

Generalized Subresultants for Computing the Smith Normal Form of Polynomial Matrices

GILLES VILLARD

LMC-IMAG, 46, Av. F. Viallet, F38031 Grenoble

(Received 19 July 1995)

We describe a new algorithm for the computation of the Smith normal form of polynomial matrices. This algorithm computes the normal form and pre- and post-multipliers in deterministic polynomial time. Noticing that the computation reduces to a linear algebra problem over the field of the coefficients, we obtain a good worst-case complexity bound.

©1995 Academic Press Limited

1. Introduction

This paper establishes that pre- and post-multipliers for the Smith normal form of polynomial matrices can be computed in deterministic polynomial time. The Smith normal form is generally defined over a principal ideal domain, it is entirely computed within the domain and consists in a diagonalization of the input matrix. We will also deal with the Hermite normal form as an intermediate form: the Hermite form is a triangularization of the input matrix. Those normal forms are well known from a theoretical point of view, the reader may refer to Gantmacher (1966), but some problems remain to be solved when they have to be computed.

In the case of matrices with integer entries, as shown by Frumkin (1977) for the Hermite form and by Kannan and Bachem (1979) for the Smith form, the two normal forms can be computed in polynomial time. The diagonalization is computed using repeated triangularizations of the matrix. The bounds on the number of digits of the integers appearing during the calculations have been first improved by Chou and Collins (1982) by changing the order in which the computations were done, and then by several authors using modulo determinant arithmetic: Domich *et al.* (1987), Iliopoulos (1989), Kaminski and Paz (1986) and Schrijver (1986). More recently, asymptotically faster algorithms have been given in Hafner and Mc Curley (1991), and a rigorous study of modulo determinant methods has been developed by Labhalla *et al.* (1992) for the general case of non full rank matrices.

These methods may also be applied for polynomial matrices over $K[x]$ and bound the degrees of the polynomials involved in the calculations. But, for instance when working with ground field $K = \mathbf{Q}$, the field of the rationals, they are not sufficient to correctly bound the coefficients of these latter polynomials. The related problems are similar to

those encountered when computing a polynomial greatest common divisor using Euclid's algorithm. A first direct polynomial time method bringing a matrix into Hermite normal form with ground field \mathbf{Q} was given by Kannan (1985). Following Kannan and Bachem (1979), Kannan has also proposed to compute the Smith normal form by repeated applications of his algorithm, but it is not clear that this way, the solution would be obtained in a polynomial number of bit operations. The first polynomial time algorithm for the Smith normal form appeared in Kaltofen *et al.* (1987): for a polynomial matrix over the rationals, *SMITH FORM* (computing the normal form) is in \mathcal{P} , the class of sequential polynomial time problems. The solution is obtained via the Chinese remainder algorithm, but the process does not apply to the computation of associated multipliers of size polynomial in the input size in a deterministic manner. Multipliers (or equivalence transforms) are unimodular polynomial matrices U and V such that if S is the Smith form of A then $UAV = S$. The last breakthrough was done by Kaltofen *et al.* (1987,1990). In these papers a Monte-Carlo and a Las Vegas probabilistic algorithm are given for computing the Smith form. The authors have shown that with high probability, the cost of the computation of the Smith form is the cost of the computation of the Hermite form. The Smith form and the multipliers can be obtain in randomized polynomial time. In particular, they have established the existence of multipliers for the form, which entries are polynomially bounded in the dimensions and coefficient lengths of the input matrices. Their key idea is to "pre-condition" the input matrix by multiplying it with a certain randomly chosen constant matrix, say a "conditioning" matrix. The Hermite form of this new randomized matrix has, with high probability, the coefficients of the Smith form on its diagonal. Unfortunately "good-conditioning" matrices, *i.e.* directly leading to the Smith form without repetition of Hermite, were characterized as being matrices which entries do not form a root of a polynomial of a large degree in many variables, thus randomization was necessary. Adapting the Monte-Carlo algorithm of Kaltofen *et al.* (1987), the Las Vegas complexity of the computation of the Smith form has been improved by Storjohann (1994). Nevertheless, the problem of computing multipliers of size polynomial in a deterministic way was still an open question.

For matrices over $F[x]$, F a commutative field, we propose in Section 3 a deterministic algorithm that returns the Smith form and multipliers after a polynomial number of operations in F . As a consequence, we establish that *REDUCTION TO SMITH FORM* (computing the normal form and multipliers) over $F[x]$ is in \mathcal{P} when F is a concrete field such as \mathbf{Q} or $\text{GF}(p)$, the field with p elements. We obtain the result by explicitly computing a good-conditioning matrix. We triangularize the input matrix in such a way that at each step, the diagonal coefficient we obtain is exactly the corresponding coefficient of the Smith form.

Furthermore, the idea we use can be combined with the recent method of Labhalla *et al.* (1995) to obtain good complexity bounds. By viewing the computation of the Hermite form as the computation of a "big gcd", the authors have developed an efficient method based on generalized subresultants. They have shown that the computation of the Hermite form over $F[x]$ reduces to the computation of a row echelon form of a big matrix over F . Using their idea we will show in Section 4 that the computation of the Smith form over $F[x]$ may also be computed by triangularizing a big matrix over F .

Throughout the paper we assume the field F to be large enough ($\#F = O(n^2)$). In any case, if F is too small, we may compute over an algebraic extension having the required number of elements. This method is widely used to obtain fast algorithms over any fields, we refer for instance to Kaltofen *et al.* (1987,1990). Its drawback is

that returned multipliers are polynomials over the used algebraic extension. Another approach for small fields, since there is no coefficient growth in this case, could be to use repeated triangularizations as proposed by Kannan (1985); or to apply the reduction to the computation of the Frobenius normal form that we have given in Villard (1995).

In order to simplify the presentation, throughout the paper to introduce the new ideas, we will restrict ourselves to square non-singular input matrices. We will show in Section 4.3 that this can be done without loss of generality. Our main results are valid for singular or rectangular matrices.

2. Previous results

We recall some basic results that can be found in Gantmacher (1966) concerning the Hermite and Smith normal forms of an input matrix A of dimension n which entries are polynomials of $F[x]$, with F a commutative field, and the main algorithms for their computation. The degrees of the entries of A are bounded by d . A matrix of $F[x]^{n \times n}$ is called unimodular if its determinant is a non-zero element of F .

A non-singular square matrix H of $F[x]^{n \times n}$ is in *Hermite normal form* if it is upper-triangular, its diagonal entries are monic and in each column the entries above the diagonal entry are of lower degree.

- (i) Every non-singular matrix A of $F[x]^{n \times n}$ is left equivalent to a unique matrix H which is in Hermite normal form: $UA = H$, U unimodular.
- (ii) Let h_i^* denote the greatest common divisor of all the $i \times i$ minors formed with the first i columns of A ; the diagonal entries of the Hermite normal form H of A are $H_{1,1} = h_1^*$ and $H_{i,i} = h_i^*/h_{i-1}^*$, $i = 2, \dots, n$.

Any upper-triangular matrix left equivalent to A and with the $h_{i,i}$'s as diagonal entries will be said to be a *non-normal Hermite form* of A .

A non-singular square matrix S of $F[x]^{n \times n}$ is in *Smith normal form* if it is diagonal, its diagonal entries are monic and each divides the next.

- (iii) Every non-singular matrix A of $F[x]^{n \times n}$ is equivalent to a unique matrix S which is in Smith normal form: $UAV = S$, U and V unimodular.
- (iv) Let s_i^* denote the greatest common divisor of all the $i \times i$ minors of A ; the diagonal entries of the Smith normal form S of A are $s_1 = s_1^*$ and $s_i = s_i^*/s_{i-1}^*$, $i = 2, \dots, n$. The i -th diagonal entry s_i is called the i -th invariant factor of A .

The following standard lemma shows that, when computing the Smith normal form of A , the difficulty is essentially in computing a triangular form equivalent to A with the correct entries on its diagonal.

LEMMA 2.1. Let T be a non-singular upper-triangular matrix of $F[x]^{n \times n}$, with $T_{i,i}$ the i -th diagonal entry, $1 \leq i \leq n$. If $T_{i,i}$ is the i -th invariant factor of T then there exists a unique unimodular matrix V such that TV is in Smith normal form. In other words, the Hermite normal form of tT and the Smith normal form of T are equal.

In the following, we will say that a matrix T satisfying the conditions of lemma 2.1 is in *triangular Smith form*. We apply the lemma to focus on the computation of this

triangular form, this will give the algorithm F -TSF at Section 4. The Smith form will then be directly obtained by reducing the off-diagonal entries using column operations.

2.1. KANNAN'S ALGORITHM FOR HERMITE

The algorithm of Kannan (1985) is an elimination process to compute the Hermite normal form. It works in $n - 1$ steps, after step i the $(i + 1) \times (i + 1)$ principal minor is in Hermite form, and the matrix is denoted by $A^{(i)}$.

Algorithm KHNF

Input: $A^{(0)} \leftarrow A$, $n \times n$ matrix.

for i from 1 to $n - 1$

 Put the $(i + 1) \times (i + 1)$ principal minor in upper-triangular form.

 Reduce off-diagonal entries of the $(i + 1) \times (i + 1)$ principal minor.

Output: $H \leftarrow A^{(n-1)}$.

At step i unimodular row operations are performed on the first $i + 1$ rows only. The entries $A_{i+1,j}$, $j = 1, \dots, i$, become zero by left multiplying by the Bezout's matrices:

$$\begin{pmatrix} p & q \\ -A_{i+1,j}/r & A_{j,j}/r \end{pmatrix},$$

with $r = \gcd(A_{j,j}, A_{i+1,j})$, and p, q defined such that $r = pA_{j,j} + qA_{i+1,j}$. Then it is easy to perform unimodular row operations so that each off-diagonal entry has degree strictly lower than that of the diagonal entry in its column.

THEOREM 2.1. Kannan (1985). *Algorithm KHNF finds the Hermite normal form (and the multiplier) of a square non-singular matrix over $\mathbb{Q}[x]$ or over $GF(p)[x]$ in a polynomial number of bit operations.*

A different polynomial time algorithm for computing the Hermite form can be found in Kaltofen *et al.* (1987). From the unicity of the form, one can also deduce that the multiplier is unique: $U = HA^{-1}$. It is possible to build from A , in a polynomial number of operations, a linear system over F whose unique solution is (H, U) . This method has been developed from a parallel point of view and seems to be costly in sequential. We will use instead, the next result.

2.2. SUBRESULTANTS FOR HERMITE

We present in this section the method of Labhalla *et al.* (1995). As in Kaltofen *et al.* (1987), the Hermite form computation over $F[x]$ is reduced to a linear system solution over F , but the cost of finding the appropriate system is avoided. The main idea of the method is to generalize the use of the Sylvester matrix and of the subresultants for the computation of polynomial gcd to the computation of the Hermite form. Indeed, we know from Laidacker (1969) or Geddes *et al.* (1992) that if the Sylvester matrix is brought into row echelon form (by row operations only), then the last non-zero row gives the coefficients of the polynomial gcd.

The method begins by associating to the input matrix A a matrix $A^{(\delta)}$ with entries in F , where δ is a bound on the degrees of the entries of the multiplier U ($UA = H$). $A^{(\delta)}$ plays the role of the Sylvester matrix. If d is a bound on the degrees of the entries of A ,

we can take $\delta = (n - 1)d$. Then the Hermite form is obtained by bringing $A^{(\delta)}$ into row echelon form, using row operations only. At Section 4 we will extend this method, and compute a row echelon form with some column operations for the computation of the Smith form.

The lemma and theorem below give an incomplete insight into the results of Labhalla *et al.* (1995). This will be sufficient for our purpose, the reader will refer to the original paper for more details, especially about the proofs.

Let e_1, \dots, e_n be the canonical basis of the module $F[x]^n$. A natural basis of the F -vector space formed by the elements of $F[x]^n$ whose entries have degrees less than $d + \delta$ is provided by

$$\mathcal{B}^{(\delta)} = (x^{d+\delta}e_1, \dots, e_1, \dots, x^{d+\delta}e_n, \dots, e_n).$$

Let L_i be the row-vectors of A , $A^{(\delta)}$ is the $n(\delta + 1) \times n(d + \delta + 1)$ matrix with entries in F whose row-vectors are

$$[x^\delta L_1, \dots, x^\delta L_n, x^{\delta-1} L_1, \dots, x^{\delta-1} L_n, \dots, L_1, \dots, L_n]$$

written in the base $\mathcal{B}^{(\delta)}$.

It can be shown that by construction, the row-vectors of $A^{(\delta)}$ generate a F -vector space that contains the row vectors of the Hermite form H of A . Next lemma from Labhalla *et al.* (1995) indicates how to compute a non-normal Hermite form of A from $A^{(\delta)}$. The form computed is in fact “almost normal”: in each column the entries above the diagonal entry are of lower or equal degree.

LEMMA 2.2. *The matrix $H^{(\delta)}$ is a row echelon form of $A^{(\delta)}$ computed by row operations only. Well chosen n row vectors of $H^{(\delta)}$ are n row vectors, written in the base $\mathcal{B}^{(\delta)}$, of a non-normal Hermite form of A .*

PROOF. We just give the construction. Each block of $\Delta = (d + \delta + 1)$ columns of $A^{(\delta)}$ corresponds to a column of A . In the same way, intuitively, each block of Δ columns of $H^{(\delta)}$ will give the index of one of the target row vectors. Let these blocks be denoted by $H_i^{(\delta)}$, $1 \leq i \leq n$. Since a triangular form is computed, we need to point out the same structure in $H^{(\delta)}$. For each block $H_i^{(\delta)}$ let \mathcal{L}_i be the set of the indices of the rows whose first $i\Delta$ entries are not identically zero: the first i entries of the constructed row vector are not all equal to zero. Now the good candidate must give the row vector which first entry has the lowest possible degree. Precisely, this will be the case if the index of the chosen vector is the maximum in \mathcal{L}_i .

Let k_i be the maximum in \mathcal{L}_i , then the k_i -th row-vector of $H^{(\delta)}$ is exactly the i -th row-vector of a non-normal Hermite form of A , written in the base $\mathcal{B}^{(\delta)}$. \square

If the normal form is needed, a degree reduction of the upper-diagonal entries of the non-normal form, with respect to the diagonal ones, completes the algorithm.

THEOREM 2.2. Labhalla *et al.* (1995). *The row-vectors of the Hermite normal form of A (written in the base $\mathcal{B}^{(\delta)}$) are computed by bringing $A^{(\delta)}$ into row echelon form, using row operations only, then by reducing the degrees of the off-diagonal entries.*

The Hermite form is triangular and is computed by a Gaussian elimination. The Smith form is diagonal, we are going to compute it by elaborating a Gauss-Jordan like elimination on $A^{(\delta)}$.

2.3. RANDOMIZED ALGORITHM FOR SMITH

As found in Kannan and Bachem (1979), a usual method to compute the Smith normal form consists in iterating Hermite normal form computations on the matrix and on its transpose. The number of iterations is theoretically bounded by $O(n^2d)$ although in practice two iterations suffice. The randomized algorithm of Kaltofen *et al.* (1990) begins by pre-conditioning the input matrix A by multiplying it by a randomly chosen constant matrix C : $A' = AC$. With high probability, the diagonal entries of the Hermite T , of this new matrix A' , are the invariant factors of A . As defined above, T is in triangular Smith form. Consequently, by lemma 2.1, a second application of Hermite, on the transpose of T , gives the Smith form S of A .

Algorithm RSNF

Input: A , $n \times n$ matrix.

$C \leftarrow$ unit lower triangular matrix whose entries are chosen at random in F .

$A' \leftarrow AC$.

$T \leftarrow$ Hermite normal form of A' .

$S \leftarrow$ Hermite normal form of tT .

Output: S if it is in Smith normal form or *Failed*.

In fact, provided that F is large enough (working in an algebraic extension of F if needed), the entries of C can be chosen in a subset of F . Otherwise it may happen that such a matrix C does not exist.

THEOREM 2.3. Kaltofen *et al.* (1990). *Let A be a matrix over $Q[x]$ or over $GF(p)[x]$ of dimension n with the degrees of the entries bounded by d . There is a polynomial π in $n(n-1)/2$ variables of degree $O(n^3d)$ such that if the entries of C do not form a root of π , then the algorithm RSNF computes the Smith normal form and multipliers in polynomial time.*

The polynomial time complexity is the simple consequence of the two previous theorems on Hermite. This theorem does not provide a way to compute good pre-conditioning matrices (which entries do not form a root of π): the algorithm chooses them at random. The key point of our method in subsequent sections, is precisely to show how such matrices can be computed in a deterministic manner. This will immediately give a deterministic polynomial time algorithm to compute the Smith form.

Before concluding this overview of the previous algorithms, let us point out the seldom cited results of Kazimirski and Petrichkovich (1977) and of Petrichkovich (1993) that have been available to us in the meantime.

When equivalence of matrices over $F[x]$ is satisfied with one constant multiplier, the matrices are said to be *semiscalar equivalent*. For instance, above we have $T = UAC$ with C constant, T is semiscalar equivalent to A . In Kazimirski and Petrichkovich (1977) for fields of characteristic zero and in Petrichkovich (1993) for finite fields, it is shown (when the field is large enough) that any matrix is semiscalar equivalent to a matrix in triangular Smith form. This result, even if non fully constructive, is very similar to the property used in the previous probabilistic algorithm and thus to the method we are going to explain.

3. A new method for Smith

We have seen that the algorithms computing the Hermite form perform only row operations. The main stage of the method we propose for the Smith form also computes a triangular form, but does some simple column operations to ensure that the diagonal entries that are computed are those of the Smith form. We give the main idea of the method in this section. A corresponding algorithm, based on the generalized subresultants, is studied in next section. As said previously (lemma 2.1), if the triangular form has the correct diagonal entries, *i.e.* is in triangular Smith form, the Smith form is then directly obtained by reducing the off-diagonal entries by column operations.

DEFINITION 3.1. *Let B be an $n \times n$ non-singular matrix. A good-conditioning of B is an $(n - 1)$ -tuple $(\alpha_2, \alpha_3, \dots, \alpha_n)$ of F^{n-1} such that if the first column of B is replaced by the linear combination of the other columns given by:*

$$G_1 \leftarrow B_1 + \alpha_2 B_2 + \alpha_3 B_3 + \dots + \alpha_n B_n,$$

then

$$\gcd_{1 \leq k \leq n}(G_{k,1}) = \gcd_{1 \leq k, l \leq n}(B_{k,l}).$$

The gcd of the entries in the first column of B is now equal to the gcd of all the entries in B .

A triangular Smith form is computed by an elimination process in $n - 1$ steps. After step i , in each column j , $j \leq i$, the entries under the diagonal entry are zero and the diagonal entries in the i first rows are the s_j , $j \leq i$, the diagonal entries of the Smith form:

$$A^{(i)} = \begin{pmatrix} s_1 & * & * & * & * & * \\ 0 & \ddots & * & * & * & * \\ 0 & \dots & s_i & * & * & * \\ 0 & \dots & 0 & & & \\ 0 & \dots & 0 & & \bar{A}^{(i+1)} & \\ 0 & \dots & 0 & & & \end{pmatrix}. \tag{3.1}$$

Let $A_j^{(i)}$ be the j -th column-vector of $A^{(i)}$, and let $\bar{A}^{(i+1)}$ denote the submatrix formed by the $n - i$ last rows and columns of $A^{(i)}$. We compute the triangular matrix $A^{(n-1)}$ as follows.

Algorithm $F[x]$ -TSF (Triangular Smith form)

Input: $A^{(0)} \leftarrow A$, $n \times n$ non-singular matrix.

for i from 1 to $n - 1$

Compute a good-conditioning of $\bar{A}^{(i)}$: $(\alpha_{i+1}^{(i)}, \dots, \alpha_n^{(i)}) \in F^{n-i}$.

$G_i^{(i)} \leftarrow A_i^{(i-1)} + \alpha_{i+1}^{(i)} A_{i+1}^{(i-1)} + \dots + \alpha_n^{(i)} A_n^{(i-1)}$.

[Now $\gcd_{i \leq k \leq n}(G_{k,i}^{(i)}) = \gcd_{i \leq k, l \leq n}(A_{k,l}^{(i-1)})$.]

Zero the sub-diagonal entries in column i using unimodular transformations on the last $n - i + 1$ rows.

Output: $T \leftarrow A^{(n-1)}$.

The temporary value of the matrix after the good-conditioning at step i is denoted by $G^{(i)}$. Let us notice that unlike in the Hermite form, the off-diagonal entries in the triangular Smith form need not to be of lower degree than the diagonal ones, thus no

such reduction is done here. Provided that a good-conditioning can always be found, we prove with the proposition below that the algorithm is correct, *i.e.* that it computes a triangular Smith form. Then we will show how to compute such good transformations.

PROPOSITION 3.1. *For a non-singular input matrix A of $F[x]^{n \times n}$, the algorithm $F[x]$ -TSF computes, by unimodular transformations, a matrix T in triangular Smith form.*

PROOF. Clearly, the transformations are unimodular. We prove by induction that at the end of each step i , $A^{(i)}$ has the form (3.1).

At step $i = 1$, the algorithm $F[x]$ -TSF computes $A_{1,1}^{(1)} = \gcd_{1 \leq k \leq n}(G_{k,1}^{(1)})$. Indeed, when the sub-diagonal entries are zeroed in the first column using unimodular transformations, the diagonal entry is replaced by the gcd of the entries in the column. Since unimodular row transformations are used, the gcd of the entries in the column remains unchanged, this gcd is $A_{1,1}^{(1)}$ after the transformations. Now, from the definition of a good-conditioning, $\gcd_{1 \leq k \leq n}(G_{k,1}^{(1)}) = \gcd_{1 \leq k,l \leq n}(A_{k,l}^{(0)})$ thus $A_{1,1}^{(1)} = s_1$. We proceed by induction for $1 < i < n$. At step i , denoting $G_{i,i}^{(i)}$ by g_i , the algorithm $F[x]$ -TSF computes:

$$A^{(i)} = \begin{pmatrix} s_1 & * & * & * & * & * & * \\ 0 & \ddots & * & * & * & * & * \\ 0 & \dots & s_{i-1} & * & * & * & * \\ 0 & \dots & \dots & g_i & * & * & * \\ 0 & \dots & \dots & 0 & & & \\ 0 & \dots & \dots & 0 & \bar{A}^{(i+1)} & & \\ 0 & \dots & \dots & 0 & & & \end{pmatrix}.$$

Since a good-conditioning has been used ($\bar{A}^{(i)}$ is non-singular), $g_i = \gcd_{i \leq k,l \leq n}(A_{k,l}^{(i-1)})$ for the same reasons as above. We have to prove that $g_i = s_i$. From the definition, $s_1 s_2 \dots s_{i-1}$ divides all $(i-1) \times (i-1)$ minors and by construction, g_i divides all the entries in the last $n-i+1$ rows and columns. Now, all $i \times i$ minors can be computed from entries in these latter rows and columns multiplied by $(i-1) \times (i-1)$ minors: they are multiple of $s_1 s_2 \dots s_{i-1} g_i$. Since $s_1 s_2 \dots s_{i-1} g_i$ is a minor, it is the gcd of all $i \times i$ minors, and $g_i = s_i$. \square

We now turn to the problem of finding the good-conditionings. We give an algorithm to compute them, the associated proposition studies its cost.

Before, let us partly re-formulate a lemma of Kaltofen *et al.* (1990). This lemma gives a construction for a good-conditioning in the case of a $n \times 2$ matrix. We also refer to lemma 2 of Petrichkovich (1993).

LEMMA 3.1. *Let B be a $n \times 2$ matrix of rank 2 in $F[x]$ with the degree of the entries bounded by d . There exists a "test polynomial" π in $F[\alpha]$, of degree at most $2d$, such that for any α_2 that is not a root of π ,*

$$\gcd_{1 \leq i \leq n}(B_{i,1} + \alpha_2 B_{i,2}) = \gcd_{1 \leq i \leq n}(B_{i,1}, B_{i,2}).$$

In other words, α_2 is a good-conditioning for B .

PROOF. B is a matrix in $F[x]^{n \times 2}$ of rank 2. From Kaltofen, Krishnamoorthy and Saunders (1990), we know that there is a polynomial π in $F[\alpha]$, of degree at most $2d$, such that if:

- R in $F^{2 \times 2}$ is unit lower triangular,
- H is the row equivalent echelon form of BR ,
- s_1 is the first invariant factor of B ,

then $s_1 = H_{1,1}$ unless the entry below the diagonal in R is a root of π . Clearly, π is the desired polynomial:

$$BR = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \\ \dots & \dots \\ B_{n,1} & B_{n,2} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \alpha_2 & 1 \end{pmatrix} \\ = \begin{pmatrix} B_{1,1} + \alpha_2 B_{1,2} & B_{1,2} \\ B_{2,1} + \alpha_2 B_{2,2} & B_{2,2} \\ \dots & \dots \\ B_{n,1} + \alpha_2 B_{n,2} & B_{n,2} \end{pmatrix},$$

since H is obtained from BR by unimodular row operations,

$$H_{1,1} = \gcd_{1 \leq i \leq n} (B_{i,1} + \alpha_2 B_{i,2}),$$

and by definition, the first invariant factor of B is the gcd of all the entries of B :

$$s_1 = \gcd_{1 \leq i \leq n} (B_{i,1}, B_{i,2}),$$

the assertion of the lemma is simply the previously obtained identity $s_1 = H_{1,1}$. \square

For a square input matrix B of dimension n , the algorithm GC computes a good-conditioning in $n - 1$ steps. Let B_l be the l -th column-vector of B . At step j only B_1 and B_j are involved, throughout the algorithm only B_1 is modified.

Algorithm GC (Good conditioning)

Note: we assume $\#F > 2d$.

Input: B , $n \times n$ non-singular matrix.

$f \leftarrow \{f_1, f_2, \dots, f_{2d+1}\}$ a subset of F with $2d + 1$ elements

for j from 2 to n

$$g(x) \leftarrow \gcd_{1 \leq i \leq n} (B_{i,1}, B_{i,j})$$

$$\tilde{g}(x) \leftarrow \gcd_{1 \leq i \leq n} (B_{i,1})$$

$$k \leftarrow 0$$

while $\tilde{g}(x) \neq g(x)$

$$k \leftarrow k + 1$$

$$\tilde{g}(x) \leftarrow \gcd_{1 \leq i \leq n} (B_1 + f_k B_j)$$

$$\alpha_j \leftarrow f_k$$

$$B_1 \leftarrow B_1 + \alpha_j B_j$$

Output: $(\alpha_2, \alpha_3, \dots, \alpha_n)$.

PROPOSITION 3.2. *Let B be a non-singular square matrix of dimension n ($n > 1$) over $F[x]$ with the degree of the entries bounded by d ; we assume that the field F has at least $2d + 1$ elements. With B in input, the algorithm GC finds a good-conditioning $(\alpha_2, \alpha_3, \dots, \alpha_n)$ in a polynomial number of operations in F : in at most $O(nd)$ computations of greatest common divisors of n polynomials. Over the rationals this good-conditioning can be computed such that $|\alpha_j| \leq d$ for all j . If $\#F \leq 2d$, the good-conditioning is computed over an algebraic extension of F having the required number of elements.*

PROOF. Algorithm GC works by applying lemma 3.1 iteratively with the first and the $n - 1$ other columns of B . We begin by showing that the algorithm terminates after $O(nd)$

iterations of the *while loop*. The lemma may be applied: since the matrix is non-singular, the first column associated with another column j forms a $n \times 2$ matrix of rank 2. From the lemma, we may associate to each column j a test polynomial of degree at most $2d$: with at most $2d$ roots. Each time the algorithm passes through the *while loop* a new value f_k is tested. Consequently, after at most $2d + 1$ tests, the treatment of column j is finished. With j varying from 2 to n we obtain that the algorithm passes through the *while loop* at most $(n - 1)(2d + 1)$ times.

Now we show that a good-conditioning is computed. Recall that only B_1 is modified. The column-vector B_1 during step j will be denoted by $B_1^{(j)}$ and its entries by $B_{i,1}^{(j)}$. We have the relation: $B_{i,1}^{(j)} = B_{i,1}^{(j-1)} + \alpha_{j-1}B_{i,j-1}$. At the end of step $j = 2$, α_2 is such that

$$\gcd_i(B_{i,1}^{(2)}) = \gcd_i(B_{i,1} + \alpha_2 B_{i,2}) = \gcd_i(B_{i,1}, B_{i,2}).$$

We proceed by induction for $2 < j \leq n$. At the end of step j , α_j is computed such that,

$$\gcd_i(B_{i,1}^{(j)}) = \gcd_i(B_{i,1}^{(j-1)} + \alpha_j B_{i,j}) = \gcd_i(B_{i,1}^{(j-1)}, B_{i,j}).$$

Now, from step $j - 1$ we know that,

$$\gcd_i(B_{i,1}^{(j-1)}) = \gcd_i(B_{i,1}^{(j-2)} + \alpha_{j-1}B_{i,j-1}) = \gcd_i(B_{i,1}^{(j-2)}, B_{i,j-1}),$$

with the previous identity, this gives,

$$\gcd_i(B_{i,1}^{(j)}) = \gcd_i(B_{i,1}^{(j-2)} + \alpha_{j-1}B_{i,j-1} + \alpha_j B_{i,j}) = \gcd_i(B_{i,1}^{(j-2)}, B_{i,j-1}, B_{i,j}).$$

Applying the same reasoning from $j - 2$ to $j = 2$ finally leads to the expected result,

$$\gcd_i(B_{i,1}^{(j)}) = \gcd_i(B_{i,1} + \alpha_2 B_{i,2} + \dots + \alpha_j B_{i,j}) = \gcd_i(B_{i,1}, B_{i,2}, \dots, B_{i,j}).$$

From there, the cost of the whole process is easily shown to be polynomial in the dimension and the degree of B . The greatest common divisors are computed on polynomials that are constant linear combinations of the input ones. If F is the field of the rationals, we can choose every α_j between $-d$ and d : $|\alpha_i| \leq d, 2 \leq j \leq n$. If $\#F \leq 2d$, we can construct an algebraic extension of F having the required number and take the f_k 's in this extension. The corresponding cost will be studied at Section 4.2. \square

REMARK 3.1. Algorithm $F[x]$ -TSF associates to A a matrix C in $F[x]^{n \times n}$,

$$C = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ \alpha_2^{(1)} & 1 & 0 & \dots & 0 \\ \alpha_3^{(1)} & \alpha_3^{(2)} & 1 & \dots & 0 \\ \dots & \dots & \dots & \ddots & \dots \\ \alpha_n^{(1)} & \alpha_n^{(2)} & \dots & \alpha_n^{(n-1)} & 1 \end{pmatrix},$$

such that the Hermite form of $A' = AC$ is a triangular Smith form of A .

To compute a good-conditioning in polynomial time is clearly not sufficient to prove that the algorithm $F[x]$ -TSF itself is polynomial. This latter result could certainly be obtained by reasoning as done in Kannan (1985): by performing a direct elimination in $F[x]$. We prefer to combine the results of this section with the use of generalized subresultants that we have presented in section 2.2 This will lead to more satisfying complexity bounds.

4. Subresultants for Smith

To reduce a problem over $F[x]$ to a problem over F is of main interest both from a theoretical and from a practical point of view. We have seen how this can be done for the

computation of the Hermite form in Section 2.2. The problem of limiting the coefficient growth when computing gcd's over $F[x]$ is solved: the normal form is computed by a Gaussian elimination process. Reducing the computation to some linear algebra over F will lead to much simpler bounds both for the coefficients of the polynomials and for the global complexity. For the triangularization of a matrix over a field we refer to Bareiss (1972) and to Cabay and Lam (1977). All the results we will use about the Hermite normal form come from Labhalla *et al.* (1995).

In this section we show that computing the Smith form also reduces to the computation of a row echelon form of a matrix over F . We first give the algorithm. A detailed cost study will be done in Section 4.2.

4.1. THE NEW ALGORITHM

The algorithm of Section 2.2 can be extended to compute the Smith form: it suffices to introduce some block-column operations that correspond to good-conditionings. As previously, without loss of generality, we focus on the computation of a triangular matrix which has the same diagonal entries as the Smith form.

First of all, we have to determine the dimension of the F -vector space we need. In other words, we have to find the dimension of the big matrix that we associate to the input matrix A . This dimension is given by the following lemma: we show that proposition 4 of Labhalla *et al.* (1995) can be used for the Smith form. We keep the same notation as in section 2.2. In particular, the vectors of $F[x]^n$ whose entries have degrees less than a fixed degree, will be also viewed as belonging to a F -vector space. Let C be the constant matrix constructed by the algorithm $F[x]$ -TSF, and let L'_i be the row-vectors of $A' = AC$. Our target matrix, a triangular Smith form T of A , is going to be obtained as a non-normal Hermite form of A' .

LEMMA 4.1. *If $\delta = (n - 1)d$, where d is a bound on the degrees of the entries of A , the vectors $x^j L'_i$, $1 \leq j \leq \delta$, generate a F -vector space that contains the row-vectors of a triangular Smith form of A .*

PROOF. Let U be the multiplier for the Hermite form H' of A' : $UA' = UAC = H'$, H' is a triangular Smith form of A . Let A'^* be the adjoint matrix of A' , we have, $\det(A')U = A'^*H'$. For any matrix M , we denote by $\deg(M)$ the maximum degree of the entries of M . The previous identity gives:

$$\deg(U) \leq \deg(A'^*) + \deg(H') - \deg(\det(A')).$$

Since

$$\deg(H') \leq \deg(\det(A)) = \deg(\det(A'))$$

and

$$\deg(A'^*) \leq (n - 1)d$$

we get,

$$\deg(U) \leq (n - 1)d = \delta,$$

and the assertion of the lemma follows. \square

By applying the lemma, we know that it is sufficient to work in the F -vector space formed by the elements of $F[x]^n$ whose entries have degrees less than $d + \delta$. As in Section 2.2 we use the canonical basis

$$\mathcal{B}^{(\delta)} = (x^{d+\delta}e_1, \dots, e_1, \dots, x^{d+\delta}e_n, \dots, e_n).$$

Let L_i be the row-vectors of A . To the matrix A we associate the matrix $A^{(\delta)}$ whose row-vectors are the

$$[x^\delta L_1, \dots, x^\delta L_n, x^{\delta-1} L_1, \dots, x^{\delta-1} L_n, \dots, L_1, \dots, L_n]$$

written in the base $\mathcal{B}^{(\delta)}$. We know by lemma 2.2 that a non-normal Hermite form of A is obtained by bringing $A^{(\delta)}$ into row echelon form.

REMARK 4.1. Let $\Delta = d + \delta + 1$. Each block of Δ consecutive columns in $A^{(\delta)}$ stands for a column-vector in $F[x]^{n(\delta+1)}$. $A^{(\delta)}$ consists in n consecutive blocks of Δ columns. For instance adding α times a block Bck_j to a block Bck_i consists in adding, for all k , $1 \leq k \leq \Delta$, α times the k -th column of Bck_j to the k -th column of Bck_i . Conversely, to each block-column one may associate in a natural way a vector of $F[x]^{n(\delta+1)}$ (the entries of the matrix give the coefficients of the polynomials), a block-column operation therefore corresponds to the same operation in $F[x]^{n(\delta+1)}$.

In this new framework, we rewrite here the algorithm $F[x]$ -TSF: we compute a row echelon form of $A^{(\delta)}$ in $n - 1$ steps by a usual Gaussian elimination, following Bareiss (1972) for instance. The matrix after step i is denoted by $A^{(\delta,i)}$. The $n - 1$ steps correspond to a block-triangularization: at step i the sub-matrix consisting of the i first block-columns of $A^{(\delta,i)}$ is in row echelon form.

At each step i a good-conditioning is computed; the associated operations that were column operations over $F[x]$ are now block-column operations over F applied on $A^{(\delta,i-1)}$ to obtain $A^{(\delta,i)}$. In the following, let $A_j^{(\delta,i)}$ be the j -th block-column of $A^{(\delta,i)}$. And let $\bar{A}^{(\delta,i+1)}$ denote the submatrix formed by the $(n - i)(\delta + 1)$ last rows and $n - i$ last block-columns of $A^{(\delta,i)}$.

Algorithm F -TSF (Triangular Smith form)

Input: $A^{(\delta,0)} \leftarrow A^{(\delta)}$.

[The $n - 1$ steps of the Gaussian block-elimination.]

for i from 1 to $n - 1$

 Compute a good-conditioning of $\bar{A}^{(\delta,i)}$: $(\alpha_{i+1}^{(i)}, \dots, \alpha_n^{(i)}) \in F^{n-i}$.

 [Block-column operations]

$$G_i^{(\delta,i)} \leftarrow A_i^{(\delta,i-1)} + \alpha_{i+1}^{(i)} A_{i+1}^{(\delta,i-1)} + \dots + \alpha_n^{(i)} A_n^{(\delta,i-1)}.$$

 [Usual operations of Gaussian elimination over F]

 [Zero the sub-diagonal entries in block-column i]

$$A^{(\delta,i)} \leftarrow \text{for } k \text{ from } (i - 1)(d + \delta + 1) + 1 \text{ to } i(d + \delta + 1)$$

 Elimination of the sub-diagonal entries in column k .

Output: $T^{(\delta)} \leftarrow A^{(\delta,n-1)}$.

The temporary matrix after the good-conditioning at step i is denoted by $G_i^{(\delta,i)}$. The proposition and the corollary below lead to our main result. They establish that the algorithm F -TSF outputs the correct triangular Smith form and deduce the Smith form.

A detailed cost study will be done in the next section. First let us look at the behaviour of the algorithm on a basic example.

EXAMPLE 4.1. Let $A = \begin{pmatrix} x - 1 & 3x + 2 \\ x - 1 & 2x + 3 \end{pmatrix}$. Taking $\delta = 1$, $A^{(\delta)}$ is

$$\begin{pmatrix} 1 & -1 & 0 & 3 & 2 & 0 \\ 1 & -1 & 0 & 2 & 3 & 0 \\ 0 & 1 & -1 & 0 & 3 & 2 \\ 0 & 1 & -1 & 0 & 2 & 3 \end{pmatrix}.$$

Rows 2 and 4 of the row echelon form of $A^{(\delta)}$ (following the fraction free elimination of Bareiss (1972) without column operations) would lead to a non-normal Hermite form of A :

$$\begin{pmatrix} 1 & -1 & 0 & 3 & 2 & 0 \\ 0 & 1 & -1 & 0 & 3 & 2 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \equiv \begin{pmatrix} x-1 & 3x+2 \\ 0 & x-1 \end{pmatrix}.$$

This is not a triangular Smith form of A . But if we first add the second block of three columns ($\delta = 1, d = 1$) to the first block (thus taking $\alpha_2 = 1$),

$$\begin{pmatrix} 4 & 1 & 0 & 3 & 2 & 0 \\ 3 & 2 & 0 & 2 & 3 & 0 \\ 0 & 4 & 1 & 0 & 3 & 2 \\ 0 & 3 & 2 & 0 & 2 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 4 & 1 & 0 & 3 & 2 & 0 \\ 0 & 5 & 0 & -1 & 6 & 0 \\ 0 & 0 & 5 & 4 & -9 & 10 \\ 0 & 0 & 0 & -5 & 10 & -5 \end{pmatrix},$$

rows 3 and 4 of the row echelon form of the new matrix give

$$\begin{pmatrix} 5 & 4x^2 - 9x + 10 \\ 0 & -5x^2 + 10x - 5 \end{pmatrix}, \text{ or } \begin{pmatrix} 1 & 4/5x^2 - 9/5x + 2 \\ 0 & x^2 - 2x + 1 \end{pmatrix},$$

which is in triangular Smith form. The upper-diagonal entry may be zeroed by a column operation to give the Smith normal form.

PROPOSITION 4.1. *For a non-singular input matrix A of degree d in $F[x]^{n \times n}$, F is a field with at least $2nd + 1$ elements, the algorithm F -TSF computes, in a polynomial number of operations in F , a triangular matrix T in $F[x]^{n \times n}$ whose diagonal entries are the entries of the Smith form of A , a unimodular matrix U in $F[x]^{n \times n}$ and an invertible constant matrix C in $F^{n \times n}$ such that $T = UAC$. If $\#F \leq 2nd$, the constant matrix C is computed over an algebraic extension of F having the required number of elements.*

PROOF. Each block-column of $A^{(\delta)}$ has Δ columns: $\Delta = d + \delta + 1$. Using blocks of Δ columns in $A^{(\delta)}$ and in its row echelon form $H^{(\delta)}$ (obtained with row operations only), we have seen, at lemma 2.2, how a non-normal Hermite form of A is deduced from $H^{(\delta)}$.

We now turn to the computation of the triangular Smith form. Computing a non-normal Hermite form of the conditioned matrix $A' = AC$ will give a triangular Smith form of A . By extension of definition 3.1 in the previous section we define a good-conditioning of $\bar{A}^{(\delta,i)}$. In a natural way (remark 4.1) we associate to $\bar{A}^{(\delta,i)}$ (which is written in the base $\mathcal{B}^{(\delta)}$) a matrix $\bar{A}^{(i)}$ over $F[x]$. A good-conditioning for $\bar{A}^{(\delta,i)}$ is defined to be a good-conditioning for $\bar{A}^{(i)}$, with the associated block-column operations.

We look at what happens at the first step $i = 1$ of the algorithm. The proposition is obtained by applying the same reasoning to all steps $i, 1 < i < n$. At any time during the first step, to each block-column we associate a vector in $F[x]^{n(\delta+1)}$ (remark 4.1). The Gaussian elimination on the first Δ columns computes the gcd of the polynomial entries in the first column-vector in $F[x]^{n(\delta+1)}$. Since a block-column operation stands exactly for the same operation over $F[x]^{n(\delta+1)}$, once the good-conditioning is applied

the Gaussian elimination on the first Δ columns computes the gcd of all the polynomial entries of the matrix over $F[x]$. In the same way, at each step i , the Gaussian elimination on the first Δ columns of $\tilde{A}^{(\delta,i)}$ computes the gcd of all the polynomial entries of $\tilde{A}^{(i)}$. Finally, we know from the proof of the algorithm $F[x]$ -TSF (proposition 3.1), that those gcd's are the diagonal entries of the Smith form. When the row echelon form is obtained, the triangular Smith form is deduced as above by considering the sets \mathcal{L}_i of the indices of the rows whose first $i\Delta$ entries are not identically zero.

Concerning the multipliers U and C such that $UAC = T$: we have seen that the entries of C are the good-conditionings (remark 3.1), U is computed using a bordering identity matrix. At the beginning of the elimination, $A^{(\delta)}$ is augmented on the right by a $n(\delta + 1) \times n(\delta + 1)$ identity matrix to record the row transformations. At the end, as done for T , the matrix U is obtained by considering the rows which indices are in \mathcal{L}_i .

It remains to see that the triangular form is computed using a polynomial number of operations. The number of operations is bounded by the Gaussian elimination and by proposition 3.2. Indeed, from lemma 4.1 the degrees are implicitly bounded by $\Delta - 1$, so each step i computes at most $(n - i)(\Delta - 1)$ gcd's ($(n - i)$ block-columns are involved) of at most $(n - i + 1)(\delta + 1)$ polynomials (number of rows of $\tilde{A}^{(\delta,i)}$).

The good-conditionings are computed with polynomials of degree at most $\Delta - 1 = nd$, by proposition 3.2, F must have at least $2\Delta + 1 = 2nd + 1$ elements. Otherwise an algebraic extension has to be constructed to provide enough candidates for the entries of C . \square

COROLLARY 4.1. *The Smith normal form of a non-singular matrix A of degree d in $F[x]^{n \times n}$ can be computed using a polynomial number of operations in F .*

PROOF. By proposition 4.1 we compute a triangular Smith form T of A , its entries are of degree at most $\Delta - 1$. As seen at lemma 2.2, in each column the entries above the diagonal entry are of lower or equal degree. We now construct the matrix V of lemma 2.1 to get the Smith form. We may use a procedure similar to the *reduce-off-diagonal* procedure of section 3.1 in Kannan (1985). This procedure is used to ensure that the off-diagonal entries of the Hermite form have degree strictly less than that of the diagonal entries. In our case, we know that it will result in a diagonal matrix, the Smith form. Indeed, since V exists, each off-diagonal entry in T is a polynomial multiple of the diagonal entry in its row.

We may first add a suitable polynomial multiple of column $n - 1$ to the column n so that the $(n - 1, n)$ entry is zeroed. Let d_n and d_{n-1} be the degrees of the invariant factors s_n and s_{n-1} . Since in each column the entries above the diagonal entry are of lower or equal degree, the new column n is obtained by adding to the old one the column $n - 1$ multiplied by a polynomial of degree at most $d_n - d_{n-1}$. The new entries in column n are the old ones plus polynomials of degrees at most $(d_n - d_{n-1}) + d_{n-1} \leq d_n$. Thus the property on the degrees in the column remains true. Next we add a suitable multiple of column $n - 2$ to the column n to zero the $(n - 2, n)$ entry. As previously, we add to the column n the column $n - 2$ multiplied by a polynomial of degree at most $d_n - d_{n-2}$. The degrees in column n remains bounded by d_n . Next we add suitable multiples of columns $n - 3, \dots, 1$ to the column n so that all its off-diagonal entries are zeroed. We repeat this process with columns $n - 1, n - 2, \dots, 2$ and obtain the Smith form. The matrix V built this way is upper-triangular with diagonal unity. If d_j is the degree of the

j -th invariant factor, the (i, j) entry, $1 \leq i < j \leq n$, is a polynomial of degree $d_j - d_i$. Clearly this algorithm is polynomial time bounded. \square

4.2. COST STUDY

We have seen in the corollary above that the Smith form is computed using a polynomial number of operations in the ground field. We now precisely estimate this number. Further, we prove that over the rationals, the number of bit operations is polynomially bounded in the dimension, the degree and the coefficient lengths of the input matrix. We are going to use soft O notations: $\tilde{O}(f(n)) = O(f(n)) \log^{O(1)}(n)$.

THEOREM 4.1. *Let REDUCTION TO SMITH FORM be the problem of computing the Smith normal form and associated multipliers for a matrix of dimensions and degree bounded by $O(n)$. Using asymptotically fast arithmetic, for a non-singular square matrix over $F[x]$, the problem can be solved with $\tilde{O}(n^9)$ operations in F . Over $F = \mathbf{Q}$, the field of the rationals, if the coefficients of the input polynomials have numerators and denominators bounded in magnitude by $2^{O(n)}$, then the problem is in \mathcal{P} , it can be solved with $\tilde{O}(n^{17})$ bit operations using standard arithmetic.*

PROOF. We first recall the cost for computing a gcd of many polynomials. Next we derive the bound for computing the form over F and we will terminate by measuring the coefficient growth over \mathbf{Q} . A function $C^*(n)$ will denote a cost in terms of operations in F with fast arithmetic. Without the star, $C(n)$ will denote a cost over the rationals using standard arithmetic. When $F = GF(p)$ is too small for computing a good-conditioning, for any given l we can construct an algebraic extension \tilde{F} of F having l elements. This can be done by computing an irreducible polynomial of degree $O(\log l)$ in $F[x]$ using the algorithm of Shoup (1993). Then the arithmetic operations in \tilde{F} can be done with $O(P(\log l) \log \log l)$ operations in F . [If the product of two polynomials of degree l can be computed with $P(l)$ operations in F .] Since in the following, the required number l of elements in F will be a polynomial in n , this will never increase the costs by a factor more than poly-logarithmical in n and this will be omitted.

We know from Aho, Hopcroft and Ullman (1974) that kl gcd's of kl polynomials of degree l can be computed with

$$G^*(k, l) = O(k^2 l^2 P(l) \log l) = \tilde{O}(k^2 l^3) \quad (4.1)$$

operations in F . Over the rationals, with input coefficients bounded in magnitude by b , the computation can be done via the Chinese remainder algorithm, we apply for that the results and the costs given in Aho, Hopcroft and Ullman (1974). We know from Loos (1982), that we can compute the gcd's using $O(l \log(lb)/q)$ primes of length $q = O(\log l + \log \log b)$, the number of "bad primes" being also bounded by $O(l \log(lb)/q)$. Using standard arithmetic, a gcd of kl polynomials of degree l modulo one prime is computed in time $O(kl \times l^2 q^2)$. One application of the Chinese remaindering for one number costs $O(l^2 \log^2(lb))$. Thus the cost of the computation of the residues of the kl gcd's dominates, the overall cost is

$$GC(k, l, b) = O(kl \times kl^3 q^2 \times l \log(lb)) = \tilde{O}(k^2 l^5 \log(b)) \quad (4.2)$$

bit operations.

We now look at the cost of the computation of the normal form and of multipliers. If d is the input degree, $\Delta = d + \delta + 1$, with $\delta = (n - 1)d$, bounds all degrees during the computation. As seen during the proof of proposition 4.1, each step i of the algorithm F -TSF, $1 \leq i \leq n$, firstly consists of the computation of $O((n - i)\Delta)$ gcd's of $O((n - i)\delta)$ polynomials of degree Δ . This is the computation of a good-conditioning. From (4.1) with $k = n - i$ and $l = O(nd)$ this can be done with $\tilde{O}((n - i)^2 n^3 d^3)$ operations in F . Secondly we have to perform Δ steps of triangularization of a $(n - i + 1)(\delta + 1) \times (n - i + 1)\Delta$ matrix. This is done in time $\tilde{O}((n - i)^2 n^3 d^3)$. If the left multiplier U is computed, transformations are applied to a bordering matrix with kl columns. This is done within the same cost. Summing over the n steps, this gives $\tilde{O}(n^6 d^3)$ operations. After these two phases, the triangular Smith form is known. It remains to bring the matrix into diagonal form, this is of lower cost. As seen at lemma 4.1, the upper-diagonal entries are zeroed by exact divisions of polynomials. The associated right transformation matrix is computed as the product of C by a triangular matrix of degree Δ : in time $O(n^3 P(nd)) = \tilde{O}(n^4 d)$. Taking $d = O(n)$, the Smith form and associated multipliers can be computed with $S^*(n) = \tilde{O}(n^6 \times n^3) = \tilde{O}(n^9)$ operations in F .

We now assume that the coefficients of the input polynomials are rationals with numerators and denominators bounded in magnitude by β . As given by proposition 3.2, the entries of C , the matrix of the good-conditionings, are bounded in magnitude by Δ . The triangular Smith form is computed by triangularizing the big matrix $A^{(\delta)}$ associated to $A' = AC$. The entries of A' are bounded in magnitude by $\beta + (n - 1)\Delta\beta = O(n^2 d\beta)$, its dimensions are $O(n^2 d)$. At step i of the elimination, we take $k = n - i$ and $l = O(nd)$ in (4.2). The lengths of the numerators and denominators of the coefficients of the polynomials are bounded by $(n^2 d\beta)^{O(ind)}$. Identity (4.2) clearly bounds the overall cost at step i by $\tilde{O}((n - i)^2 \times n^5 d^5 \times ind \log(nd\beta))$. Summing over the n steps we get that the triangular Smith form and U are computed with $\tilde{O}(n^{10} d^6 \log(nd\beta))$ bit operations. As previously, the final computation of the diagonal form is of lower cost. Indeed, the column j of the right multiplier is computed in $j - 1$ steps. At each step we compute an exact division by an invariant factor and multiplications involving the corresponding quotient and entries of the triangular Smith form. It is easily seen the sizes the numbers are multiplied by n , thus the final bound on the magnitudes is $b = (n^2 d\beta)^{O(n^3 d)}$ but this does not affect the overall cost. With $d = O(n)$ and $\beta = 2^{O(n)}$, over the rationals using standard arithmetic, the Smith form and associated multipliers can be computed with $S(n) = \tilde{O}(n^{17})$ bit operations. \square

These bounds are very large and need some comments. In Kaltfen *et al.* (1987) the problem of computing the Hermite normal form is reduced to the problem of solving constant linear systems of dimension $O(n^3 d)$. Thus the reduction in Labhalla *et al.* (1995) is better, the problem is solved via the triangularization of the matrix $A^{(\delta)}$ of dimensions $O(n^2 d)$. Consequently, the lengths of the rational coefficients in the Hermite form and in the corresponding multiplier are bounded by $O(n^2 d \log(nd\beta)) = \tilde{O}(n^4)$. One can take advantage of the structure $A^{(\delta)}$ to reduce the number of operations needed for the triangularization (as it can be done when computing a gcd with the Sylvester's matrix). It can be seen that the triangularization can be done in nd steps of n^2 eliminations thus giving a cost of $\tilde{O}(nd \times n^2 \times n^2 d) = \tilde{O}(n^5 d^2)$ operations in F . Using standard arithmetic via Chinese remaindering this asks for $\tilde{O}(n^5 d^2 \times n^3 d) = \tilde{O}(n^8 d^3)$ bit operations. Thus using the reduction to linear system solution the bit complexity for the Hermite form using standard arithmetic is $\tilde{O}(n^{11})$. This is slightly more than the Las Vegas complexity

$\tilde{O}(n^8)$ in Storjohann (1994) for computing the Smith form (without the multipliers). The difference is essentially due to the fact that coefficients in the Smith form are smaller than in the multiplier for Hermite, their lengths are only in $O(n(d + \log \beta)) = \tilde{O}(n^2)$.

Now we see that the ratio of the deterministic complexity for the Smith form and multipliers to the one for the Hermite form is $O(n^6)$. This is quite large and we may hope that more investigations, especially to also take advantage of the structure of $A^{(\delta)}$ will reduce this ratio.

4.3. SINGULAR OR RECTANGULAR MATRICES

The normal forms has been originally developed for square matrices. However, the definition of the Smith form remains valid in the general case. A matrix A of rank r in $F[x]^{n \times m}$ is equivalent to a unique diagonal form with r non-zero invariant factors s_1, \dots, s_r . To compute the form for singular or rectangular matrices and extend theorem 4.1, we may follow the treatment in Kaltofen *et al.* (1990).

For a rectangular matrix A of $F[x]^{n \times m}$ of rank r , it is easily seen that the algorithm F -TSF leads to a slightly different form than in lemma 2.1 and in proposition 3.1. We get a unimodular matrix U of dimension n and an invertible constant matrix C of dimension m such that $T = UAC$ is in row echelon form, with non-zero entries only in its first r rows and with its diagonal entries given by $T_{i,i} = s_i, 1 \leq i \leq r - 1$. As we have done at corollary 4.1 we can zero the off-diagonal entries in the first $r - 1$ rows using unimodular column transformations. We are led to

$$T' = UAW = \begin{pmatrix} s_1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \ddots & 0 & \dots & \dots & \dots & 0 \\ 0 & \dots & s_{r-1} & 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & T'_{r,r} & T'_{r,r+1} & \dots & T'_{r,m} \\ 0 & \dots & \dots & 0 & & & \\ 0 & \dots & \dots & 0 & & 0 & \\ 0 & \dots & \dots & 0 & & & \end{pmatrix}.$$

If g is the gcd of the entries in row r then $s_1 s_2 \dots s_{r-1} g$ is the gcd of all the $r \times r$ minors of T' . Thus g is the last invariant factor s_r . From T', U and W the Smith form of A and multipliers are thus computed by solving a diophantine equation. Clearly, the overall cost given at theorem 4.1 in terms of field operations is not increased. Unfortunately, even if it will not either increase the bit complexity, the rational coefficients of the polynomials will grow again. In T' their lengths are bounded by $O(n^3 d \log(nd\beta))$, the polynomials in row r are of degree $O(nd)$ so the solution of the diophantine equation may generate numbers of length $O(n^4 d^2 \log(nd\beta))$.

COROLLARY 4.2. *The problem REDUCTION TO SMITH FORM for general polynomial matrices can be solved with $\tilde{O}(n^9)$ operations in F using asymptotically fast arithmetic and with $\tilde{O}(n^{17})$ bit operations using standard arithmetic when $F = \mathbb{Q}$.*

5. Conclusion

We have established that computing the Smith normal form and multipliers can be done in deterministic polynomial time. The proposed algorithm is interesting from a theoretical point of view. Concerning practical aspects, input matrices may be far from the

worst-case and our approach has to be used together with the previously known probabilistic algorithms. In addition, the method presented in the meantime in Villard (1995), should allow to improve the complexity bounds that remains very large compared with the ones for the Hermite form or with the probabilistic ones.

References

- Aho, A.V., Hopcroft, J.E., Ullman, J.D. (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley
- Bareiss, E.H. (1972). Computational solution of matrix problems over an integral domain. *J. Inst. Math. Appl.* **10**, 68–104.
- Cabay, S., Lam, T.P.L. (1977). Congruence techniques for the exact solution of integer systems of linear equations. *ACM Trans. Math. Software* **3** (4), 386–397.
- Chou, T.J., Collins, G.E. (1982). Algorithms for the solution of systems of linear diophantine equations. *SIAM J. Comput.* **11** (4), 687–708.
- Domich, P.D., Kannan, R., Trotter, L.E. (1987). Hermite normal form computation using modulo determinant arithmetic. *Mathematics of Operations Research* **12** (1), 50–59.
- Frumkin, M.A. (1977). Polynomial time algorithms in the theory of linear diophantine equations. In *Proc. Fundamentals of Computation Theory*, Springer-Verlag LNCS 56, 386–392.
- Gantmacher, F.R. (1966). *Théorie des Matrices*. Paris: Dunod
- Geddes, K.O., Czapor, R., Labahn, G. (1992). *Algorithms for Computer Algebra*. Kluwer: Academic Press.
- Hafner, J.L., Mc Curley, K.S. (1991). Asymptotically fast triangularization of matrices over rings. *SIAM J. Comput.* **20** (6), 1068–1083.
- Iliopoulos, C.S. (1989). Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. *SIAM J. Comput.* **18** (4), 658–669.
- Kaltofen, E., Krishnamoorthy, M.S., Saunders, B.D. (1987). Fast parallel computation of Hermite and Smith forms of polynomials matrices. *SIAM J. Alg. Disc. Meth.* **8** (4), 683–693.
- Kaltofen, E., Krishnamoorthy, M.S., Saunders, B.D. (1990). Parallel algorithms for matrix normal forms. *Linear Algebra and its Applications* **136**, 189–208.
- Kaminski, M., Paz, A. (1986). Computing the Hermite normal form on an integer matrix. Tech. Rep. # 417, Computer Science Dpt., Technion Israel Inst. of Tech.
- Kannan, R. (1985). Solving systems of linear equations over polynomials. *Theoretical Computer Science* **39**, 69–88.
- Kannan, R., Bachem, A. (1979). Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.* **8** (4), 499–507.
- Kazimirski, P.S., Petrichkovich, V.M. (1977). Equivalence of polynomial matrices. In *Theoretical and Applied Problems of Algebra and Differential Equations* (in Ukrainian), Naukova Dumka, Kiev, Ukraine, 61–66.
- Labhalla, S.E., Lombardi, H., Marlin, R. (1992). Algorithmes modulaires de calcul des réductions d'Hermite et de Smith. Preprint Université de Besançon, France.
- Labhalla, S.E., Lombardi, H., Marlin, R. (1995). Algorithmes de calcul de la réduction d'Hermite d'une matrice à coefficients polynomiaux. *Theoretical Computer Science*. To appear.
- Laidacker, M.A. (1969). Another theorem relating Sylvester's matrix and the greatest common divisor. *Mathematics Magazine* **42**, 126–128.
- Loos, R. (1982). Generalized polynomial remainder sequences. In (G.E. Collins, B. Buchberger, R. Loos, eds), *Computer Algebra - Symbolic and Algebraic Computation*, Springer-Verlag, 115–137.
- Petrichkovich, V.M. (1993). Semiscalar equivalence and the Smith normal form of polynomial matrices. Translated from *Matematicheskie Metody i Fiziko-mekhanicheskie Polya* **26**, 13–16 (1987).
- Schrijver, A. (1986). *Theory of Linear and Integer Programming*. Wiley-Interscience series in Discrete Mathematics.
- Shoup, V. (1993). Fast construction of irreducible polynomials over finite fields. In *Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, Austin USA*, 484–492.
- Storjohann, A. (1994). Computation of Hermite and Smith normal forms of matrices. Master's thesis, Dept. of Computer Science, University of Waterloo, Canada.
- Villard, G. (1995). Fast parallel algorithms for matrix reduction to normal forms. Tech. Rep. Apache 16, IMAG, Grenoble, France.