Numerical algorithms for the computation of the Smith normal form of integral matrices

C. Koukouvinos*, M. Mitrouli†, and Jennifer Seberry‡

Abstract

Numerical algorithms for the computation of the Smith normal form of integral matrices are described. More specifically, the compound matrix method, methods based on elementary row or column operations and methods using modular or p-adic arithmetic are presented. A variety of examples and numerical results are given illustrating the execution of the algorithms.

AMS Subject Classification: Primary 65F30, Secondary 15A21, 15A36.

Key words and phrases: Smith normal form, numerical algorithms, integral matrices.

1 Introduction

The Smith normal form of matrices has applications to many problems in pure and applied mathematics: for example, in the determination of the invariant polynomials and the elementary divisors of polynomial matrices, see [5, p137-139]; in the matrix pencil approach for the study of linear time-invariant control systems, see[13];

*Department of Mathematics, National Technical University of Athens, Zografou 15773, Athens, Greece.

†Department of Mathematics, University of Athens, Panepistimiopolis, 15784, Athens, Greece.

‡Department of Computer Science, University of Wollongong, Wollongong, NSW 2522, Australia.

CONGRESSUS NUMERANTIUM 133 (1998), pp.127-162

in solving systems of linear diophantine equations [3]; in group theory for the determination of basis of groups [21]; for the computation of the set of finite and infinite elementary divisors, see [20]; in design theory, and more specifically in Hadamard matrices, SBIBD(v, k, λ), D-optimal designs etc, see [24, p410-415], [15, 16]; in integer linear programming, see [9, p317-354], in digital signal processing and in algebraic topology research.

First we introduce some notation and summarize results on the Smith normal form.

Let $\mathcal{R}^{m\times n}$ be the set of all $m \times n$ real matrices. The unit (*or* unimodular) elements of $\mathcal{R}^{m\times n}$ are those with determinant ± 1.

Definition 1 The matrices A, B of $\mathcal{R}^{m\times n}$ are equivalent (written $B \sim A$) if $B = PAQ$ for some unimodular matrices $P \in \mathcal{R}^{m\times m}$ and $Q \in \mathcal{R}^{n\times n}$.

The following Theorem is due to Smith [23] and has been reworded from the theorems and proofs in MacDuffee [17, p41] and Marcus and Minc [18, p44]. It is the most important theorem in the area of equivalence, and quite possibly the most important theorem in all of elementary matrix theory.

Theorem 1 *If $A = (a_{ij})$ is any integer (or with elements from any Euclidean domain) matrix of order n and rank r, then there is a unique integer (or with elements from any Euclidean domain) matrix*

$$D = \text{diag}\{a_1,\ a_2,\ \cdots,\ a_r,\ 0,\ \cdots,\ 0\},$$

such that $A \sim D$ and $a_i \mid a_{i+1}$ where the a_i are non-negative. The greatest common divisor of the $i \times i$ subdeterminants of A is

$$a_1 a_2 \cdots a_i.$$

If $A \sim E$ where

$$E = \text{diag}\{a_1,\ a_2,\ \cdots,\ a_i\} \oplus F$$

then a_{i+1} is the greatest common divisor of the non-zero elements of F. □

Definition 2 The a_i of theorem 1 are called the *invariants* of A and the diagonal matrix D is called the *Smith normal form* (SNF).

The above form of the SNF of a matrix was described in the original paper of Smith [23] on this topic. The SNF can be similarly defined on a nonsquare matrix.

Several numerical methods have been proposed and developed for the computation of the SNF of a given integral matrix, see [1,2,5,6,8,9,10, 15,16] and references therein.

In the present paper we present a survey of the most important existing numerical algorithms. Moreover, we develop a new numerical algorithm to obtain the SNF of a matrix using exclusively elementary row operations and greatest common divisor (GCD) evaluations. The described methods apply their computations to square matrices but they can also be used for nonsquare matrices too.

Throughout this paper, $\mathcal{Z}^{n\times n}$ denotes the set of all $n \times n$ integer matrices, the symbol $\| \ \|$ denotes any matrix norm and finally the notation $A(i:n,\ i:n)$ denotes a submatrix of the original matrix A consisting of rows $i, i+1, \ldots, n$ and columns $i, i+1, \ldots, n$ of A. All the algorithms presented are described in a MATLAB like language which is very convenient for the description of procedures involving matrices, see [19].

2 The compound-matrix method

2.1 Theoretical background

In this method the theoretical notion of compound matrices can be applied. More precisely, the following theoretical background is required. Let us first introduce some useful notation concerning sequences of integers and compound matrices.

Notation (2.1) see [18]

(i) $Q_{p,n}$ denotes the set of strictly increasing sequences of p integers ($1 \leq p \leq n$) chosen from 1, 2, ..., n. If $a, \beta \in Q_{p,n}$ we say that a precedes β ($a < \beta$), if there exists an integer t ($1 \leq t \leq p$) for which $a_1 = \beta_1, \ldots, a_{t-1} = \beta_{t-1}$, $a < \beta$, where a_i, β_i denote the elements of a, β respectively. This describes the lexicographic ordering of the elements of $Q_{p,n}$. The set of sequences $Q_{p,n}$ will be assumed to be lexicographically ordered.

(ii) Suppose $A = [a_{i,j}] \in \mathcal{R}^{m \times n}$, let k, p be positive integers satisfying $1 \leq k \leq m$, $1 \leq p \leq n$, and let $a = (i_1, i_2, \ldots, i_k) \in Q_{k,m}$ and $\beta = (j_1, j_2, \ldots, j_p) \in Q_{p,n}$. Then $A[a/\beta] \in \mathcal{R}^{k \times p}$ denotes the submatrix of A which contains the rows $i_1, i_2, \ldots, i_k$ and the columns $j_1, j_2, \ldots, j_p$.

(iii) Let $A \in \mathcal{R}^{m \times n}$ and $1 \leq p \leq min\{m, n\}$, then the p-th compound matrix or p-th adjugate of A is the $\binom{m}{p} \times \binom{n}{p}$ matrix whose entries are $\det(A[a/\beta])$, $a \in Q_{p,m}$, $\beta \in Q_{p,n}$ arranged lexicographically in a and β. This matrix will be designated by $C_p(A)$.

Example 1: If $A \in \mathcal{Z}^{3 \times 3}$ and $p = 2$ then

$$Q_{2,3} = \{(1,2),(1,3),(2,3)\} \text{ and } C_2(A) =$$

$$\begin{bmatrix} \det(A[(1,2)/(1,2)]) & \det(A[(1,2)/(1,3)]) & \det(A[(1,2)/(2,3)]) \\ \det(A[(1,3)/(1,2)]) & \det(A[(1,3)/(1,3)]) & \det(A[(1,3)/(2,3)]) \\ \det(A[(2,3)/(1,2)]) & \det(A[(2,3)/(1,3)]) & \det(A[(2,3)/(2,3)]) \end{bmatrix}$$

□

The above procedure produces the following numerical algorithm.

Algorithm COMREL

For a given matrix $A \in \mathcal{Z}^{m \times n}$ and for an integer p, $1 \leq p \leq min\{m, n\}$, the following algorithm computes the p-th compound matrix of A, $\mathrm{Cp}(A) \in \mathcal{Z}^{\binom{m}{p} \times \binom{n}{p}}$.
If $\underline{r} \in \mathcal{R}^n$ the symbol $A = [A, r]$ adds a column to matrix A while the symbol $A = [A; r]$ adds a row to matrix A. The notation $A = [\,]$ denotes an empty matrix while $a = [\,]$ denotes an empty vector.

Construct the sets
$Q_{p,m} = \{r_1, \ldots, r_{\binom{m}{p}}\}$
$Q_{p,n} = \{c_1, \ldots, c_{\binom{m}{p}}\}$
COMP = [ ]
for $i = 1, 2, \ldots,$ length$(Q_{p,m})$
    M = [ ]
    for $j = 1, 2, \ldots,$ length$(Q_{p,n})$
        elem = det$(A[r_i/q_j])$
        M = [M,elem]
    COMP = [COMP;M]

Algorithm 2.1

2.2 Application of compounds to Smith form evaluation

We apply the notion of compound matrices to the definitions given in Section 1. The following procedure is derived for the evaluation of the Smith form via compound matrices. We suppose that $A \in \mathcal{Z}^{n \times n}$ consists of coprime entries (otherwise the computation of the GCD of all the given integer entries of A will be required).

$D_1 := 1$
$a_1 := 1$
for $p = 2, \ldots, n$
    evaluate $C_p(A) = [c_{ij}]$, $i, j = 1, 2, \ldots, \binom{n}{p}$
    find $D_p :=$ GCD(c_{ij}), $i, j, = 1, 2, \ldots, \binom{n}{p}$
    $a_p := D_p / D_{p-1}$
$S_M :=$ diag$\{a_p\}$, $p = 1, \ldots, n$

Algorithm 2.2

2.3 Numerical interpretation of the Algorithm

The compound matrix technique can work satisfactorily for any given integral matrix. The internal determinantal computations required are based on using Gaussian elimination with partial pivoting which forms a stable numerical method. A severe disadvantage of this method is the rather high computational complexity, since calculations of order $O(k_p^2 \frac{n^4}{4})$, $k_p \in \mathcal{R}$ are required. Thus, although the method theoretically can work for the evaluation of any Smith normal form, in practice it is not recommended due to its high complexity. Another drawback of this method is that it is not possible to identify the left and right unimodular matrices and thus it is not suitable in cases where these matrices are needed. One reasonable method for compound matrix implementation is the parallel method as parallel SNF computations may be very effective in practice. In [12] parallel algorithms working for matrices of polynomials are presented.

Example 2: Let

$$A = \begin{bmatrix} 2 & 12 & 8 & 20 \\ 38 & 6 & 16 & 40 \\ 22 & 4 & 10 & 14 \\ 50 & 18 & 26 & 34 \end{bmatrix} \in \mathcal{Z}^{4\times 4}$$

be a given integral matrix. Compute its SNF ($S_M(A)$) using the compound matrix method.

$$D_1 = \text{GCD}(a_{ij}) = 2, \; a_1 = 2$$

STEP 1:

$$C_2(A) = \begin{bmatrix} -444 & -272 & -680 & 144 & 360 & 0 \\ -256 & -156 & -412 & 88 & 88 & -88 \\ -564 & -348 & -932 & 168 & 48 & -248 \\ 20 & 28 & -348 & -4 & -76 & -176 \\ 384 & 188 & -708 & -132 & -516 & -496 \\ 196 & 72 & 48 & -76 & -116 & -24 \end{bmatrix}$$

$D_2 := \text{GCD}(C_2(A)) = 4$

$a_2 := 4/2 = 2$

STEP 2:

$$C_3(A) = \begin{bmatrix} -184 & 4424 & 2992 & -1584 \\ 552 & 15144 & 8432 & -4464 \\ 552 & 3112 & 1008 & -880 \\ -184 & 3144 & 1200 & -1320 \end{bmatrix}$$

$$D_3 := \mathrm{GCD}(C_3(A)) = 8$$

$$a_3 := 8/4 = 2$$

STEP 3:

$$C_4(A) = 11776$$

$$a_4 = 11776/8 = 1472$$

$$S_M = \mathrm{diag}\{2, 2, 2, 1472\}$$

□

3 Elementary Row(Column) Operations Methods

In order to produce algorithms which have less computational complexity than the compound matrix method various different techniques have been developed to carry out the required computations. The central idea of these methods is to perform specific operations on the original matrix and finally modify it to an appropriate form from which the SNF can be directly derived. The following operations are defined:

1. Elementary Row(Column) Operations (EROS or ECOS)

   The EROS or ECOS operations consists of

   a. Addition of integer multiples of one row (column) to another.

   b. Multiplication of a row (column) by -1.

c. Interchange of two rows (columns).

2. GCD Operations

The GCD operations are divided into the following two classes:

a. GCD determination.
For a given matrix $A = [a_{ij}] \in \mathcal{Z}^{n \times n}$ let $g = \mathrm{GCD}(a_{ij}) = \mathrm{GCD}(A)$ be the GCD of its all entries. The above definition can be realized after the application of techniques based on the Euclidean algorithm. Subsequently we present the Euclidean algorithm which for two given numbers a,b constructs their gcd.

Algorithm EUCLID

```
while a or b ≠ 0
      if b = 0
         gcd=a
         break
      else if a > b
              t=a mod b
              a=a-tb
              if a = 0
                 gcd=b
                 break
              end
              else swap a and b
```

b. The Make To Divide property.
For a given matrix $A = [a_{ij}] \in \mathcal{Z}^{n \times n}$ in many approaches one random element a_{ij} of the matrix is required to divide another a_{kl}. For example,during reduction using row (column) operations then a_{ij} is required to divide a_{kj} (a_{ik}) for various values of k and random i, j.

For a given matrix $A = [\underline{r}_1, \underline{r}_2, \ldots, \underline{r}_n]^t \in \mathcal{R}^{n \times n}$ the following algorithm ensures the above property for the elements r_{ij} and r_{kj} when

row operations are performed. The symbol $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x whereas the symbol $\lceil x \rceil$ denotes the least integer greater than or equal to x. Let also sign(x) be $-1, 0, 1$ according to whether x is negative, zero or positive.

Procedure MakeToDivide

while r_{ij} does not divide r_{kj}
  if $r_{ij} > r_{kj}$
    $d = \lfloor r_{ij}/r_{kj} \rfloor$
    if $\text{sign}(r_{ij}) = \text{sign}(r_{kj})$
      $\underline{r}_i^t := \underline{r}_i^t - d \cdot \underline{r}_k^t$
    else
      $\underline{r}_i^t := \underline{r}_i^t + d \cdot \underline{r}_k^t$
  else
    $d = \lfloor r_{kj}/r_{ij} \rfloor$
    if $\text{sign}(r_{kj}) = \text{sign}(r_{ij})$
      $\underline{r}_k^t := \underline{r}_k^t - d \cdot \underline{r}_i^t$
    else
      $\underline{r}_k^t := \underline{r}_k^t + d \cdot \underline{r}_i^t$
end

Algorithm 3.1

Implementation of the Algorithm

When $r_{ij} > r_{kj}$, Algorithm 3.1 requires at most $log_2(r_{kj})$ operations for its execution. This is due to the fact that after the execution of each row operation the row affected is reduced to less than one half of its previous value.

In the following we develop some of the most important methods of this category.

3.1 Bradley's method

Bradley's method was developed in [1].

For a given matrix $A \in \mathcal{Z}^{n\times n}$ the basic idea of Bradley's method is twofold:

1. After the application of EROS and GCD operations reduce A to a diagonal form $D = \text{diag}\{d_1, \ldots, d_n\}$.

2. Using GCD operations ensure the Smith property $d_1|d_2|\ldots|d_n$ for the diagonal elements of D.

In the following, we describe the numerical version of the method.

Let $A = [\underline{r}_1, \underline{r}_2, \ldots, \underline{r}_n]^t \in \mathcal{Z}^{n\times n}$ be a given integer matrix. The following algorithm constructs its SNF stored on a diagonal matrix S_M. The notation $A(\kappa : \lambda, j)$ denotes the vector $(r_{k,j}, r_{k+1,j}, \ldots, r_{\lambda,j})^t$.

STEP 1: Reduction of A to a diagonal form
for $i = 1, 2, \ldots, n$
if $\underline{r_i}^t \neq \underline{0}^t$
for $j = i+1, \ldots, n$
if r_{ii} does not divide r_{ij} and r_{ji}
make it divide them
end
end
Apply Gaussian elimination to the i-th column of A
transforming it to a form A_G with the property
$A_G(2 : n-i+1, i) = 0$
$A := A_G$
end
$S_M := A$
STEP 2: Ensure the property $S_{M_{11}}|S_{M_{22}}|\ldots|S_{M_{nn}}$
for the elements of S_M
for $i = 1, 2, \ldots, n$
if $S_{M_{ii}}$ does not divide $S_{M_{kk}}$,
for some $k = i+1, \ldots, n$
Make it divide
end
end

Algorithm 3.2

Implementation of the Algorithm

In [1] analytical remarks concerning the performance of the above algorithm have been developed. The required "make it divide" property can be implemented using the algorithm 3.1. An upper bound for the required computational complexity of STEP 1 equals $\frac{1}{2}n^2$ $(\log_2 \ |\det(A)| + 3) \ + \frac{1}{3}2n^3$. STEP 2 of the above algorithm in general, requires only a small number of multiplications. A significant disadvantage of this algorithm is that it is not polynomial as was pointed out in [4]. Thus, in transforming the integer matrix to its SNF the number of digits of integers occuring in intermediate steps does not appear to be bounded by a polynomial in the length of the input data.

Example 3: Let A be the integer matrix of Example 2. Next we compute its SNF, $S_M(A)$ according to Bradley's algorithm.

STEP 1: By Gauss transformations modify it to the form

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & -222 & -136 & -340 \\ 0 & -128 & -78 & -206 \\ 0 & -282 & -174 & -466 \end{bmatrix}$$

STEP 2: Apply procedure MakeToDivide and after 6 steps modify it to the form:

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 6 & 340 \\ 0 & 14 & -4 & 206 \\ 0 & -36 & 30 & 466 \end{bmatrix} \xrightarrow[\text{Gaussian}]{\text{Apply}} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -46 & 2174 \\ 0 & 0 & 138 & 6586 \end{bmatrix}$$

STEP 3: Apply procedure MakeToDivide and after 4 steps modify it to the form:

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 10 \\ 0 & 0 & -262 & 162 \end{bmatrix} \xrightarrow[\text{Gaussian}]{\text{Apply}} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1472 \end{bmatrix}$$

The last matrix gives the SNF $S_M(A)$ of matrix A.

□

3.2 Kannan and Bachem's method

Kannan and Bachem's method was developed in [11]. For a given matrix $A \in \mathcal{Z}^{n\times n}$ the method is based on the computation of the Hermite normal form (HNF) of A.

Theorem 2 *Given a nonsingular $n\times n$ integer matrix A, there exists an $n\times n$ unimodular matrix K such that AK is lower triangular with positive diagonal elements. Further, each off-diagonal element of AK is nonpositive and strictly less in absolute value than the diagonal element in its row. AK is called the Hermite normal form (HNF) of A.* □

Next, we describe the algorithm HNF which successively puts the principal minors of orders $1, 2, \ldots, n$ of A into HNF.

Algorithm HNF

Permute the columns of A such as
$det\,(1:k, 1:k) \neq 0, \forall\ k = 1, \ldots, n$
$i := 1$
While $i \neq n$
  for $j := 1 : i$
    calculate
    $r = GCD(A(j,j), A(j,i+1))$
    and p, q such as
    $r = p \cdot A(j,j) + q \cdot A(j,i+1)$

$$D = \begin{bmatrix} p & -A(j,i+1)/r \\ q & A(j,j)/r \end{bmatrix}$$

$A(:,j) = A(:,j) \cdot D$
$A(:,i+1) = A(:,i+1) \cdot D$
end
if $j > 1$
REDUCE SIZE (j, A)
end
REDUCE SIZE $(i+1, A)$
$i := i + 1$
end

Algorithm 3.3

Function REDUCE SIZE

If $A(:,k) < 0$
$A(:,k) = -A(:,k)$
end
For $z = 1 : k-1$
$A(:,z) = A(:,z) - \lceil A(k,z)/A(k,k) \rceil A(:,k)$
end

Algorithm 3.4

Implementation of the Algorithm

The above algorithm is polynomial since all the intermediate calculations are polynomial [11]. The following result was proved in [11].

Lemma 3 *If* $A = A^{(1)}, A^{(2)}, \ldots, A^{(n)}$ *are the matrices produced after the application of algorithm* **HNF**, *then*

$$max|a_{i,j}^{(k)}| \leq 2^{3n} \cdot n^{20n^3} \cdot max|a_{i,j}^{(1)}|^{12n^3}$$

□

Thus a bound for the size of the intermediate numbers can be produced in terms of the given initial matrix.

An extension of the above algorithm can easily calculate the left an right multipliers. In the case where row operations are performed, instead of column, Algorithm 3.3 computes the Left Hermite normal form of the matrix (LHNF) i.e. matrix A will be transformed into an upper triangular form H with positive diagonal elements. Further, each off-diagonal element of H is nonpositive and strictly less in absolute value than the diagonal element in its column.

Based on the above algorithms, we next describe an algorithm computing the SNF of A. In the algorithm the following notation is used. HNF$(n-i, n-i, A)$ is the procedure which puts the bottom-right-hand minor consisting of the last $(n-i)$ rows and columns into Hermite normal form. LHNF$(n-i, i+1, A)$ is the procedure which puts the submatrix of A consisting of the last $(n-i)$ rows and the column $i+1$ into left Hermite normal form.

Algorithm SNF

$i := 0$
While $i \neq n$
  repeat
    for $k = 1 : n-1$
      Call LHNF(n-i,i+1,A)
      Call HNF(n-i,n-i,A)
    end
    if $A(i+1, i+1) \not| A(j,k),\ i+1 \leq j \leq k,$
      $i+1 \leq k \leq n$
      Specify j, k
      $A(:, i+1) = A(:, i+1) + A(:, k)$
    end
  until $\{A(i+1, i+1) \mid A(j,k),\ i+1 \leq j \leq k,$
    $i+1 \leq k \leq n\}$
end

Algorithm 3.5

Implementation of the Algorithm

Algorithm SNF is polynomial since it performs at most $n^2(logn\ max|a_{i,j}|) + 2n$ HNF computations.

Example 4: Let A be the integer matrix of Example 2. Next we compute its SNF, $S_M(A)$, using Kannan and Bachem's method.

STEP 1: Compute the LHNF taking as submatrix the first column of the given matrix:

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & -94 & -58 & -134 \\ 0 & -128 & -78 & -206 \\ 0 & -282 & -174 & -466 \end{bmatrix} \underset{\text{HNF}}{\longrightarrow} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & -326 & -64 & 1472 \end{bmatrix}$$

STEP 2: Compute the LHNF taking as submatrix the second column of the modified matrix:

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & -64 & 1472 \end{bmatrix} \underset{\substack{\text{HNF} \\ \text{LHNF}}}{\longrightarrow} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1472 \end{bmatrix} = S_M(A)$$

□

Remark 1 Chou and Collins' method

An extension and improvement of Kannan and Bachem's method has been developed in [3]. In this method, starting with a given square nonsingular matrix $A \in \mathcal{R}^{n \times n}$ we apply preconditioning in order to ensure that all its principal minors are nonsingular. In the sequel, the $n \times n$ identity matrix is adjoined to the bottom of A, and the new matrix is called $\bar{A}$. Since $\bar{A}$ is of rank n, there exists a nonsingular submatrix A^* of $\bar{A}$ consisting of r linearly independent rows of A and $n-r$ rows of the $n \times n$ identity matrix. The main idea of the algorithm is the transformation of A^* into a pseudo-Hermite matrix by applying a sequence of unimodular transformations to $\bar{A}$. A square nonsingular matrix is a *pseudo-Hermite* matrix or in a *pseudo-Hermite* form, if it is lower triangular and the absolute value of any off-diagonal element is less than the absolute value of the diagonal element to its right. The transformation of A^* into a pseudo-Hermite matrix enables the derivation of very good bounds,

of order $n + r\lfloor \log_\beta(|maxa_{i,j}| + 1)\rfloor, maxa_{i,j} \neq 0$, on the lengths of the coefficients pertaining to $\bar{A}$.

□

3.3 The new method (Simplify and Divide method).

In order to perform efficiently the row operations and reduce as much as possible the number of GCD operations required on the one hand and the size of the elements produced on the other, the following new technique is proposed.

Let $A \in \mathcal{Z}^{n\times n}$ be a given matrix. We apply to this matrix the following procedure:

For $i = 1, 2, \ldots, n$
STEP 1: Make the GCD of $A(i:n, i:n)$ appear in the (i,i) position of A
STEP 2: Eliminate the entries of the vector $(a_{i+1,i}, a_{i+2,i}, \ldots, a_{n,i})^t$

After the termination of the above procedure the original matrix A will have been transformed to an upper triangular form. The diagonal elements of this form are actually the diagonal elements of the SNF of A. In the implementation of the above technique the problem of ensuring the GCD exists as an element is encountered.

Theorem 4 *Let $A = [a_{ij}]$ be an integer matrix. Suppose y=gcd of all elements of A. Then y occurs in a matrix which is obtained from A by using elementary row and column operations.*

Proof. Let $A = [\underline{r}_1, \underline{r}_2, \ldots, \underline{r}_n]^t \in \mathcal{R}^{n\times n}$ be a given matrix with $y = \text{GCD}(A) \neq r_{ij} \quad \forall i, j$. We want to modify the matrix A to $A' = [\underline{r'}_1, \ldots, \underline{r'}_i, \ldots, \underline{r'}_k, \ldots, \underline{r'}_n]^t$ for which an index j exists such as $y = r'_{ij}$ or r'_{kj}. Let $\gcd(A)$ is y. Divide all the elements of A by y forming A'. Put y outside into the invariant. Now if ± 1 appears in

the matrix we are through. Otherwise we know that the gcd of A' is 1 (otherwise we wouldn't have had gcd(A)=y.) We check to see if any row has two elements with gcd =1. If two elements have gcd=1 we use the Euclid process to get zero in one of these columns and 1 in the other. If not, every element in every row has a row gcd y_i for the ith row. If for some i, $y_i = 1$ we check if the greatest common divisors of the elements of the ith row are coprime. If two greatest common divisors of specific elements are coprime we use the Euclid process to make these greatest common divisors to appear and then to get zero in one of these columns and 1 in the other. We now check to see if any column has two elements with gcd =1 If two elements in a column have gcd = 1 use Euclid process to get zero in one of those rows and zero in the other. If not, every element in every column has a column gcd c_j for the jth column. If for some j, $c_j = 1$ we check if the greatest common divisors of the elements of the jth column are coprime. If two greatest common divisors of specific elements are coprime we use the Euclid process to make these greatest common divisors to appear and then to get zero in one of these rows and 1 in the other. If none of the above cases holds,we select randomly two rows i, j and two columns k, m. This means the gcd of the matrix can only occur across the contents of a rectangle for we have

$$\begin{bmatrix} y_i c_k & y_i c_m \\ y_j c_k & y_j c_m \end{bmatrix}$$

Suppose gcd($y_j c_k, y_i c_m$) = 1 (it must occur somewhere as gcd(A') = 1. If it is not straightforward, by performing elementary operations we can always get gcd(ac_k, bc_m) = 1). Use Euclid process to get

$$\begin{bmatrix} c_k & ac_m \\ 0 & bc_m \end{bmatrix}$$

Suppose we have gcd(c_k, bc_m) = 1 so $Ac_k + Bbc_m = 1$. Further suppose gcd(c_k, ac_m) $\neq 1$, for if it were 1 we could use the Euclid process to get a 1 and a 0 in the first row by using elementary column operations

Add A times the first column to the second and B times the

second row to the first obtaining

$$\begin{bmatrix} c_k & ac_m + Ac_k + Bbc_m = ac_m + 1 \\ 0 & bc_m \end{bmatrix}$$

Now if $\gcd(c_k, ac_m) \neq 1$ then $\gcd(c_k, ac_m + 1) = 1$ and we use the Euclid process to obtain 1 and 0 in the first row.

We now use a similar procedure where the gcd is not one for each pair. Suppose $\gcd(c_k, ac_m) = \alpha$, $\gcd(a, b) = \beta$ and $\gcd(c_k, bc_m) = \gamma$. Write $\alpha = A_1 c_k + B_1 ac_m$, $\beta = A_2 a + B_2 b$ and $\gamma = A_3 c_k + B_3 bc_m$. Add A_3 times the first column to the second and B_3 times the second row to the first obtaining

$$\begin{bmatrix} c_k & ac_m + \gamma \\ 0 & bc_m \end{bmatrix}.$$

Now suppose $\gcd(c_k, ac_m) \neq 1$ and $\gcd(c_k, bc_m) \neq 1$. Since c_k divided every element of its column we use elementary row operations to ensure c_k is the only non-zero element in its column. Rearrange rows so we have

$$\begin{bmatrix} c_k & b_1 c_m & d_1 c_n & . & . & . \\ 0 & b_2 c_m & d_2 c_n & . & . & . \\ 0 & b_3 c_m & d_3 c_n & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \end{bmatrix}$$

We now continue supposing the gcd of no pair of diagonal elements is 1, nor the gcd of pairs of elements in any row or column. Suppose $\gcd(b_2, b_3, \ldots, b_n) = \beta$. Then we use the Euclidean process, elementary row operations and rearrangement of rows to obtain

$$\begin{bmatrix} c_k & \beta_1 c_m & e_1 c_n & f_1 c_p & . & . & . \\ 0 & \beta c_m & e_2 c_n & f_2 c_p & . & . & . \\ 0 & 0 & e_3 c_n & f_3 c_p & . & . & . \\ 0 & 0 & e_4 c_n & f_4 c_p & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix}$$

where $\beta_1 < \beta$. Let $\gamma = \gcd(e_3, e_4, \ldots, e_n)$ and $\gamma_1 < \gamma$, $\gamma_2 < \gamma$. Then proceeding as before we obtain

$$\begin{bmatrix} c_k & \beta_1 c_m & \gamma_1 c_n & \ldots & \delta_1 c_r \\ 0 & \beta c_m & \gamma_2 c_n & \ldots & \delta_2 c_r \\ 0 & 0 & \gamma c_n & \ldots & \delta_3 c_r \\ 0 & 0 & 0 & \ldots & \delta_4 c_r \\ . & . & . & \ldots & . \\ . & . & . & \ldots & . \\ 0 & 0 & 0 & \ldots & \delta c_r \end{bmatrix}$$

□

The following examples demonstrates the use of the above Theorem.

Example 5: Make the gcd of the following matrix to appear among its entries.

$$A = \begin{bmatrix} 3150 & 170170 & 868434 & 2485830 \\ 4410 & 190190 & 1036518 & 2803170 \\ 11025 & 293930 & 1397046 & 3213390 \\ 7350 & 248710 & 1108002 & 3063930 \end{bmatrix}$$

Since gcd(A)= 1, how can we make 1 to appear? We find the gcd's of the rows and columns of the matrix. We have: $y_1 = 2, y_2 = 2, y_3 = 1, y_4 = 2$, $c_1 = 105, c_2 = 70, c_3 = 42, c_4 = 30$. Since $y_3 = 1$ we check if the gcd's of the elements of the third row are coprime. If two gcd's of specific elements are coprime we can use the Euclid process to make these gcd's to appear and then to get zero in one of these columns and 1 in the other. We remark that: gcd(a_{31}, a_{32})= 35, gcd(a_{31}, a_{33})= 21, gcd(a_{31}, a_{34})= 15, gcd(a_{32}, a_{33})= 14, gcd(a_{32}, a_{34})= 10, gcd(a_{33}, a_{34})= 6, and gcd (gcd (a_{31}, a_{33}), gcd(a_{32}, a_{34}))= 1. Thus, we will make 21 and 10 to appear in the third row by using the Euclidean process. Finally we get the matrix A' equal to:

$$10^9 * \begin{bmatrix} -0.034726566 & -6.66225415 & 0.003284988 & 5.05923526 \\ -0.035349762 & -7.71117077 & 0.003343956 & 5.85576986 \\ 0.000000021 & 0.000000010 & 0.000000147 & 0.000000020 \\ -0.013071198 & -3.67395673 & 0.001236564 & 2.78995834 \end{bmatrix}$$

Since 21 and 10 appeared in the final matrix, by applying again the Euclid process we will make 1 to appear among the entries of the matrix. □

Example 6: Make the gcd of the following matrix to appear among its entries.

$$A = \begin{bmatrix} 6 & 6 \\ 10 & 15 \end{bmatrix}$$

Since gcd(A)= 1, how can we make 1 to appear? We find the gcd's of the rows and columns of the matrix. We have: $y_1 = 6, y_2 = 5,\ c_1 = 2, c_2 = 3$. The gcd of the matrix will occur across the contents of the rectangle:

$$\begin{bmatrix} 6 \cdot 2 & 6 \cdot 3 \\ 5 \cdot 2 & 5 \cdot 3 \end{bmatrix}$$

If we subtract the second row from the first we get:

$$\begin{bmatrix} 2 & 3 \\ 5 \cdot 2 & 5 \cdot 3 \end{bmatrix}$$

for which gcd(2,3) = 1. By using the Euclid process we get:

$$\begin{bmatrix} 2 & 3 \\ 0 & 0 \end{bmatrix}$$

and since gcd(2,3) = 1, after the application of Euclid process the gcd will appear in the matrix. □

A crucial step in reducing the size of the elements produced at intermediate steps for each value of i is the division of the current matrix $A(i:n, i:n)$ by its present GCD g_i.

Thus we can finally compute the following SNF for A: diag$\{\overbrace{1,\ldots,1}^{\leftarrow ind_1 - 1 \rightarrow}\}, g_1 * \{\overbrace{1,\ldots,1}^{\leftarrow ind_2 - ind_1 \rightarrow}\}, g_1 * g_2 * \{\overbrace{1,\ldots,1}^{\leftarrow ind_3 - ind_2 \rightarrow}\}, \ldots, g_1 * \ldots * g_n * \{\overbrace{1,\ldots,1}^{\leftarrow n - ind_n + 1 \rightarrow}\}\}$ The vectors fact $= (g_1, g_2, \ldots, g_n)^t$ and ind $= (ind_1, \ldots, ind_n)^t$ form the factors and their indicies for the SNF of A. Next we describe the numerical algorithm for this technique.

Algorithm SDM

Given a matrix $A \in \mathcal{Z}^{n \times n}$ the following procedure computes its SNF, $S_M(A)$. We suppose that a_{11} is the GCD of all $a_{i,j}$, $i,j = 1,2,\ldots,n$.

function S_M = SMITH(A)
n:= rows(A) S_M:= [ ] fact:= [ ] ind:=[ ]
for $i = 1,2,\ldots,n-1$
  Perform Gauss operations on the matrix
  A:= A_G such as $A(2 : n-i+1, i) = 0$
  $S_M := [S_M, |a_{11}|]$
  $A := A(2 : n-i+1, 2 : n-i+1)$
  g:= GCD(a_{ij}), $i,j = 1,2,\ldots,n-i+1$
  if $g \neq 1$
    $A := A/g$
    fact:= [fact, g]
    ind:= [ind, $i+1$]
  end
  if $|a_{11}| \neq 1$
    make GCD appear in the (1,1) position of A
    by performing elementary row/column operations
  end
$S_M = [S_M, |a_{nn}|]$

Algorithm 3.6

In order to develop a convenient method to produce 1 (which forms the GCD) in the (1,1) position of each submatrix of A, the following technique can be applied:

if $a_{ik} = 1$ for some i,k
  Interchange rows 1, i and column 1, k of A
else if
  two specific entries of columns or rows of A
  differ by 1, perform a direct subtraction of
  them and move the GCD to the (1,1) position by
  interchanging
  else
    Make the GCD appear by using the

```
            procedure MakeToDivide
            Move the GCD into (1,1) position
            by interchanging.
        end
    end
```

Algorithm 3.7

Implementation of the algorithm

For a given matrix $A \in \mathcal{Z}^{n \times n}$ the algorithm requires in its implementation the following three operations: Gaussian elimination, GCD determination and GCD appearance. Gaussian elimination of specific entries is a process requiring $O(n^3)$ operations. Let us suppose that M is the maximum value of the matrix entries produced through the steps of the elimination. Determination of the GCD of a matrix is a process requiring $O(n^2 \ln M)$ computations; since the Euclidean algorithm for the determination of the GCD of two numbers a, b with $1 \leq a, b \leq M$ requires $0.842766 \ln M + 0.06$ computations (see [12]). Finally, when the procedure MakeToDivide is needed, an extra $O(\log_2 M)$ operations will be spent. In total we see that the whole process requires $O(n^3 + n^2 \ln M)$ computations. In trying to give an upper bound to the number of computations required, we need to know the size of the elements produced in the matrix. Unfortunately, since the algorithm is concerned only with integer arithmetic and only determinantal operations are allowed, we cannot apply any kind of scaling which can reduce and bound the elements of the matrix. Also, since pivoting is not used in the Gaussian process, as the GCD must be always be kept in the (1,1) position of the matrix, we cannot guarantee the size of the elements produced after several steps of Gaussian repetition. A drawback of the SDM method is that we cannot always specify the required row operations in order to ensure the $\gcd(a_{i,j}, a_{k,j}) = \gcd(A)$.

3.4 The phenomenon of "entry explosion"

During the execution of algorithms derived from row operation methods the following two important numerical problems appear:

1. Since no pivoting is used in Gaussian elimination or in other similar techniques, the size of the emergent error matrix cannot be bounded exactly.

2. We cannot reduce by scaling the size of the matrices computed at each step of the methods.

Due to these problems all row operation methods for computing the SNF of integer matrices suffer from the problem of **coefficient growth or entry exposion or expression swell**. Therefore, when the SNF of matrices of large order is required, large coefficients are expected to appear during the intermediate steps. Theoretically, the only method which does not face the problems of coefficient growth is the compound matrix method since the internal determinantal computations are performed in a stable manner.

In order to keep entries small heuristic techniques can be used. These techniques are time-consuming and not fully effective but in some cases can be rather useful. In the sequel we describe some of these techniques.

1. A column reduction technique

In the new Simplify and Divide method, the procedure NORMCOL helps to keep the elements of the matrices within reasonable ranges. Also the division of the matrix by its GCD provides a remarkable reduction in the size of the emerging elements.

In order to reduce the sizes of the matrix elements achieved after each Gaussian iteration we apply the following determinantal scaling to the matrix.

Procedure NORMCOL

Let $A = [\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_n] \in \mathcal{R}^{m \times n}$ be a given matrix. The following algorithm produces a scalar matrix A_S satisfying the property $\|A_S\| \leq \|A\|$

Reorder the columns of A such that
$\|\underline{c}_1\| \leq \|\underline{c}_2\| \leq \ldots \leq \|\underline{c}_n\|$
$A_S := A$
norma:= $\|A\|$
norms:= $\|A_S\|$

while norms <= norma
norma:= norms
for $k = n, n-1, \ldots, 2$
if $\text{sign}(c_{1k}) = \text{sign}(c_{1k-1})$
$\underline{c}_k := \underline{c}_k - \underline{c}_{k-1}$
else
$\underline{c}_k := \underline{c}_k + \underline{c}_{k-1}$
Reorder the columns of A so that
$\|\underline{c}_1\| \leq \|\underline{c}_2\| \leq \cdots \leq \|\underline{c}_n\|$
$A_S := A$
norms:= $\|A_S\|$

Algorithm 3.8

Implementation of the algorithm

A major difficulty connected with the above algorithm is deciding when to apply it. Practical experience showed that if the algorithm was applied just once or twice at critical times the entries did not exhibit further explosion.

2. Rosser-type techniques

In [3] the following technique was proved to restrain coefficient growth very well in the early stages of the algorithm. Although the growth becomes fast in the final stage of Chou and Collins' algorithm it is still quite moderate compared to that in some other algorithms.

Procedure ROSSER

Let $A = [\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_n] \in \mathcal{Z}^{n \times n}$ be a given matrix. The following algorithm implements Rosser's technique. We assume that $a_{i,j} \geq 0$ (otherwise multiplication by -1 in the corresponding column can take place).

Reorder the columns of A such that
$c_{1,1} \geq c_{1,2}, \ldots \geq c_{1,n}$
while $c_{1,2} \neq 0$
$A(:,1) = A(:,1) - \lfloor c_{1,1}/c_{1,2} \rfloor A(:,2)$

Sort $A(:,1), A(:,2), \ldots, A(:,n)$ into descending order according to their leading elements

end

Algorithm 3.9

Implementation of the algorithm

At the end of the above algorithm the first row of the matrix will be of the form $c, 0, \ldots, 0$, where $c = gcd(c_{1,1}, c_{1,2}, \ldots, c_{1,n})$. Since $c_{1,1}$ and $c_{1,2}$ are the largest and the second largest elements in the first row of A, the integer $\lfloor c_{1,1}/c_{1,2} \rfloor$ computed in the above algorithm is usually small so the part $A(2:n,:)$ of the matrix will contain uniformly small elements, especially when n is large, while other methods will find a $A(2:n,:)$ with some large elements and the rest quite small, including many zeros and ones.

3.5 Numerical Results

The Simplify and Divide method was programmed on a IBM - compatible 486/33 computer using MATLAB. This machine has $\beta = 2$, word length 32 bits and floating point relative accuracy provided from MATLAB of $1.0 * 10^{-16}$. In the remainder of the paper we present some representative examples. At the end of each example the total number of floating point required operations and the execution time required are estimated using appropriate MATLAB functions. More examples and the MATLAB code of the algorithm are available from the authors.

Example 7: Let A be the integer matrix of Example 2. We compute its SNF according to SDM algorithm.

By procedure NORMCOL the original matrix is reduced to:

$$A_1 = \begin{bmatrix} -1 & 3 & 5 & 12 \\ -9 & 6 & -2 & 6 \\ 4 & 2 & 3 & 2 \\ 8 & 9 & 4 & 4 \end{bmatrix}, \text{factor} = \{2\}$$

SNF of a perturbed original matrix.

STEP 1: By Gauss and simplification transformations the matrix is modified to the form:

$$\begin{bmatrix} -1 & 3 & 5 & 12 \\ 0 & -1 & 5 & 14 \\ 0 & -13 & 5 & -21 \\ 0 & -10 & 22 & 38 \end{bmatrix}$$

STEP 2: Repeating the Gaussian transformations we obtain:

$$\begin{bmatrix} -1 & 3 & 5 & 12 \\ 0 & -1 & 5 & 14 \\ 0 & 0 & -18 & 14 \\ 0 & 0 & -5 & -37 \end{bmatrix} \xrightarrow[\text{ToAppear}]{\text{Make}} \begin{bmatrix} -1 & 3 & 5 & 12 \\ 0 & -1 & 5 & 14 \\ 0 & 0 & 1 & -245 \\ 0 & 0 & -3 & -1 \end{bmatrix}$$

STEP 3: After the final Gaussian operations we have:

$$\begin{bmatrix} -1 & 3 & 5 & 12 \\ 0 & -1 & 5 & 14 \\ 0 & 0 & 1 & -245 \\ 0 & 0 & 0 & -736 \end{bmatrix}$$

$S_M(A) = 2 * \text{diag}\{1, 1, 1, 736\}$

flops: 337, time: 0.17sec.

□

Example 8: To construct a Hadamard matrix of order 12, we observe that $12 = 11 + 1$. The quadratic elements of GF(11) are 1, 3, 4, 5 and 9. Let Q be the circulant matrix of order 11 with first row $(0 - + - - - + + + - +)$, where -,+ represents -1,1 respectively.

Then the matrix

$$H = \begin{bmatrix} 0 & \underline{e}^T \\ -\underline{e} & Q \end{bmatrix} + I_{12}$$

is a Hadamard matrix of order 12, where $\underline{e}^T = (1, 1, \ldots, 1)$ is the 1×11 vector of 1's.

We compute the Smith normal form of H.

$$S_M(H) = \text{diag}\{1, \underbrace{2,2,2,2,2}_{5 \text{ times}}, \underbrace{6,6,6,6,6}_{5 \text{ times}}, 12\}$$

flops: 3400, time: 0.22sec.

See [10, p.410] for a theoretical formula.

□

Example 9: A design is a pair (X, B) where X is a finite set of elements and B is a collection of (not necessarily district) subsets B_i (called blocks) of X. A balanced incomplete block design, BIBD(v, b, r, k, λ), is an arrangement of v elements into b blocks such that:

(i) each element appears in exactly r blocks;

(ii) eack block contains exactly $k(< v)$ elements; and

(iii) each pair of distinct elements appear together in exactly λ blocks.

As $r(k-1) = \lambda(v-1)$ and $vr = bk$ are well known necessary conditions for the existence of BIBD(v, b, r, k, λ) we denote the design by BIBD(v, k, λ).

The incidence matrix of a (v, b, r, k, λ) design is a $b \times v$ matrix $A = (a_{ij})$ defined by

$$a_{ij} = \begin{cases} 1 & \text{if the } i\text{th block contains the } j\text{th element,} \\ 0 & \text{otherwise} \end{cases}$$

If $b = v$, then $r = k$ and the design is said to be symmetric. This is denoted by SBIBD(v, k, λ). If A is the incidence matrix of a SBIBD(v, k, λ) then

$$A^T A = (k - \lambda)I + \lambda J$$

where I is the $v \times v$ indentity matrix and J is the $v \times v$ matrix every entry of which is 1.

Let S be the circulant matrix of order 31 with first row:

$$(1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$

which is the incidence matrix of a SBIBD(31, 6, 1). Then

$$S^T S = 5I + J$$

where I is the indentity matrix of order 31, and J is the 31 x 31 matrix with every element 1. We compute the Smith normal form of S:

$$S_M(S) = \text{diag}\{\underbrace{1,1,\ldots,1}_{16 \text{ times}}, \underbrace{5,5,\ldots,5}_{14 \text{ times}}, 30\}$$

flops:60343 , time:0.61 sec.

See[10, p.411] for a theoretical formula.

□

4 p-adic and modular arithmetic methods

In order to avoid the phenomenon of "entry explosion" that characterised the elementary row or column operations methods, modular techniques can be used. These techniques are fast and their main characteristic is the performance of calculations in prime fields $\mathcal{Z}_p$ rather than $\mathcal{Z}$. Next, we describe some of the methods developed.

4.1 Gerstein's method

Gerstein's method was developed in [6]. For a given matrix $A \in \mathcal{Z}^{n \times n}$ the method is based on matrix equivalence over principal ideal domains using the technique of localization from commutative algebra.

Let R be a principal ideal domain with quotient field $F \supset R$ and let p be a fixed prime element of R. Every element $x \in F = F - \{0\}$

can be written in the form $x = p^\nu \alpha/\beta, \alpha, \beta \in R$ relatively prime to p and $\nu \in Z$. The integer ν is uniquely determined by x, and we write $\nu = ord_p x$. We also define $ord_p 0 = \infty$, with the convention that $\infty > \nu$ for all $\nu \in Z$. Now define the *localization of R with respect to the prime ideal (p)*:

$$R_{(p)} \stackrel{def}{=} \{x \in F \mid ord_p x \geq 0\}$$

Thus $R_{(p)}$ is the subring of F generated by R and the inverses in F of all elements of R that are outside (p). An element $\epsilon \in R_{(p)}$ is a unit (a *p-adic unit*) if and only if $ord_p \epsilon = 0$. Also, p is the only prime element of $R_{(p)}$, except for associates; hence every nonzero element $x \in R_{(p)}$ has the form $x = \epsilon p^\nu$ for some p-adic unit ϵ and $\nu = ord_p x \geq 0$. The divisibility relation $\alpha \mid \beta$ holds in $R_{(p)}$ if and only if $ord_p \alpha \leq ord_p \beta$. The ring $R_{(p)}$ is a principal ideal domain, and it is a local ring with unique maximal ideal (p). Results over $R_{(p)}$ will be called *local*, while those over R are *global* results.

Fix a complete set $\mathcal{P}$ of nonassociated prime elements for $\mathcal{R}$; all primes under discussion from now on will be assumed to come from $\mathcal{P}$, and the letter p will also denote such a prime. The next theorem describes the local-global principle.

Theorem 5 *Let $A, B \in \mathcal{R}^{n \times n}$. Then $A \equiv B$ if and only if $A \equiv_p B$ for all $p \in \mathcal{P}$; moreover,*

$$S(A) = \prod_{p \in \mathcal{P}} S_p(A)$$

□

Next we describe an algorithm for computing the SNF of a given matrix $A = (a_{i,j}) \in \mathcal{R}^{n \times n}$ using the technique previously described.

Procedure P-ADIC

Compute $|A|$
Factor $|A|$ into primes
$|A| = \epsilon p_1^{\alpha_1} \ldots p_r^{\alpha_r}, p_i$ are distinct
primes in $\mathcal{P}$, ϵ is an R-unit, and $\alpha_i \geq 1$ for $1 \leq i \leq r$
for $i = 1, \ldots, r$
  if $\alpha_i = 1$
    $S_{p_i}(A) = diag(1, \ldots, 1, p_i)$
  else if $\alpha_i > 1$
    $\nu_1 = ord_p a_{1,1}$
    $S_{p_i}(A) = (p^{\nu_1} + S_p((\frac{a_{1,1}a_{i,j} - a_{i,1}a_{1,j}}{p^{\nu_1}}))_{2 \leq i,j \leq n}$
  end
end
$S_p(A) = \prod_{1 \leq i \leq r} S_{p_i}(A)$

Algorithm 3.10

Example 10: Let A be the integer matrix of Example 2. Next we compute its SNF, $S_M(A)$ according to Gerstein's method.

$A = A/2$, factor=$\{2\}$.

$|A| = 736 = 2^5 \cdot 23$

Thus, $S_M(A) = S_2(A) \cdot S_{23}(A)$

But $S_{23}(A) = diag\{1, 1, 1, 23\}$ whereas

$$S_2(A) = (1) + S_2(\begin{bmatrix} -111 & * & * \\ * & * & * \\ * & * & * \end{bmatrix}) = diag\{1, 1, 1, 2^5\}$$

(Since -111 is a 2-adic unit, we don't have to compute the other elements of the matrix). Hence we have
$S_M(A) = 2 \cdot diag\{1, 1, 1, 2^5 \cdot 23\}$.

□

Remark 2. Other modular methods Other modular methods ,without requiring integer factorizations, have also been developed. In [8], for a given matrix $A \in \mathcal{Z}^{m \times n}$, the following technique is applied.

1. Determine the number r of nonzero elementary factors $b_{i,i}, i = 1, \ldots, r$ of the given matrix A.
2. Calculate S a multiple of $\prod_1^r b_{i,i}$.
3. Perform Gauss-Jordan elimination in Z_s the ring of integers modulo S.

Detailed comments about the implementation of the above technique can be found in [8].

In [10] a different modular technique is implemented. Actually, the original matrix $A \in \mathcal{Z}^{n \times n}$ is extended by adding to its end the matrix $det(A)I_n$. Subsequently Gaussian elimination operations mod $det(A)$ are performed. Finally, another modular technique is described in [7].

In applications to group theory the matrices arising are not usually square. One important method for computing the SNF of such matrices is the LLL algorithm [22].

5 Conclusions

There is no reasonable way of expecting to find the "best" strategy for SNF computation. According to the nature of the specific application one can select the method that is more suitable for a given matrix. For matrices with reasonable sizes (less than 100) elementary row (column) operations methods can be applied. The MATLAB code of the new SDM method is available from the authors.

In various computational group theory, number theory and homology theory applications, matrices with dimensions into the thousands are arising. Then modular techniques must be employed for their SNF computation. The modular techniques described in [8] are built into the computer algebra language CAYLEY [2]. Other computer algebra packages including MAGMA, GAP, PARI, KANT and QUOTPIC can also handle similar computations. Finally, a free computer algebra system available for MACS performing computation of the SNF and tested for at least 500 × 500 sparse matrices can be downloaded from: http://www.math.unl.edu/~ bharbour/fermat/fermat.html

The following table summarizes the basic characteristics of each of the methods described.

Compound matrix method	*Origin*: This method works on the given matrix applying the compound matrix definition without performing any row or column operations on the matrix and thus it overcomes problems of numerical instability. *Numerical characteristics* • High computational complexity • Does not compute the transformation matrices
Bradley's method	*Origin*: This method is based on explicit calculation of the GCD and of a set of multipliers for each of the rows and columns. *Numerical characteristics* • Rapid coefficient growth • It is not a polynomial algorithm

Kannan and Bachem's method	*Origin*: This method transforms successively all the $(i+1) \times (i+1)$ principal minors of the matrix into its Hermite normal form. *Numerical characteristics* • The number of algebraic operations and the number of digits of all intermediate numbers are bounded by polynomials in the length of the input data • The transformation matrices are computed
Chou and Collin's method	*Origin*: This method transforms the original matrix to its pseudo-Hermite form. *Numerical characteristics* • More effective polynomial time bounds compared with those of Kannan and Bachem's method are given • The algorithm controls the intermediate expression swell very well
Simplify and Divide method	*Origin*: This method performs exclusively row operations and GCD evaluations on the original matrix *Numerical characteristics* • Reduces the size of the coefficients by dividing with the GCD of the matrix

Gerstein's method	*Origin*: This method applies the technique of localization from commutative algebra *Numerical characteristics* • Integer factorization techniques are employed • The transformation matrices are not computed

Table 1: Comparison of existing methods

References

[1] G. H. Bradley, Algorithms for Hermite and Smith normal matrices and linear diophantine equations.*Math. of Computation*, 25 (1971), 897-907.

[2] J. Canon and W. Bosma, *A Handbook of Cayley Functions*, Computer Algebra Group, Univ. of Sydney, 1991.

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

[4] M. A. Frumkin, Polynomial time algorithms in the theory of linear diophantine equations, M. Karpinski ed., *Fundamentals of Computation Theory*, Springer, Lecture Notes in Computer Science Vol. 56., New York (1977) 386-392.

[5] F. R. Gantmacher, *The Theory of Matrices*, Vol.1, Chelsea, New York, 1959.

[6] L. J. Gerstein, A local approach to matrix equivalence. *Linear Algebra Appl.*, 16 (1977), 221-232.

[7] J. L. Hafner and K. S. McCurley, Asymptotically fast triangulization of matrices over rings. *SIAM J. of Computing*, 20(6) (1991), 1068-1083.

[8] G. Havas, D. F. Holt and S. Rees, Recognizing badly presented Z-modules. *Linear Algebra Appl.*, 192 (1993), 137-163.

[9] T. C. Hu, *Integer Programming and Network Flows.* Addison-Wesley, Reading Mass., 1969.

[10] 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 (1989), 658-669.

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

[12] E. Kaltofen, M. S. Krishnamoorthy and B. D. Saunders, Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM J. Alg. Discr. Meth.*, 8 (1987), 683-690.

[13] N. Karcanias and G. Kalogeropoulos, Geometric theory and feedback invariants of generalized linear systems: A matrix pencil approach. *Circuits Systems Signal Process*, 8 (1989), 375-397.

[14] D. E. Knuth, *The Art of Computer Programming, Seminumerical Algorithms*, Vol II, Addison-Wesley, Reading Mass., 1969.

[15] C. Koukouvinos, M. Mitrouli and J. Seberry, On the Smith normal form of D-optimal designs. *Linear Algebra Appl.*, 247 (1996), 277-295.

[16] C. Koukouvinos, M. Mitrouli and J. Seberry, On the Smith normal form of weighing matrices. *Bull. Inst. Combin. Appl.*, 19 (1997), 57-69.

[17] C. C. Macduffee, *The Theory of Matrices.* Reprint of First Ed., Chelsea, New York, 1964.

[18] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities.* Allyn and Bacon, Boston, 1964.

[19] MATLAB, *High Performance Numeric Computation and Visualization Software.* User and Reference Guide, The MathWorks Inc., 1992.

[20] M. Mitrouli and G. Kalogeropoulos, A compound matrix algorithm for the computation of the Smith form of a polynomial matrix. *Numerical Algorithms*, 7 (1994) 145-159.

[21] M. Newman, *Integral Matrices.* Academic Press, New York, 1972.

[22] C.C. Sims, *Computing with Finitely Presented Groups.* Cambridge University Press, 1994.

[23] H. J. S. Smith, Arithmetical notes. *Proc. London Math. Soc.*, 4 (1873), 236-253.

[24] J. Seberry Wallis, Hadamard Matrices, Part IV, *Combinatorics: Room squares, sum free sets and Hadamard matrices.* Lecture notes in Mathematics, Vol. 292, eds. W. D. Wallis, A. P. Street and J. Seberry Wallis, Springer-Verlag, Berlin - Heidelberg - New York, 1972.

[25] J. Wilkinson, *Rounding Error in Algebraic Processes.* Her Majesty's Stationery Office, London, 1985.