutation) the actions of one iteration during the inner while loop. Here $D_s = D_s^{\mathcal{M}}$ and for $j > i$, d_j is the degree of the bullet entry in column j. The idea is to let $[\,\mathcal{M} \mid V \quad \mathcal{A}\,]$ always have the following property.

Property 1. *For $j > i+1$ and $s < j$ the degree of entry (s, j) of $[\,\mathcal{M} \mid \mathcal{V} \quad \mathcal{A}\,]$ is at most $D_s + d_j$.*

So Property 1 ensures that the degrees of the entries in the light gray area are not too big. Note that for $s > i+1$ we have $D_s = -1$ and thus when $[\,\mathcal{M} \mid \mathcal{V} \quad \mathcal{A}\,]$ has Property 1, then for $i+1 < s < j$ the degree of entry (s, j) is less than d_j. This means that the lower triangular part of $\mathcal{A}$ is in Hermite form, and at the end of algorithm *HermiteForm* $[\,\mathcal{V} \quad \mathcal{A}\,]$ is in Hermite form.

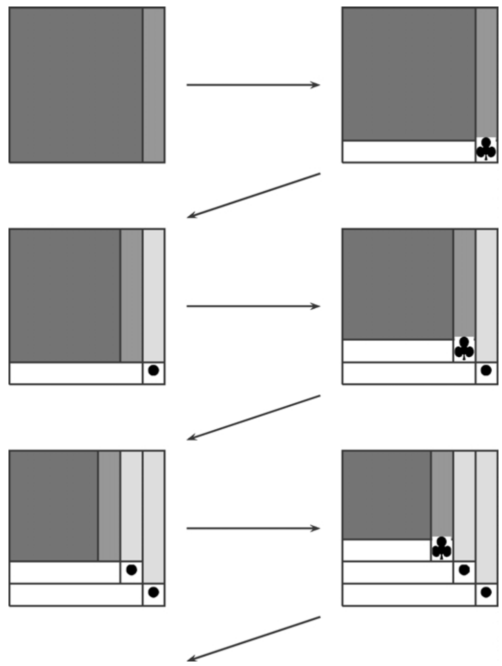

Fig. 8. Flow of algorithm *HermiteForm*.

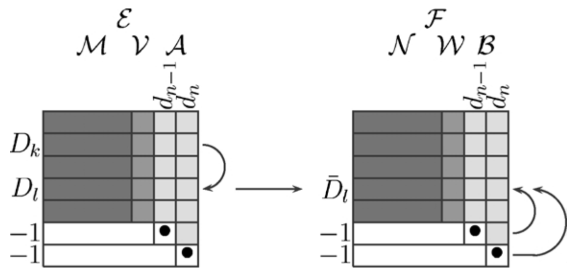

Fig. 9. One iteration during **while** loop.

Suppose $\mathcal{E} = [\,\mathcal{M} \mid \mathcal{V} \quad \mathcal{A}\,]$ has Property 1 and let $\mathcal{F} = [\,\mathcal{N} \mid \mathcal{W} \quad \mathcal{B}\,]$ be the matrix we get after applying on $[\,\mathcal{M} \mid \mathcal{V} \quad \mathcal{A}\,]$ the simple transformation of the first kind from row k on row l. Let $\bar{D}_l = D_l^{\mathcal{N}}$. For $j > i + 1$ we have by Lemma 4.2 $\deg(\mathcal{F}_{l,j}) \le \bar{D}_l + d_j + D^{\mathcal{M},\mathcal{N}} = D_l + d_j$. So if $\bar{D}_l = D_l$, $[\,\mathcal{N} \mid \mathcal{W} \quad \mathcal{B}\,]$ still has Property 1 and nothing has to be done in step (1). If however $\bar{D}_l < D_l$, the entries in the lth row of $[\,\mathcal{N} \mid \mathcal{W} \quad \mathcal{B}\,]$ may violate Property 1 and we have to restore the property in step (1). Let q be the quotient of $\mathcal{F}_{l,i+2}$ by $\mathcal{F}_{i+2,i+2}x^{\bar{D}_l+1}$, i.e. $\deg(\mathcal{F}_{l,i+2} - q\mathcal{F}_{i+2,i+2}x^{\bar{D}_l+1}) \le \bar{D}_l + \deg(\mathcal{F}_{i+2,i+2})$. Then $\deg(q) < D_l - \bar{D}_l$. Let $\mathcal{G}$ be the result of subtracting q times row $i + 2$ from

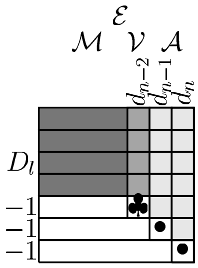

Fig. 10. Snapshot after **while** loop.

row l in $\mathcal{F}$. Then the $(l, i+2)$ entry of $\mathcal{G}$ has degree at most $\bar{D}_l + \deg(\mathcal{G}_{i+2,i+2})$ and thus satisfies Property 1. Moreover, for $j > i+2$

$$\begin{aligned}\deg(\mathcal{G}_{l,j}) &\leq \max(\deg(\mathcal{F}_{l,j}), \deg(q) + \deg(\mathcal{F}_{i+2,j})) \\ &\leq D_l + d_j\end{aligned}$$

since $\deg(\mathcal{F}_{l,j}) \leq D_l + d_j$, $\deg(q) < D_l$ and $\deg(\mathcal{F}_{i+2,j}) \leq d_j - 1$. Subtracting in a similar way in sequence multiples of rows $i+3, \ldots, n$ from row l, we restore Property 1 for row l in step (1).

Now we describe step (2). Fig. 10 represents (up to row permutation) the situation just after the while loop has completed. Before we enlarge $\mathcal{A}$ with column $\mathcal{V}$, we make sure that the entries in $\mathcal{V}$ satisfy Property 1, i.e. make $\deg(\mathcal{V}_l) \leq D_l + d_{i+1}$. Step (2) takes care of this. We could apply row transformations as in step (1), using the ♣- and •-entries, for this. However, this would be too costly.

Let s be the maximum degree excess in column $\mathcal{V}$, that is, for $1 \leq l \leq i$ we have $\deg(\mathcal{V}_i) \leq D_l + d_{i+1} + s$. Let $\mathcal{Q}^0$ be the $(i+1)$th row of $\mathcal{E}$. For $u = 1, \ldots, s$ let $\mathcal{Q}^u$ be the row vector we get by multiplying $\mathcal{Q}^{u-1}$ by x and, like in step (1), reducing all entries from left to right using rows $i+2, \ldots, n$ of $\mathcal{E}$. Then $\deg(\mathcal{Q}^u_{i+1}) = d_{i+1} + u$ and $\deg(\mathcal{Q}^u_j) < d_j$ for $j > i+1$. Now we can add appropriate monomial multiples of the $\mathcal{Q}^u$ to rows $1, \ldots, i$ to make the entries of $\mathcal{V}$ satisfy Property 1. Notice that this does not destroy Property 1 for the entries in $\mathcal{A}$.

Finally, we describe step (3). When the last column of $\mathcal{M}$ is deleted before we enter the while loop again, $D_l^{\mathcal{M}}$ may decrease and thus the entries in the lth row may violate Property 1. In step (3) we then apply the same procedure as in step (1) to make sure that the entries in row l satisfy the property again.

Theorem 5.1. *The cost of algorithm HermiteForm is bounded by $O(nm^2d^2)$ field operations, where d is a bound on the degree of $\mathcal{T}$.*

Proof. By Theorem 2.1 computing $\bar{\mathcal{T}}$ can be accomplished in the allotted time.

By Corollary 2.1 the degree $\mathcal{M}$ is bounded by d. By Lemma 4.2 the entries in $\mathcal{V}$ have degree bounded by $O(md)$. The sum of the degrees of the entries in one row of $\mathcal{A}$ is at most $\sum_{j=i+2}^{m}(d + d_j)$. Since the product of all •-entries divides the determinant of $\bar{\mathcal{T}}$ we have $\sum_{j=i+1}^{m}(d + d_j) = O(md)$.

As in the proof of Theorem 4.2 we see that the number of simple transformations applied is bounded by $S^{\bar{\mathcal{T}}} = O(m^2 d)$. One simple transformation costs $O(md)$ and thus the cost of all simple transformations is $O(m^3 d^2)$.

Adding a multiple of a row as in step (1) costs $O((D_l - \bar{D}_l)md)$ and thus performing step (1) once takes $O((D_l - \bar{D}_l)m^2 d)$. Since the total degree drop, i.e. the sum of all $D_l - \bar{D}_l$ is at most md, the total cost of all steps (1) is $O(m^3 d^2)$.

The cost of performing step (2) once is bounded by $O(sm^2 d)$. By Lemma 4.2 s is bounded by the sum of d and the degree drop during the last invocation of the while loop. So the sum of all s during the algorithm is $O(md)$ and thus the total cost of all steps (2) is $O(m^3 d^2)$.

As in step (1), the cost of step (3) is bounded by $O(m^3 d^2)$ and making the diagonal entries monic can be accomplished with $O(m^2 d)$. □

5.1. Triangular factorization

Algorithm *HermiteForm* can be used to obtain a triangular factorization of a full column rank $A \in F[x]^{n\times m}$, that is, compute the Hermite form H of A together with a unimodular matrix V such that $A = VH$. Proceed as follows.

Compute the column rank profile of A and, if necessary, permute the rows so that the first m rows are linearly independent. For simplicity, assume no permutation of rows is required. Append to A the $(n-m) \times (n-m)$ identity matrix to the right bottom, yielding the non-singular matrix

$$\bar{A} = \left[\begin{array}{c|c} A & \begin{array}{c} 0 \\ I \end{array} \end{array}\right] \in F[x]^{n\times n}.$$

Now compute the Hermite form $\bar{H}$ of $\bar{A}$. Let H be the first m columns of $\bar{H}$.

Finally, we are going to compute $V = \bar{A}\bar{H}^{-1}$. To do this efficiently, let D_i be the $n \times n$ identity matrix except with ith diagonal entry equal to that of $\bar{H}$. Similarly, let E_i be the $n \times n$ identity matrix except with off-diagonal entries in the ith column equal to those of $\bar{H}$. Then $\bar{H} = D_n E_n \cdots D_3 E_3 D_2 E_2 D_1 E_1$. Compute $V = \bar{A} E_1^{-1} D_1^{-1} E_2^{-1} D_3^{-1} \cdots E_n^{-1} D_n^{-1}$, evaluating from left to right.

Theorem 5.2. *Let $A \in F[x]^{n\times m}$ have rank m and degree bounded by d. A triangular factorization of A can be computed with $O(n^3 d^2)$ field operations. Moreover, the degree of the unimodular transformation matrix is bounded by d.*

Proof. Use the method described above. Determine the m independent rows of A and compute $\bar{H}$ using algorithms *RankProfile* and *HermiteForm*. Because $\bar{H}$ is in Hermite form, we have the bounds $\deg(\det(\bar{H})\bar{H}^{-1}) \le \deg(\det(\bar{H}))$ and $\sum_{1\le j\le n} \deg(E_i) \le \sum_{1\le j\le n} \deg(D_i) = \deg(\det(\bar{H})) = O(md)$. The former shows $\deg(V) \le \deg(\bar{A})$. Using this and the latter bound it follows that V can be computed as indicated with $O(n^2 md^2)$ field operations. □

6. Polynomial linear system solving

Let $M \in F[x]^{n\times m}$ and $b \in F[x]^{1\times m}$ be given. This section shows how to solve the polynomial linear system $vM = b$ in the following general sense:

1. If the system does not have a rational solution, that is, if there does not exist a $v \in F(x)^{1\times n}$ such that $vM = b$, then report this.
2. If the system does have a rational solution, then find the minimal degree monic $e \in F[x]$ such that $vM = eb$ has a polynomial solution, and
3. find a particular solution $v \in F[x]^{1\times n}$ for $vM = eb$.

These problems have been well studied. Let r be the rank of M and d be a bound on the degree of M. The complexity bounds we state allow the target vector b to have degree as large as $O(rd)$. Mulders and Storjohann (2000b) solve problem 1 with $O((n+m)r^2d^{1+\epsilon})$ field operations. A rational solution vector, if one exists, is computed in the same time. Problems 2 and 3 are more subtle. The fastest methods are based on randomized preconditioning. The Las Vegas algorithm of Mulders and Storjohann (to appear) solves all the problems using an expected number of $O((nmr^{\theta-2} + r^\theta(\log r))(d + \log_{\#F} r)^{1+\epsilon})$ field operations. If $\theta = 3$ the $\log r$ factor can be avoided and the result becomes $O(nmr(d+\log_{\#F} r)^{1+\epsilon})$ field operations. Here we show how to solve the problems without randomization with $O(nmrd^2)$ field operations.

Our solution will be divided into three phases. The first phase is to solve problems 1 and 2 above. The second phase is to reduce the system $vM = eb$ to an equivalent system $vA = c$ which has full column rank. The third phase is to find a particular solution of $vA = c$. The first two phases use standard methods together with the algorithms presented in previous sections. Similarly, the third phase is easy to solve with $O(n^3d^2)$ field operations, but this may be too expensive for an input system that is overdetermined (i.e. $n \gg m$) or is rank deficient. Our main contribution here is to show how to solve the third phase with only $O(nr^2d^2)$ field operations.

Phase 1: Computation of minimal denominator e

If the rank of M augmented with b is greater than the rank of M alone, then the linear system $vM = b$ does not have a rational solution. We can perform this rank check and solve problem 2 simultaneously by doing the following. Use algorithm *WeakPopovForm* to compute the non-zero rows $R \in F[x]^{r\times m}$ of a weak Popov form of M. Now use algorithm *ExtendedPopovForm* to transform the matrix

$$\left[\begin{array}{c|c} R & \\ \hline b & 1 \end{array}\right] \in F[x]^{(r+1)\times(m+1)}. \tag{1}$$

If there does not exist a row in the transformed matrix which has first m entries zero, then report that the system has no solution; otherwise, the monic associate of the last entry in this row is the desired minimal denominator e.

Phase 2: Reduction to full column rank system $vA = c$

First use algorithm *RankProfile* to compute the row and column rank profiles of M in order to identify a non-singular $r \times r$ submatrix. Now construct A from M as follows:

permute the rows and columns so that the principal $r \times r$ submatrix is non-singular, then remove the last $m - r$ columns. Let $c \in F[x]^{1\times r}$ be the corresponding subvector of eb. Any solution of $vA = c$ will be, up to permutation of entries in v, also a solution of $vM = eb$, and vice versa. Thus, we have reduced our problem to finding a particular solution $v \in F[x]^{1\times n}$ to the system $vA = c$, where A is $n \times r$ with principal $r \times r$ submatrix non-singular.

Phase 3: Particular solution of $vA = c$

Let $k = \lceil n/r \rceil$ and decompose A as

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}$$

where each A_* is $r \times r$ except for possibly A_k which has row dimension $n - (k-1)r$. Consider transforming the following augmented matrix to Hermite form:

$$\left[\begin{array}{c|c|ccc} 1 & -c & & & \\ \hline & A_1 & & & \\ & A_2 & I & & \\ & \vdots & & \ddots & \\ & A_k & & & I \end{array}\right] \longrightarrow \left[\begin{array}{c|c|ccc} 1 & & v_2 & \cdots & v_k \\ \hline & H_1 & * & \cdots & * \\ & & H_2 & \cdots & * \\ & & & \ddots & \vdots \\ & & & & H_k \end{array}\right]. \tag{2}$$

Note that the block above H_1 is necessarily zero (as shown) because $-c$ is in the lattice generated by the rows of A, that is, the system $vA = c$ has a polynomial solution for v. Once the v_* in (2) have been computed, solve the non-singular system $v_1A_1 = c - v_2A_2 - v_3A_3 - \cdots - v_kA_k$ for v_1 using the algorithm of Mulders and Storjohann (2000b). Then $v = [\, v_1 \quad v_2 \quad v_3 \quad \cdots \quad v_k \,]$ is easily seen to be a solution to the system $vA = c$.

We could apply algorithm *HermiteForm* to compute the v_* in (2) but this would cost $O(n^3d^2)$ field operations. By pipelining the computation we can avoid computation of the off-diagonal blocks $*$ and reduce the cost to $O(nr^2d^2)$. Proceed as follows.

Use algorithm *WeakPopovForm* to compute a weak Popov form R_k of A_1. For $i = k-1, k-2, \ldots, 2$ in succession, let R_i be the non-zero rows of a weak Popov form of

$$\begin{bmatrix} R_{i+1} \\ A_{i+1} \end{bmatrix},$$

computed using algorithm *WeakPopovForm*. Now compute v_i for $i = 2, 3, \ldots, k$ in succession as follows: set $c_i = -c + v_2A_2 + v_3A_3 + \cdots + v_{i-1}A_{i-1}$ and use algorithm *HermiteForm* to effect the following transformation:

$$\left[\begin{array}{c|c|c} 1 & c_i & \\ \hline & R_i & \\ & A_i & I \end{array}\right] \longrightarrow \left[\begin{array}{c|c|c} 1 & & v_i \\ \hline & * & * \\ & & H_i \end{array}\right]. \tag{3}$$

This ends the description of phase 3. We now show using induction on i that the Hermite form in (3) will be as shown, cf. (2).

For some i ($i = 2, 3, \ldots, k$) assume that $v_2, v_3, \ldots, v_{i-1}$ have been correctly computed. Note that for $i = 2$ (the base case) this assumption is vacuously true. Let $w = [\, v_2 \quad v_3 \quad \cdots \quad v_{i-1} \,]$. Write A using a conformal block decomposition as

$$A = \begin{bmatrix} A_1 \\ X \\ A_i \\ Y \end{bmatrix}.$$

Now consider the transformation to Hermite form shown in (2), but restricted to the first $ir + 1$ columns and using a sequence of unimodular transformations:

$$\left[\begin{array}{c|ccc} 1 & -c & & \\ \hline & A_1 & & \\ & X & I & \\ & A_i & & I \\ & Y & & \end{array}\right] \xrightarrow{\text{(a)}} \left[\begin{array}{c|ccc} 1 & -c & & \\ \hline & R_i & & \\ & X & I & \\ & A_i & & I \\ & & & \end{array}\right] \xrightarrow{\text{(b)}} \left[\begin{array}{c|ccc} 1 & c_i & w & \\ \hline & R_i & & \\ & X & I & \\ & A_i & & I \\ & & & \end{array}\right]$$

$$\xrightarrow{\text{(c)}} \left[\begin{array}{c|ccc} 1 & & w & v_i \\ \hline & * & & * \\ & X & I & \\ & & & H_i \\ & & & \end{array}\right] \xrightarrow{\text{(d)}} \left[\begin{array}{c|ccc} 1 & & w & v_i \\ \hline & H_1 & * & * \\ & & * & * \\ & & & H_i \\ & & & \end{array}\right].$$

Transformation (a) corresponds to the definition of R_i and involves only rows containing A_1 and Y. Indeed, R_i is the non-zero rows of a weak Popov form of A_1 augmented with Y. Transformation (b) adds $w \times \left[\,\middle|\, X \quad I\right]$ to the first row. Note that $c_i = -c + wX$. Transformation (c) is that shown in (3), and is restricted to the rows containing R_i and A_i. The key point is that the first row is already in correct form after transformation (c) completes. Thus, transformation (d), which completes the transformation to Hermite form, can be avoided.

Theorem 6.1. *Let $M \in F[x]^{n\times m}$ have rank r and degree bounded by d. Let $b \in F[x]^{1\times m}$ have degree bounded by $O(rd)$. The cost of the algorithm described above for solving the polynomial linear system $vM = b$ is bounded by $O(nmrd^2)$ field operations.*

Proof. As indicated, almost all of the computation is done by algorithms *WeakPopovForm*, *ExtendedPopovForm*, *RankProfile* and *HermiteForm*. There are a couple of places where we need to take care that these algorithms run in the allotted time.

The transformation using algorithm *ExtendedPopovForm* shown in (1) needs to be done in a special way because we are allowing $\deg b = O(rd)$. Perform the transformation in two phases. For the first phase, apply simple transformations of the first kind involving the rows of R on the last row until either the last row has degree $\leq d$ or the transformed matrix is in weak Popov form. A similar argument as used in the proof of Theorem 2.1 shows that the number of such simple transformations is bounded by $O(r \deg(b))$. To estimate the cost of the first phase it remains to bound the cost of a single simple transformation of the first kind of a row with degree bounded by d on a row with degree bounded by

$O(\deg(b))$. Before beginning, store the coefficients of the polynomials in b in m arrays of length $1 + \deg(b)$. By modifying these arrays in-place, each simple transformation can be accomplished with $O(md)$ instead of $O(m \deg(b))$ field operations. Thus, the total cost for phase one is $O(mrd \deg(b))$ field operations. For the second phase, use algorithm *ExtendedWeakPopovForm* to complete the transformation.

Next we bound the degree of c and the c_*. Note that e will be a divisor of an $r \times r$ minor of R and hence $\deg(e) \leq rd$. This shows that $\deg(c) \leq rd + \deg b$. The degree of the Hermite form shown in (2) will be bounded by $\deg(\det(A_1))$, which is $\leq rd$. Note that for $i > 2$, c_i can be computed as $c_{i-1} + v_{i-1}A_{i-1}$. Using this, we see that all the c_* can be computed from the v_* in the allotted time and will have degree bounded by $O(rd)$.

Now consider the computation of the v_* using the transformation to Hermite form shown in (3). Again, some care needs to be taken because $\deg(c_i)$ may be as large as $O(rd)$. The transformation should be done in two phases. First, use the technique described above to apply simple transformations of the rows in R_i to the first row to reduce the degree of the first row to $\leq d$. Then complete the transformation using algorithm *HermiteForm*.

For the final computation of $v_1 = A_1^{-1}(-c_k - v_k A_k)$ use the algorithm of Mulders and Storjohann (2000b). □

7. The Popov form

In this section, we show how we can transform a matrix that is in weak Popov form into Popov form. Combined with algorithm *WeakPopovForm* this will yield an algorithm to transform any matrix into Popov form.

In Kailath (1980) and Villard (1996) the Popov form of a matrix is computed via translation to problems over F with bigger dimensions. Consider the case of a non-singular $m \times m$ input matrix with degree d. Villard (1996) reduces the problem to inverting a single $m \times m$ matrix over $F[x]$ with degree d and computing the rank profiles of two $md \times md$ matrices over F. This approach also yields a fast parallel algorithm. Using the best known sequential algorithms for these problems the cost estimate becomes about $O(m^{\theta+1}d + (md)^{\theta} + m^2(md)^{1+\epsilon})$ field operations. The algorithm we propose here has cost $O(m^3d^2)$ field operations for this case.

Definition. M is said to be in ascending order if for $i < l$ we have $D_i^M < D_l^M$ or $(D_i^M = D_l^M \neq -1$ and $I_i^M < I_l^M)$.

Note that when M is in ascending order, the zero rows of M are on top, i.e. have smallest row index.

Definition (See also Kailath, 1980). M is said to be in Popov form if

1. M is in weak Popov form;
2. M is in ascending order;
3. P_i^M is monic for $i \in C^M$;
4. $\deg(m_{i,I_l^M}) < D_l^M$ for $l \in C^M$ and $i \neq l$.

When M is in weak Popov form we can transform M into ascending order by permuting the rows of M.

Assume that M already satisfies properties 1 and 2. We will make M satisfy property 4 by applying simple transformations of the second kind on M. In order that a simple transformation does not cancel progress made earlier, we apply the simple transformations in a particular order.

Suppose that the first $k-1$ rows of M already satisfy property 4, that is $\deg(m_{i,I_l^M}) < D_l^M$ for $l \in C^M$, $i \neq l$ and $i, l < k$.

If the kth row of M is the zero row, then the first k rows of M are all zero rows and satisfy property 4.

Now suppose that the kth row of M is not the zero row. For $i < k$ we then have:

1. If $D_i^M = -1$, then $\deg(m_{i,I_k^M}) = -\infty < D_k^M$.
2. If $D_i^M < D_k^M$, then $\deg(m_{i,I_k^M}) \leq D_i^M < D_k^M$.
3. If $D_i^M = D_k^M$, then $I_i^M < I_k^M$ and thus $\deg(m_{i,I_k^M}) < D_i^M = D_k^M$.

So $\deg(m_{i,I_k^M}) < D_k^M$ for $i < k$ and we only have to make the entries in row k satisfy property 4.

Let $\delta^M = \max_{i<k, i\in C^M}(\deg(m_{k,I_i^M}) - D_i^M)$. If $\delta^M < 0$, then the first k rows of M satisfy property 4. Otherwise let $l < k, l \in C^M$ such that $\delta^M = \deg(m_{k,I_l^M}) - D_l^M$ and $N = (n_{i,j})$ the matrix we get when we apply the simple transformation (of the second kind) of row l on row k. By Lemma 2.2 D_k^M and I_k^M do not change and thus N still satisfies properties 1 and 2 and still $\deg(n_{i,I_k^N}) < D_k^N$ for $i < k$.

Let $\delta^N = \max_{i<k, i\in C^N}(\deg(n_{k,I_i^N}) - D_i^N)$. If $\delta^N < 0$, the first k rows of N satisfy property 4. Otherwise, let $\nu^M = \#\{i < k | \delta^M = \deg(m_{k,I_i^M}) - D_i^M\}$ and $\nu^N = \#\{i < k | \delta^N = \deg(n_{k,I_i^N}) - D_i^N\}$. We now show that $(\delta^N, \nu^N) < (\delta^M, \nu^M)$ in the lexicographic order. For this we only have to show that $\delta^N \leq \delta^M$ and if $\delta^N = \delta^M$, then $\nu^N < \nu^M$.

For $i < k$ such that $i \neq l$ and $i \in C^N$ let $j = I_i^N = I_i^M$ and note that $D_i^N = D_i^M$. Then

$$\deg(n_{k,j}) - D_i^N \leq \max(\deg(m_{k,j}) - D_i^N, \delta^M + \deg(m_{l,j}) - D_i^N).$$

Since the first $k-1$ rows of M already satisfy property 4 we have $\deg(m_{l,j}) - D_i^N < 0$. So if $\deg(m_{k,j}) - D_i^M < \delta^M$, then $\deg(n_{k,j}) - D_i^N < \delta^M$; if $\deg(m_{k,j}) - D_i^M = \delta^M$, then $\deg(n_{k,j}) - D_i^N = \delta^M$. Moreover, $\deg(n_{k,I_l^N}) - D_l^N < \deg(m_{k,I_l^N}) - D_l^N = \delta^M$, since we applied the simple transformation of row l on row k. We see that either ($\delta^N = \delta^M$ and $\nu^N = \nu^M - 1$) or $\delta^N < \delta^M$.

Fig. 11 describes an algorithm to compute the Popov form of a matrix based on our previous observations.

Theorem 7.1. *Algorithm PopovForm is correct. The cost of algorithm PopovForm is bounded by $O(nmrd^2)$ field operations, where r is the rank of $\mathcal{M}$ and d is a bound on the degree $\mathcal{M}$.*

algorithm PopovForm
input: $\mathcal{M} \in F[x]^{n\times m}$.
output: $\mathcal{N}$ in Popov form, left equivalent to $\mathcal{M}$.
$\mathcal{A}$:= WeakPopovForm($\mathcal{M}$);
Permute rows of $\mathcal{A}$ such that $\mathcal{A}$ is in ascending order;
for k **to** n **do**
  if kth row is not the zero row **then**
    do
      Let $\delta = \max_{i<k, i\in C^{\mathcal{A}}}(\deg(m_{k,I_i^{\mathcal{A}}}) - D_i^{\mathcal{A}})$;
      if $\delta < 0$ **then**
        break
      fi;
      Let $l < k, l \in C^{\mathcal{A}}$ such that $\deg(m_{k,I_l^{\mathcal{A}}}) - D_l^{\mathcal{A}} = \delta$;
      Apply simple transformation of row l on row k
    od
  fi
od;
Multiply nonzero rows of $\mathcal{A}$ with constant to make pivots monic;
$\mathcal{N}$:=copy($\mathcal{A}$);
return $\mathcal{N}$

Fig. 11. Algorithm *PopovForm*.

Proof. Since always $\delta^M \le d$ and $\nu^M < r$ it follows from the previous observations that in the loop at most $O(rd)$ simple transformations are applied on each non-zero row. So the total number of simple transformations applied in the loop is $O(r^2 d)$. From Lemma 2.2 it follows that the degree of $\mathcal{A}$ is always bounded by d. Thus the cost of the loop is $O(r^2 m d^2)$. The theorem now follows from Theorem 2.1. □

8. Reduced basis

In von zur Gathen (1984) the notion of reduced basis is introduced. For a polynomial matrix $M = (m_{i,j}) \in F[x]^{n\times m}$ of rank r this boils down to the following.

Definition. M is said to be *reduced* if

1. Rows $r+1, \ldots, n$ are zero rows;
2. For $1 \le i \le r$ we have $\deg(m_{i,k}) < \deg(m_{i,i})$ for $1 \le k < i$ and $\deg(m_{i,k}) \le \deg(m_{i,i})$ for $i \le k \le m$;
3. $\deg(m_{i,i}) \le \deg(m_{j,j})$ for $1 \le i \le j \le r$.

In von zur Gathen (1984) and von zur Gathen and Gerhard (1999, Exercise 16.12) an algorithm is described to transform a full row rank matrix, up to column permutation, into a reduced matrix by a unimodular row transformation. The complexity of this algorithm turns out to be $O(mn^3 d^{2+\epsilon})$ field operations.

Now suppose M is already in Popov form. If $\deg(P_k^M) \leq \deg(P_l^M)$ for $k \neq l$, then $\deg(m_{l,I_k^M}) < \deg(P_k^M) \leq \deg(P_l^M)$. From this we see that by permuting the rows and columns of M such that the pivots of M end up on the diagonal with increasing degree from top to bottom, we get a reduced matrix. So we can transform any matrix in reduced form by first computing its Popov form and then permuting its rows and columns. The cost of this is $O(nmrd^2)$ by Theorem 7.1, which is one order of magnitude better than the algorithm described by von zur Gathen (1984).

Reduced basis is used by von zur Gathen (1984) to compute short vectors in modules. In the polynomial case the weak Popov form already suffices for that.

Lemma 8.1. *If M is in weak Popov form and l is such that* $\deg(P_l^M) = \min_{1\leq i\leq n}(\deg(P_i^M))$*, then all vectors in the $F[x]$-module generated by the rows of M have degree at least* $\deg(P_l^M)$.

Proof. Let $r^i \in F[x]^{1\times m}$ denote the ith row of $M = (m_{i,j})$ and let $d_i \in F[x]$ such that $r = \sum_{i=1}^n d_i r^i \neq 0$. Let k be such that $\deg(d_k P_k^M)$ is maximal and I_k^M maximal, i.e. for $i \neq k$ either $\deg(d_i P_i^M) < \deg(d_k P_k^M)$ or $\deg(d_i P_i^M) = \deg(d_k P_k^M)$ and $I_i^M < I_k^M$. Then for $i \neq k$ we have

1. if $\deg(d_i P_i^M) < \deg(d_k P_k^M)$, then $\deg(d_i m_{i,I_k^M}) \leq \deg(d_i P_i^M) < \deg(d_k P_k^M)$;
2. if $\deg(d_i P_i^M) = \deg(d_k P_k^M)$ and $I_i^M < I_k^M$, then $\deg(d_i m_{i,I_k^M}) < \deg(d_i P_i^M) = \deg(d_k P_k^M)$.

It follows that $\deg(r_{I_k^M}) = \deg(d_k P_k^M) \geq \deg(P_l^M)$. □

9. Discrete valuation rings

In this section we extend the notion of weak Popov form to the setting of discrete valuation rings.

Definition (Atiyah and MacDonald, 1969). Let K be a field. A *discrete valuation* on K is a mapping v of K^* onto $\mathbb{Z}$ such that

1. $v(ab) = v(a) + v(b)$;
2. $v(a + b) \geq \min(v(a), v(b))$.

Let R be the ring consisting of 0 and all $a \in K^*$ such that $v(a) \geq 0$. Then R is called a *discrete valuation ring*. R is a local ring and its maximal ideal $\mathcal{I}$ is the set of all $a \in K$ such that $v(a) > 0$. Let $u \in R$ such that $v(u) = 1$. Then $\mathcal{I} = (u)$, the ideal of R generated by u. The set R^* of units of R is the set of all $a \in K$ such that $v(a) = 0$. Let $S \subseteq R^* \cup \{0\}$ such that the canonical projection map $S \to R/\mathcal{I}$ is a bijection. For $a, b \in R$ with $v(a) \geq v(b)$, we have $v(u^{v(b)-v(a)}a/b) = 0$, so there exists a unique $c \in S\backslash\{0\}$ such that $u^{v(b)-v(a)}a/b - c \in \mathcal{I}$, and thus

$$v(a - cu^{v(a)-v(b)}b) > v(a). \tag{4}$$

Example 2. Let F be a field. The set $F[[x]]$ of formal power series in x is a discrete valuation ring. For $a \in F[[x]]$, $v(a)$ is the maximum $n \in \mathbb{N}$ such that x^n divides a. For S we can take F in this case.

Let $M = (m_{i,j}) \in R^{n\times m}$. As an analogue to Section 2 we define the pivot element P_i^M of row i of M as the rightmost element with minimum valuation in its row, the pivot index I_i^M as the index of P_i^M, i.e. $P_i^M = m_{i,I_i^M}$, and the pivot valuation D_i^M as $v(P_i^M)$. Again, M is said to be in weak Popov form if all (non-zero) indices are different. If $v(m_{l,I_k^M}) \geq v(P_k^M)$, let $c \in S\backslash\{0\}$ such that $v(m_{l,I_k^M} - cu^{v(m_{l,I_k^M})-v(P_k^M)} P_k^M) > v(m_{l,I_k^M})$. Then we call subtracting $cu^{v(m_{l,I_k^M})-v(P_k^M)}$ times row k from row l the simple transformation of row k on row l. The analogue of Lemma 2.2 holds also.

Lemma 9.1. *Let N be the matrix we get after applying the simple transformation of row k on row l of M. Then $I_i^N = I_i^M$, $D_i^N = D_i^M$ for $i \neq l$ and $D_l^N \geq D_l^M$. If the transformation is of the first kind, then either $D_l^N > D_l^M$ or $(D_l^N = D_l^M$ and $I_l^N < I_l^M)$. If the transformation is of the second kind, then $I_l^N = I_l^M$ and $D_l^N = D_l^M$.*

Now we can apply algorithm *WeakPopovForm* to transform M into weak Popov form. However, the algorithm may run forever as the following example shows.

Example 3. For

$$\mathcal{M} = \begin{bmatrix} \frac{x}{1-x} \\ 1 \end{bmatrix} = \begin{bmatrix} x + x^2 + x^3 + \cdots \\ 1 \end{bmatrix} \in F[[x]]^{2\times 1}$$

algorithm *WeakPopovForm* will keep on subtracting x^i from $\mathcal{M}_{1,1}$ for increasing i and thus run forever. However, it is possible to transform $\mathcal{M}$ into weak Popov form by a unimodular transformation, since

$$\begin{bmatrix} 1 & 1 \\ -x & 1-x \end{bmatrix} \begin{bmatrix} 1 \\ x + x^2 + x^3 + \cdots \end{bmatrix} = \begin{bmatrix} 1 + x + x^2 + \cdots \\ 0 \end{bmatrix}.$$

Notice that the unimodular transformation matrix is even over $F[x]$. Indeed, algorithm *WeakPopovForm* only computes transformations over $F[x]$.

Lemma 2.3 and Corollary 2.2 are still valid in the discrete valuations ring setting and thus the number of different values that a pivot index can assume during the course of algorithm *WeakPopovForm* is bounded by the rank of the matrix. The following lemma shows that the algorithm still works when $\mathcal{M}$ has full row rank.

Theorem 9.1. *Suppose $\mathcal{M}$ has full row rank. Let d be the valuation of the determinant of some non-singular $n \times n$ submatrix of $\mathcal{M}$. Then algorithm WeakPopovForm is correct and applies at most $dn + n(n-1)$ simple transformations of the first kind.*

Proof. Since the index of a row can assume at most n different values, Lemma 9.1 implies that the valuation of row l, that is $\min_{1\leq j\leq m}(v(\mathcal{M}_{l,j}))$, must have increased after applying

n simple transformations of the first kind on row l and so when s_l simple transformations of the first kind are applied on row l the valuation of that row must have increased by at least $\lfloor s_l/n \rfloor$.

Let $\mathcal{G}$ be a non-singular submatrix of $\mathcal{M}$ and $d = v(\det(\mathcal{G}))$. Suppose that algorithm *WeakPopovForm* applies more than $dn + n(n-1)$ simple transformations of the first kind and suppose $\mathcal{G}$ is transformed into $\mathcal{H}$ after applying the first $dn + n(n-1) + 1$ simple transformations. Then $v(\det(\mathcal{H})) \geq \sum_{i=1}^{n} \lfloor s_i/n \rfloor > d$, contradicting $\det(\mathcal{H}) = \det(\mathcal{G})$.

So algorithm *WeakPopovForm* does stop and is thus correct by Lemma 2.1. $\square$

As in the polynomial case, the weak Popov form in the current setting can be used to determine a vector with minimal valuation in the R-module generated by the rows of a matrix.

The analogue of Popov form would insist that $v(m_{i,I_l^M}) > D_l^M$ for $i \neq l$. It is in general not possible to transform a matrix into Popov form by only using unimodular transformations.

Example 4. Let

$$M = \begin{bmatrix} 1 & x \\ x^2 & x^2 c \end{bmatrix}.$$

Then M is non-singular and in weak Popov form. Suppose

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in F[[x]]^{2\times 2}$$

is unimodular and $N = UM$ is in weak Popov form. We may assume (eventually switch rows) that $v(a) = 0$. Then $v(N_{1,1}) = 0$, $v(N_{1,2}) = 1$ and $v(N_{2,2}) \geq 1$. So $I_1^N = 1$ and thus I_2^N must be 2. Since $v(N_{1,2}) \leq v(N_{2,2})$, N cannot be in Popov form.

10. Conclusions

We have introduced the weak Popov form of a polynomial matrix and described a simple algorithm to compute the form. The algorithm transforms a matrix by applying elementary row operations in such a way that the degrees of rows never increase. This leads to a complexity of $O(nmrd^2)$ field operations for transforming an input matrix $A \in F[x]^{n\times m}$ of rank r with entries of degree bounded by d. The algorithm is central to various other algorithms: for rank profile, determinant, Hermite form, Popov form, linear system solution and short vector computation.

The analysis in this paper only counts field operations and thus gives a good estimate of the cost when F is a finite field. The hidden constants in the big-O bounds are not explicitly computed but estimates can be derived without too much difficulty. Since these constants are small, the algorithms will perform well in practice, also for modest sized input matrices. Some comparative experiments with implementations of various algorithms in Aldor (Watt et al., 1994) confirm this.

For the problem of computing the Hermite form we did not obtain a complexity bound that was cubic in the matrix dimension for the case of an input matrix that does not have full column rank. The problem of computing the form in this case is at least as difficult as computing a unimodular transformation matrix U to achieve the form. For example, let $A \in F[x]^{n\times n}$ be non-singular with degree bounded by d. Consider transforming the $n \times 2n$ matrix $[\ A \mid I\ ]$, which is obviously not of full column rank, to Hermite form $[\ H \mid U\ ]$. The triangular factorization of A is given by $A = VH$, where $V = U^{-1}$. Note that V will have degree bounded by d but U will have degree bounded by $(n-1)d$. We have shown how to compute V and H with $O(n^3d^2)$ field operations. Can U be computed in the same time?

The performance of the algorithms for other coefficient fields F, e.g. $F = \mathbb{Q}$ (or $\mathbb{Z}$), is another issue. In this case, intermediate expression swell on the coefficient level is introduced, leading to a severe breakdown of the algorithms' performance. Combining the algorithms with homomorphic imaging schemes may be the solution to this problem. Another idea may be to introduce fraction free techniques, as is done by Beckermann et al. (1999, 2002). Further research needs to be done in this area.

We also extended the notion of weak Popov form to the setting of discrete valuation rings. Such an extension does not seem possible for the notion of Popov form. Another remaining question is how to transform in the discrete valuation ring setting a non-full row rank matrix into weak Popov form. The algorithm presented in Section 9 may run forever on such a matrix.

Acknowledgement

The work for this paper was mostly done during both authors' stay at the Institute of Scientific Computing, Department of Computer Science, ETH Zurich, Switzerland.

References

Atiyah, M.F., MacDonald, I.G., 1969. Introduction to Commutative Algebra, Addison-Wesley, New York.

Beckermann, B., Labahn, G., Villard, G., 1999. Shifted normal forms of polynomial matrices. In: Dooley, S. (Ed.), Proceedings of the Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '99, ACM Press, New York, pp. 189–196.

Beckermann, B., Labahn, G., Villard, G., 2002. Normal forms for general polynomial matrices, Research Report 2002–1, ENS Lyon, France.

Domich, P.D., Kannan, R., Trotter, L.E. Jr., 1987. Hermite normal form computation using modulo determinant arithmetic. Mathematics of Operations Research 12 (1), 50–59.

von zur Gathen, J., 1984. Hensel and Newton methods in valuation rings. Mathematics of Computation 42, 637–661.

von zur Gathen, J., Gerhard, J., 1999. Modern Computer Algebra, Cambridge University Press, Cambridge.

Kailath,T., 1980. Linear Systems, Prentice-Hall, New Jersey.

Labhalla, S., Lombardi, H., Marlin, R., 1996. Algorithmes de calcul de la réduction de Hermite d'une matrice à coefficients polynomiaux. Theoretical Computer Science 161, 69–92.

MacDuffee, C.C., 1956. The Theory of Matrices, Chelsea, New York.

Mulders, T., Storjohann, A., 2000a. Certified dense linear system solving. Techreport 355, ETH Zurich, Department of Computer Science. Journal of Symbolic Computation (to appear).

Mulders, T., Storjohann, A., 2000b. Rational solutions of singular linear systems. In: Traverso, C. (Ed.), Proceedings of the Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '00, ACM Press, New York, pp. 242–249.

Newman, M., 1972. Integral Matrices, Academic Press, New York.

Popov, V., 1969. Some properties of control systems with irreducible matrix transfer functions. In: Lecture Notes in Mathematics, vol. 144, Springer, New York.

Storjohann, A., 2002. High-order lifting. In: Mora, T. (Ed.), Proceedings of the Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '02, ACM Press, New York, pp. 246–254.

Villard, G., 1996. Computing Popov and Hermite forms of polynomial matrices. In: Lakshman, Y.N. (Ed.), Proceedings of the Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '96, ACM Press, New York, pp. 250–258.

Watt, S.M. et al., 1994. A first report on the $A^\sharp$ compiler. In: Giesbrecht, M. (Ed.), Proceedings of the Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '94, ACM Press, New York, pp. 25–31.