Computation of the Smith normal form of polynomial matrices

Gilles Villard

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

Abstract

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.

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 [7, 18] but some problems remain to be solved when they have to be computed.

In the case of matrices with integer entries, Frumkin, Kannan and Bachem have shown [6, 14] that the Hermite and Smith forms can be computed in polynomial time. The diagonalization is computed using repeated triangularizations of the matrix. Their bounds on the number of digits of the integers appearing during the calculations have been first improved by Chou and Collins [3] changing the order in which the computations were done, and then by several authors using modulo determinant arithmetic [5, 19, 9, 12]. More recently, asymptotically faster algorithms have been given in [8] and a rigourous study of modulo determinant methods has been developed in [16].

ACM-ISSAC '93-7/93/Kiev, Ukraine

These methods can also be applied for polynomial matrices and bound the degrees of the polynomials involved in the calculations, but they are not sufficient to correctly bound the coefficients of those 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 was given by Kannan [13]. And the first polynomial time algorithm for the Smith normal form appeared in [10]: but based on the Chinese remainder theorem, it did not compute the multipliers (or equivalence transforms) U and V such that if S is the Smith form of A, $UAV = S$. The last breakthrough was done by [11]. In this paper a Las Vegas probabilistic algorithm is 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. As a consequence, they have shown that there exist multipliers for the form, whose 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 "conditionning" matrix. The Hermite form of this new randomized matrix has, with high probability, the coefficients of the Smith form on its diagonal. Unfortunatly "good-conditionning" matrices, i.e. directly leading to the Smith form without repetition of Hermite, were characterized as not being root of a polynomial of a large degree in many variables. They were not computable in a deterministic manner.

We will show in section 3 that $\mathcal{REDUCTION\ TO\ SMITH\ FORM}$ over $Q[x]$ (the normal form and multipliers) is in $\mathcal{P}$, where $\mathcal{P}$ is the class of sequential polynomial time problems. We obtain the result by explicitly computing a good-conditionning 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, Lombardi and Marlin [15] to obtain good complexity bounds. By viewing Hermite as 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 $Q[x]$ reduces to the triangularization of a big matrix over Q. Using their idea we will show in section 4 that the computation of the Smith form over $Q[x]$ may also be computed by triangularizing a big matrix over Q.

2 Previous results

We will restrict ourselves to square non singular input matrices, but the approach could be generalized with no great difficulties to rectangular and singular matrices. In this section we recall some basic results [7, 18] concerning the Hermite and Smith normal forms of an input matrix A of dimension n whose entries are polynomials of $Q[x]$, and the main algorithms for their computation. The degrees and the coefficient lengths (log of the absolute values) of the entries of A are respectively bounded by d and β. A matrix of $Q[x]^{n\times n}$ is called unimodular if its determinant is a non zero element of Q.

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

(I) *Every non singular matrix A of $Q[x]^{n\times n}$ 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 $h_{1,1} = h_{1,1}^*$ and $h_{i,i} = h_{i,i}^*/h_{i-1,i-1}^*$, $i = 2, \ldots, n$.*

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

(III) *Every non singular matrix A of $Q[x]^{n\times n}$ 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 $s_i = s_i^*/s_{i-1}^*$, $i = 2, \ldots, n$.*

2.1 Kannan's algorithm for Hermite.

The Kannan's algorithm [13] 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: $A^{(0)} := A$, $n \times n$ 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: $H := A^{(n-1)}$. ∎

At step i unimodular row operations are performed on the first $i+1$ rows only. The entries $A_{i+1,j}$, $j = 1, \ldots, i$, 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 $r = pA_{j,j} + qA_{i+1,j}$. 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 1 ([13]) *Algorithm KHNF finds the Hermite normal form (and the multiplier) of a square non singular matrix over $Q[x]$ in polynomial time.*

A different polynomial time algorithm for computing the Hermite form can be found in [10]. 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 Q 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, Lombardi and Marlin [15]. As in [10], they reduce the Hermite form computation over $Q[x]$ to a linear system solution over Q, but they avoid the cost of finding the appropriate system. 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 [17] that if the Sylvester matrix is triangularized by row operations only, then the last non zero row gives the coefficients of the polynomial gcd.

The method first consists in associating to the input matrix A a matrix $A^{(\delta)}$ with entries in Q, 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 triangularizing $A^{(\delta)}$.

Let $e_1, \ldots, e_n$ the canonical basis of the module $Q[x]^n$. It provides a natural basis of the Q-vector space formed by the elements of $Q[x]^n$ whose entries have degrees less than $d+\delta$:

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

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 Q whose row-vectors are the

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

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

Theorem 2 ([15]) *The row-vectors of the Hermite normal form of A (written in the base $\mathcal{B}^{(\delta)}$) are computed by a triangularization of $A^{(\delta)}$ using row operations only, followed by the reduction of the off-diagonal entries by row operations.*

In section 4 below we extend this theorem, and perform a triangularization with column operations for the computation of the Smith form.

2.3 Randomized algorithm for Smith.

The usual method to compute the Smith normal form consists in iterating Hermite normal form computations on the matrix and on its transpose [14]. The number of iterations is theoretically bounded as $O(n^3)$ although in practice two iterations suffice. The randomized algorithm of [11] consists in pre-conditionning the input matrix A by multiplying it by a randomly chosen constant matrix. With high probability, the diagonal entries of the Hermite form, say H', of this new matrix are the entries of the Smith form. Consequently ([11], lemma 3.4), a second application of Hermite, on the transpose of H', gives the Smith form of A.

Algorithm RSNF

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

$C :=$ unit lower triangular matrix whose entries are chosen at random in Q.

$A' := AC$.

$A_1 :=$ Hermite normal form of A'.

$A_2 :=$ Hermite normal form of ${}^tA_1$.

Output: A_2 if it is diagonal or *Failed.* ∎

In fact, the entries of C can be chosen in a subset of Q whose construction is detailed in [11].

Theorem 3 ([11]) *Let A be a matrix of $Q[x]^{n\times n}$ 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 C does 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-conditionning matrices (that do not form a root of π). In sections 3 and 4 below we will show how to compute such matrices and consequently we will give a deterministic polynomial time algorithm to compute the Smith form.

Remark 1 *Let us refer to the characterizations (II) and (IV) of the diagonal entries of the Hermite and the Smith form. The diagonal entries of the Hermite form of A are the entries of the Smith form if and only if, for all i, the gcd of the $i \times i$ minors formed with the i first columns of A is equal to the gcd of all $i \times i$ minors of A.*

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 consists in computing a triangular form, but doing 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. Notice that it suffices to focus on the triangular form: as seen previously, the Smith form is then directly obtained by reducing the off-diagonal entries by column operations ([11], lemma 3.4).

Definition 1 *Let B be an $n \times n$ non singular matrix. A good conditionning of B is a $(n-1)$-uple $(\alpha_2, \alpha_3, \ldots, \alpha_n)$ of Q^{n-1} such that: if the first column of B is replaced by the linear combination of the other columns given by*

$$B'_1 := B_1 + \alpha_2 B_2 + \alpha_3 B_3 + \ldots + \alpha_n B_n,$$

then

$$\gcd_{1\le k\le n}(B'_{k,1}) = \gcd_{1\le k,l\le n}(B_{k,l}).$$

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

The triangular form with the "good" diagonal entries 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. The diagonal entries in the i first rows are the s_j, $j \leq i$, the diagonal entries of the Smith form:

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

Let $A^{(i)}_j$ 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 $Q[x]$-TSNF (Triangular Smith form)
Input: $A^{(0)} := A$, $n \times n$ non singular matrix.
for i from 1 to $n-1$
  Compute a good conditionning of $\bar{A}^{(i)}$:
    $(\alpha^{(i)}_{i+1}, \dots, \alpha^{(i)}_n) \in Q^{n-i}$.
  $A^{(i)}_i := A^{(i-1)}_i + \alpha^{(i)}_{i+1} A^{(i-1)}_{i+1} + \dots + \alpha^{(i)}_n A^{(i-1)}_n$.
  [Now $\gcd_{i \leq k \leq n}(A^{(i)}_{k,i}) = \gcd_{i \leq k,l \leq n}(A^{(i-1)}_{k,l})$.]
  Zero the sub-diagonal entries in column i using unimodular transformations on the last i rows.
  Reduce the upper-diagonal entries in column i by row operations.
Output: $A^{(n-1)}$. ∎

Provided that a good conditionning can always be found, we prove with the proposition below that this algorithm is correct, i.e. that it computes the "triangular Smith form". Then we will show how to compute such good transformations.

Proposition 1 *For a non singular square input matrix A of $Q[x]^{n \times n}$, algorithm $Q[x]$-TSNF computes, by unimodular transformations, a triangular matrix T whose diagonal entries are the entries of the Smith form of A. Furthermore, T is in Hermite normal form.*

Proof. Clearly, the transformations are unimodular. We prove by induction that at each step i, unit upper triangular operations on the columns of $A^{(i)}$ suffice to bring $A^{(i)}$ into the form:

$$\begin{pmatrix} s_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & \dots & s_i & 0 & 0 & 0 \\ 0 & \dots & 0 & & & \\ 0 & \dots & 0 & & \bar{A}^{(i+1)} & \\ 0 & \dots & 0 & & & \end{pmatrix},$$

non zero entries are in diagonal positions or in the last i rows and columns. Let this form be denoted by $[DIAG(s_1, \dots, s_i); \bar{A}^{(i+1)}]$. The conclusion will then come immediately: the latter form obtained from $A^{(n-1)}$ is the Smith form, since it is computed by unit upper triangular operations on the columns, the two matrices have the same diagonal entries.

At step $i = 1$, algorithm $Q[x]$-TSNF computes $A^{(1)}_{1,1} = gcd_{1 \leq k \leq n}(A^{(1)}_{k,1})$. Indeed, when the sub-diagonal entries are zeroed in column i using unimodular transformations on the last i rows, the diagonal entry is replaced by the gcd of the entries of the column. Now, from the definition of a good conditionning, $gcd_{1 \leq k \leq n}(A^{(1)}_{k,1}) = gcd_{1 \leq k,l \leq n}(A_{k,l}) = s_1$. Consequently, the off-diagonal entries in the first row are multiples of $s_{1,1}$ and can be zeroed by unit upper triangular operations.

We proceed by induction for $1 < i < n$. In the same way, at step i, algorithm $Q[x]$-TSNF computes $A^{(i)}_{i,i} = gcd_{i \leq k \leq n}(A^{(i)}_{k,i})$. Since a good conditionning has been used ($\bar{A}^{(i)}$ is not singular), $A^{(i)}_{i,i} = gcd_{i \leq k,l \leq n}(A^{(i-1)}_{k,l})$. Then, by induction for the rows k, $k < i$, and by substracting multiples of the diagonal entry for the row i, $A^{(i)}$ can be brought into the form $[DIAG(s_1, \dots, s_{i-1}, A^{(i)}_{i,i}); \bar{A}^{(i+1)}]$. From the definition, $s_1 s_2 \dots s_{i-1}$ divides all $(i-1) \times (i-1)$ minors and by construction, $A^{(i)}_{i,i}$ divides all the entries of $\bar{A}^{(i+1)}$. Now, all minors but the principal one can be computed from entries of $\bar{A}^{(i+1)}$ and from $(i-1) \times (i-1)$ minors: they are multiple of $s_1 s_2 \dots s_{i-1} A^{(i)}_{i,i}$. Since $s_1 s_2 \dots s_{i-1} A^{(i)}_{i,i}$ is a minor, it is the gcd of all $i \times i$ minors, and $A^{(i)}_{i,i} = s_i$.

The last assertion of the proposition comes from the reduction of the upper-diagonal entries at each step. □

We now turn to the good conditionnings. We give below an algorithm to compute them, the associated proposition shows that it runs in polynomial time.

Before, let us partly re-formulate a lemma of [11]. This lemma gives a construction for a good conditionning in the case of a $n \times 2$ matrix.

Lemma 1 *Let B be a $n \times 2$ matrix of rank 2 in $Q[x]$ with the degree of the entries bounded by d. There exists a "test polynomial" π in $Q[\alpha]$, 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 conditionning for B.

Proof. B is a matrix in $Q[x]^{n \times 2}$ of rank 2. Applying lemma 3.7 in [11], we know that there is a polynomial π in $Q[\alpha]$, of degree at most $2d$, such that if:
- R in $Q[x]^{2 \times 2}$ is unit lower triangular,
- H is the row equivalent echelon form of BR,
- s_1 is the first determinantal divisor of A,

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,

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

and by definition, the first determinantal divisor of B is the gcd of all the entries of B:

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

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

For a square input matrix B of dimension n, the algorithm computes a good conditionning 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 conditionning)
Input: B, $n \times n$ non singular matrix.
for j from 2 to n
  $g(x) := \gcd_{1 \le i \le n}(B_{i,1}, B_{i,j})$
  $\tilde{g}(x) := \gcd_{1 \le i \le n}(B_{i,1})$
  $\alpha_j := 0$
  while $\tilde{g}(x) \neq g(x)$
    $\alpha_j := \alpha_j + 1$
    $B_1 := B_1 + B_j$
    $\tilde{g}(x) := \gcd_{1 \le i \le n}(B_{i,1})$
Output: $(\alpha_2, \alpha_3, \dots, \alpha_n)$ ■

Proposition 2 *For a non singular square matrix B in $Q[x]^{n \times n}$ ($n > 1$) with the degree of the entries bounded by d, algorithm GC finds a good conditionning $(\alpha_2, \alpha_3, \dots, \alpha_n)$ that verifies $|\alpha_i| \le 2d$ for all i. In the worst case, $O(nd)$ greatest common divisors of n polynomials have to be computed: the number of operations is polynomialy bounded in the dimension and length of B.*

Proof. Algorithm GC consists in applying the lemma iteratively with the first and the $n-1$ other columns of B. We begin to show that the algorithm terminates after $O(nd)$ passing through 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 $2d+1$ roots. Each time the algorithm pass through the *while loop* a new value of α_i is tested, starting form the value 0. 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 pass through the *while loop* at most $(n-1)(2d+1)$ times.

Now we show that a good conditionning 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 verifies,

$$\begin{aligned} \gcd_{1 \le i \le n}(B_{i,1}^{(2)}) &= \gcd_i(B_{i,1} + \alpha_2 B_{i,2}) \\ &= \gcd_i(B_{i,1}, B_{i,2}). \end{aligned}$$

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

$$\begin{aligned} \gcd_{1 \le i \le n}(B_{i,1}^{(j)}) &= \gcd_i(B_{i,1}^{(j-1)} + \alpha_j B_{i,j}) \\ &= \gcd_i(B_{i,1}^{(j-1)}, B_{i,j}). \end{aligned}$$

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

$$\begin{aligned} \gcd_{1 \le i \le n}(B_{i,1}^{(j-1)}) &= \gcd_i(B_{i,1}^{(j-2)} + \alpha_{j-1} B_{i,j-1}) \\ &= \gcd_i(B_{i,1}^{(j-2)}, B_{i,j-1}), \end{aligned}$$

with the previous identity, this gives,

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

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

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

From there, the cost of the whole process is easily shown to be polynomial in the dimension of B and the lengths of its entries. The greatest common divisors are computed on polynomials that remain of degree d. As for the coefficient lengths of those latter polynomials, we have seen that only the first column-vector is modified. Assuming the lengths of the coefficients of the $B_{i,j}$ are bounded by β, and noticing that $B_{i,1}^{(n)} = B_{i,1} + \alpha_2 B_{i,2} + \dots + \alpha_j B_{i,j}$ and $|\alpha_j| \le 2d$, we get that all the coefficients of the $B_{i,1}^{(j)}$, for any i and j, have their lengths bounded as $\beta + \log(nd)$. □

Remark 2 *Algorithm TSNF associates to A a matrix C in $Q[x]^{n \times n}$,*

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

such that the diagonal entries of the Hermite form of $A' = AC$ are the diagonal entries of the Smith form of A. Furthermore, from the remark at the end of the previous section, we know that for all i, the gcd of the $i \times i$ minors formed with the i first columns of A' is equal

to the gcd of all $i \times i$ minors of A', and consequently, of A.

In conclusion, proposition 2 establishes that for a given matrix, a good conditionning can be computed in polynomial time. This is clearly not sufficient to prove that algorithm TSNF itself is polynomial. This latter result could certainly be obtained by reasoning as Kannan did in [13]: by performing a direct elimination in $Q[x]$. 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 $Q[x]$ to a problem over Q 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 at section 2.2. The problem of limiting the coefficient growth when computing gcd over $Q[x]$ is solved: the normal form is computed by a Gaussian elimination process. Furthermore linear algebra over Q leads to much simple bounds both for the coefficients and for the global complexity [1, 2]. All the results we will use about the Hermite normal form come from [15].

In this section we show that computing the Smith form also reduces to a triangularization over Q : it consists in working in a Q-vector space. Indeed, 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 conditionnings. 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 Q-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 [15] can be used for Smith. We keep the same notations than in section 2.2. In particular, the vectors of $Q[x]^n$ whose entries have degrees less than a fixed degree, will be also viewed as belonging to a Q-vector space. Let C be the constant matrix constructed by algorithm TNSF, and let L'_i be the row-vectors of $A' = AC$. We call "triangular Smith form" of A our target matrix: the Hermite form of A'.

Lemma 2 *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 Q-vector space that contains the row-vectors of the "triangular Smith form" of A.*

Proof. Let U be the multiplier for the Hermite form T of A': $UA' = UAC = T$, T is the output of algorithm TNSF, the "triangular Smith form". Let A'^* be the adjoint matrix of A', we have, $\det(A')U = A'^*T$. For any matrix M, we denote by $\deg(M)$ the maximum degree of the entries of M. The previous identity gives:

$$\deg(U) \leq \deg(A'^*) + \deg(T) - \deg(\det(A')),$$

since

$$\deg(T) \leq \deg(\det(A)) = \deg(\det(A'))$$

and

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

we get,

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

and the assertion of the lemma follows. □

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

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

Let L_i be the row-vectors of A. To the matrix A we associate the matrix $A^{(\delta)}$ whose row-vectors are the $[x^\delta L_1, \ldots, x^\delta L_n, x^{\delta-1}L_1, \ldots, x^{\delta-1}L_n, \ldots, L_1, \ldots, L_n]$ written in the base $\mathcal{B}^{(\delta)}$. We know that the Hermite form of A is obtained by triangularizing $A^{(\delta)}$.

Remark 3 *Each block of $(d+\delta+1)$ consecutive columns in $A^{(\delta)}$ stands for a column-vector in $Q[x]^{n(\delta+1)}$. $A^{(\delta)}$ consists in n consecutive blocks of $(d+\delta+1)$ columns. For instance adding α times a block Bck_j to a block Bck_i consists in adding, for all k, $1 \leq k \leq (d+\delta+1)$, α times the k-th column of Bck_j to the k-th column of Bck_i. Reciprocally, to each block-column one may associate in a natural way a vector of $Q[x]^{n(\delta+1)}$ (the entries of the matrix give the coefficients of the polynomials), a block-column operation therefore corresponds to the same operation in $Q[x]^{n(\delta+1)}$.*

In this new framework, we rewrite here the algorithm TSNF: we compute a triangularization of $A^{(\delta)}$ in $n-1$ steps by an usual Gaussian elimination [1]. Those $n-1$ steps correspond to a block-triangularization: at step i the sub-matrix consisting in the i first block-columns is in row echelon form.

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

rows and $n-i$ last block-columns of $A^{(\delta,i)}$.

Algorithm Q-TSNF (Triangular Smith form)
Input: $A^{(\delta,0)} := A^{(\delta)}$.
[The $n-1$ steps of the Gaussian block-elimination.]
for i from 1 to $n-1$
  Compute a good conditionning of $\bar{A}^{(\delta,i)}$:
    $(\alpha_{i+1}^{(i)}, \ldots, \alpha_n^{(i)}) \in Q^{n-i}$.
  [Block-column operations]
  $A_i^{(\delta,i)} := A_i^{(\delta,i)} + \alpha_{i+1}^{(i)} A_{i+1}^{(\delta,i-1)} + \ldots + \alpha_n^{(i)} A_n^{(\delta,i-1)}$.
  [Usual operations of Gaussian elimination over Q]
  [Zero the sub-diag. entries in block-column $A_i^{(\delta,i)}$]
  for k from $(i-1)(d+\delta+1)+1$ to $i(d+\delta+1)$
    Elimination of the sub-diag. entries in col. k.
Reduce the polynomials represented by the upper-diagonal entries by row operations.
Output: $T^{(\delta)} := A^{(\delta,n-1)}$. ∎

The proposition and the theorem below lead to our main result, they establish that algorithm Q-TNSF outputs the good form and that the Smith form is computed in polynomial time. At first let us look at the behaviour of the algorithm on a very basic example.

Example. Let

$$A = \begin{pmatrix} x-1 & 3x+2 \\ x-1 & 2x+3 \end{pmatrix}.$$

Taking $\delta = 1$ (the entries of U are of degree 1),

$$A^{(\delta)} = \begin{pmatrix} 1 & -1 & 0 & 3 & 2 & 0 \\ 1 & -1 & 0 & 2 & 3 & 0 \\ 0 & 1 & -1 & 0 & 3 & 2 \\ 0 & 1 & -1 & 0 & 2 & 3 \end{pmatrix}.$$

Rows 2 and 4 of the triangularization of $A^{(\delta)}$ without column operations would lead to the Hermite form of A:

$$\begin{pmatrix} 1 & -1 & 0 & 3 & 2 & 0 \\ 0 & 1 & -1 & 0 & 0 & 5 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \equiv \begin{pmatrix} x-1 & 5 \\ 0 & x-1 \end{pmatrix}.$$

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

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

rows 3 and 4 of the triangularization of the new matrix give

$$\begin{pmatrix} 1 & 6/5 - \frac{x}{5} \\ 0 & x^2 - 2x + 1 \end{pmatrix},$$

this would lead to the Smith normal form. ∎

Proposition 3 *For a non singular input square matrix A of $Q[x]^{n\times n}$, the algorithm Q-TSNF computes, in a polynomial number of operations, a triangular matrix T whose diagonal entries are the entries of the Smith form of A, and two matrices U unimodular in $Q[x]^{n\times n}$ and C non singular in $Q^{n\times n}$ such that $UAC = T$.*

Proof. Let Δ be number of columns in each block-column of $A^{(\delta)}$: $\Delta = d + \delta + 1$. We know that the triangularization $H^{(\delta)}$ of $A^{(\delta)}$ with only row operations gives the Hermite normal form: for each block $H_i^{(\delta)}$ of Δ columns of $H^{(\delta)}$, let $\mathcal{L}_i$ be the set of the indices of the rows whose first $i\Delta$ entries are not identically zero, and 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 the Hermite form of A, written in the base $\mathcal{B}^{(\delta)}$ [15].

We now turn to the Smith normal form. By extension of definition 1 at previous section we define a good conditionning of $\bar{A}^{(\delta,i)}$. In a natural way (remark 3) we associate to $\bar{A}^{(\delta,i)}$ that is written in the base $\mathcal{B}^{(\delta)}$, a matrix $\bar{A}^{(i)}$ over $Q[x]$. A good conditionning for $\bar{A}^{(\delta,i)}$ is defined to be a good conditionning for $\bar{A}^{(i)}$, with the associated block-column operations.

We look at what happen 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 $Q[x]^{n(\delta+1)}$ (remark 3). The Gaussian elimination on the first Δ columns computes the gcd of the polynomial entries in the first column-vector in $Q[x]^{n(\delta+1)}$. Since a block-column operations stands exactly for the same operation over $Q[x]^{n(\delta+1)}$, once applied the good conditionning, the Gaussian elimination on the first Δ columns computes the gcd of all the polynomial entries of the matrix over $Q[x]$. In the same way, at each step i, the Gaussian elimination on the first Δ columns of $\bar{A}^{(\delta,i)}$ computes the gcd of all the polynomial entries of $\bar{A}^{(i)}$. Finally, we know from the proof of algorithm $Q[x]$-TSNF, that those gcd are the diagonal entries of the Smith form. When the triangularization is completed, 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 $UAC = T$: U is computed using a bordering identity matrix and we have seen that the entries of C are the good conditionnings (remark 2).

It remains to see that the triangular form is obtained in polynomial time. The number of operations on polynomials is bounded by the Gaussian elimination and by proposition 2. Indeed, from lemma 2 the degrees are implicitly bounded by $\Delta - 1$, so each step i computes at most $(n-i)(\Delta-1)$ gcd ($(n-i)$ block-columns are involved) of at most $(n-i+1)(\delta+1)$ polynomials (number of rows of $\bar{A}^{(\delta,i)}$). As for the coefficients, let us look at what happens to the i-th block-column. It is modified only by the Gaussian elimination process

before step i, and it remains unchanged after step i. During step i, the algorithm performs
$A_i^{(\delta,i)} := A_i^{(\delta,i)} + \alpha_{i+1}^{(i)} A_{i+1}^{(\delta,i-1)} + \ldots + \alpha_n^{(i)} A_n^{(\delta,i-1)}$.
From proposition 2, since gcd of at most $n(\delta+1)$ polynomials of degrees at most Δ are computed, the $\alpha_j^{(i)}$ are bounded in absolute value by 2Δ. This is just an addition of a term in $O(\log n\Delta) = O(\log nd)$ to the coefficient lengths. □

Remark 4 *During the search of the good conditionning, the polynomial gcds can also be computed using the Gaussian elimination process: independently on each block-column.*

Theorem 4 *Computing the Smith normal form and multipliers over $Q[x]$, say $\mathcal{REDUCTION\ TO\ SMITH\ FORM}$ over $Q[x]$, is in $\mathcal{P}$. Indeed, the Smith form of a matrix of dimension n in $Q[x]^{n\times n}$ is computed by triangularizing a matrix of dimension $O(n^2d)$ in $Q^{n\times n}$, then by reducing the upper diagonal entries.*

Proof. This result is a direct consequence of the previous proposition. With no difficulty, the Smith form is computed in polynomial time from the triangular Smith form by reducing the upper diagonal entries using column operations. □

Corollary 1 *The coefficients appearing during algorithm Q-TSNF for computing the Smith normal form are of the same magnitude than during the computation of the Hermite normal form. The cost for Smith is the cost for Hermite plus the cost of computing $O(n^2d)$ gcds of $O(n^2d)$ polynomials.*

5 Conclusion

We have proven that computing the Smith normal form and multipliers can be done in polynomial time. Our algorithm leads to good sequential complexity bounds in the worst-case, but may be of low practical interest: matrices may be very far from the worst-case and the probabilistic algorithm of [11] gives very good results. From another point of view, we can hope that further studies will lead to another complexity result (see [4] for the definitions of parallel complexity classes): from a parallel computation point of view, $\mathcal{REDUCTION\ TO\ SMITH\ FORM}$ is not known to be in the class $\mathcal{NC}$ but only in $\mathcal{RNC}$.

Acknowledgment. I am grateful to referee 1 for his useful correction of lemma 1.

References

[1] E.H. Bareiss. Computational solution of matrix problems over an integral domain. *J. Inst. Math. Appl.*, 10:68–104, 1972.

[2] S. Cabay and T.P.L. Lam. Congruence techniques for the exact solution of integer systems of linear equations. *ACM Trans, Math. Software*, 3(4):386–397, 1977.

[3] T.J. Chou and G.E. Collins. Algorithms for the solution of systems of linear diophantine equations. *SIAM J. Comput.*, 11(4):687–708, 1982.

[4] S.A. Cook. The classification of problems which have fast parallel algorithms. In *Int. Conf. Foundations of Computation Theory, Borgholmm 1983, Lect. N. Comp. Sc. 158 pp 78-93*, 1983.

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

[6] M.A. Frumkin. Polynomial time algorithms in the theory of linear diophantine equations. In *Fundamentals of Computation Theory*, pages 386–392. LNCS 56, Springer, New-York, 1977.

[7] F.R. Gantmacher. *Théorie des matrices.* Dunod, Paris, France, 1966.

[8] J.L. Hafner and K.S. Mc Curley. Asymptotically fast triangularization of matrices over rings. *SIAM J. Comput.*, 20(6):1068–1083, December 1991.

[9] C.S. Iliopoulos. 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, August 1989.

[10] E. Kaltofen, M.S. Krishnamoorthy, and B.D. Saunders. Fast parallel computation of Hermite and Smith forms of polynomials matrices. *SIAM J. Alg. Disc. Meth.*, 8 4, pp 683-690, 1987.

[11] E. Kaltofen, M.S. Krishnamoorthy, and B.D. Saunders. Mr. Smith goes to Las Vegas: randomized parallel computation of the Smith normal form of polynomial matrices. In *EUROCAL'87*, pages 317–322. LNCS 378, Springer Verlag, 1989.

[12] M. Kaminski and A. Paz. Computing the Hermite normal form on an integer matrice. Technical Report 1986, Computer Science Dpt., TECHNION Israel Institute of Technology, June 1986.

[13] R. Kannan. Solving systems of linear equations over polynomials. *Theoretical Computer Science*, 39:69–88, 1985.

[14] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.*, 8 4, pp 499-507, 1979.

[15] S.E. Labhalla, H. Lombardi, and R. Marlin. Algorithmes de calcul de la réduction d'Hermite d'une matrice à coefficients polynomiaux. In *Comptes-Rendus de MEGA92, Nice, France.* Birkhauser, 1992. Submitted to JSC.

[16] S.E. Labhalla, H. Lombardi, and R. Marlin. Algorithmes modulaires de calcul des réductions d'Hermite et de Smith. Manuscript, April 1992.

[17] M.A. Laidacker. Another theorem relating Sylvester's matrix and the greatest common divisor. *Mathematics Magazine*, 42:126–128, 1969.

[18] M. Newman. *Integral Matrices.* Academic Press, 1972.

[19] A. Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in Discrete Mathematics, 1986.