Computing 74, 377–388 (2005)
Digital Object Identifier (DOI) 10.1007/s00607-004-0104-0

Computing
Printed in Austria

Reduction of Smith Normal Form Transformation Matrices

G. Jäger

Received September 14, 2004
Published online: November 15, 2004

Abstract

Smith normal form computations are important in group theory, module theory and number theory. We consider the transformation matrices for the Smith normal form over the integers and give a presentation of arbitrary transformation matrices for this normal form. Our main contribution is an algorithm that replaces already computed transformation matrices by others with small entries. We combine methods from lattice basis reduction with a procedure to reduce the sum of the squared entries of both transformation matrices. This algorithm performs well even for matrices of large dimensions.

AMS Subject Classifications: 65F05, 65F30.

Keywords: Smith normal form, transformation matrices, lattice basis reduction.

1. Introduction

The key to the solution of the classical problem

- For $A \in \mathbb{Z}^{m,n}$, $b \in \mathbb{Z}^m$ find *all* solutions $x \in \mathbb{Z}^n$ of the system of linear Diophantine equations $A \cdot x = b$.

is the *Smith normal form*. A matrix in $\mathbb{Z}^{m,n}$ with rank r is in Smith normal form (SNF), if it is a diagonal matrix with the first r diagonal elements being divisors of the next diagonal element and the remaining diagonal elements being zero. There exist unimodular matrices $U \in GL_m(\mathbb{Z})$ and $V \in GL_n(\mathbb{Z})$, called *transformation matrices*, such that UAV is in SNF [13]. There are many algorithms for efficiently computing the SNF. Some of these algorithms also compute the transformation matrices [3], [4], [6], [14].

The solution of the above problem is as follows:

(1) Compute the SNF C of A with transformation matrices U, V, i.e., $C = UAV$.

(2) Determine all solutions $y \in \mathbb{Z}^n$ with $C \cdot y = Ub$ (as C is in SNF, this is simple).

(3) We get the set of *all* solutions $x \in \mathbb{Z}^n$ as Vy with the $y \in \mathbb{Z}^n$ from 2.

All this is straightforward and well-known [8], but keep in mind that the transformation matrices $U \in GL_m(\mathbb{Z})$, $V \in GL_n(\mathbb{Z})$ are in general not unique.

If we want to compute the solutions of a system of linear Diophantine equations numerically, then we have to compute $U \in GL_m(\mathbb{Z})$ and $V \in GL_n(\mathbb{Z})$, and practical considerations call for matrices with small entries. In this paper we show how to compute such matrices.

In Sect. 3, we show how to get all transformation matrices U', V', if two fixed transformation matrices U, V are given. If we take only one arbitrary transformation matrix as fixed, then the other transformation matrix is uniquely determined, if and only if the rank of the input matrix equals the number of the rows and columns, respectively. These results can easily be generalized for any Euclidean ring.

In Sect. 4, we present our main result, an algorithm which produces transformation matrices with small entries. The first part of the algorithm is based on lattice basis reduction and uses size and LLL reduction. These steps are similar to methods used for the related *Hermite normal form* (HNF) which needs only one transformation matrix V. Havas, Majewski, Matthews [5] gave an HNF algorithm which uses size and LLL reduction steps during the algorithm and produces small entries. Their algorithm is implemented in the program package MAGMA, see [10]. Sims [12] proposed to use size and LLL reduction for a given transformation matrix.

Storjohann [14] was the first to consider the problem of small transformation matrices for the SNF. His method gives exponential upper bounds for the entries of the transformation matrices, but in practice it is not an optimal method, see Sect. 5.2.

Our idea is to employ any SNF algorithm and then reduce the transformation matrices (obtained). As for full rank matrices the size and LLL reduction steps cannot be applied, we develop a new procedure which reduces one transformation matrix at the expense of the other.

For an improved version of this algorithm we show that for a pre-assigned transformation matrix the Euclidean norm of the other transformation matrix is minimal up to a linear factor.

In Sect. 5, we present experimental results showing how well the algorithm performs for matrices of large dimension and small, medium and large rank.

2. Notations and Definitions

A matrix $C \in \mathbb{Z}^{m,n}$ with rank r is in *Smith normal form*, if the following conditions hold:

(a) C is a diagonal matrix.

(b) $C_{i,i} \in \mathbb{N}$ for $1 \leq i \leq r$.

(c) $C_{i,i} \mid C_{i+1,i+1}$ for $1 \leq i \leq r-1$.

(d) $C_{i,i} = 0$ for $r+1 \leq i \leq \min\{m,n\}$.

The nontrivial diagonal elements of C are called *elementary divisors* of C. Matrices $U \in GL_m(\mathbb{Z})$ and $V \in GL_n(\mathbb{Z})$ such that $C = UAV$ is in SNF are called *transformation matrices* for the SNF of A. In this case the matrices U and V are called *corresponding*.

For linearly independent vectors $b_1, \ldots, b_n \in \mathbb{R}^m$ the set $L(b_1, \ldots, b_n) := \left\{ \sum_{i=1}^n t_i \cdot b_i \middle| t_i \in \mathbb{Z} \right\} \subset \mathbb{R}^m$ is called a *lattice* with *lattice basis* $b_1, \ldots, b_n$ and *rank* n. For a lattice L with basis $b_1, \ldots, b_n$ an arbitrary basis $\bar{b}_1, \ldots, \bar{b}_n$ is given by $[\bar{b}_1, \ldots, \bar{b}_n] = [b_1, \ldots, b_n] \cdot T$ with an arbitrary $T \in GL_n(\mathbb{Z})$, see [1]. The lattice basis reduction tries to find a basis with short norm.

We consider the standard scalar product $< \cdot, \cdot >$ and the Euclidean norm $\| \cdot \| := \| \cdot \|_2$. For $a \in \mathbb{R}$ let $\lceil a \rfloor$ be the integer nearest to a and $E_{s,t}$ the matrix which is 1 in the entry (s, t) and 0 in all other entries. Further let $\mathrm{Diag}(d_1, \ldots, d_r)$ be the square diagonal matrix with diagonal elements $d_1, \ldots, d_r$.

3. Transformation Matrices for the SNF

Definition 1: *Let $C \in \mathbb{Z}^{m,n}$. We define:*

(a) $L(C) := \{W \in GL_m(\mathbb{Z}) \mid \exists Z \in GL_n(\mathbb{Z}) \text{ with } WCZ = C\}$.

(b) $R(C) := \{Z \in GL_n(\mathbb{Z}) \mid \exists W \in GL_m(\mathbb{Z}) \text{ with } WCZ = C\}$.

Lemma 1: *For $C = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{Z}^{m,n}$ in SNF with $D = Diag(d_1, \ldots, d_r)$,*

(a) $L(C) = \left\{ \begin{pmatrix} F & * \\ 0 & G \end{pmatrix} \middle| F \in GL_r(\mathbb{Z}),\ D^{-1}FD \in GL_r(\mathbb{Z}),\ G \in GL_{m-r}(\mathbb{Z}) \right\}$,

(b) $R(C) = \left\{ \begin{pmatrix} K & 0 \\ * & L \end{pmatrix} \middle| K \in GL_r(\mathbb{Z}),\ DKD^{-1} \in GL_r(\mathbb{Z}),\ L \in GL_{n-r}(\mathbb{Z}) \right\}$.

Remark 1: *Note that for $F = (F_{i,j})_{1 \le i,j \le r} \in GL_r(\mathbb{Z})$, $D^{-1}FD \in GL_r(\mathbb{Z})$ is equivalent to $\frac{d_i}{d_i} \mid F_{j,i},\ 1 \le i < j \le r$. Analogously, for $K = (K_{i,j})_{1 \le i,j \le r} \in GL_r(\mathbb{Z})$, $DKD^{-1} \in \mathbb{Z}^{r,r}$ is equivalent to $\frac{d_j}{d_i} \mid K_{i,j},\ 1 \le i < j \le r$.*

Proof: (a) Let M be the right-hand expression of (a). $L(C) \subseteq M$: Let $W \in L(C)$. By Definition 1, $W \in GL_m(\mathbb{Z})$ and $WCZ = C$ for some $Z \in GL_n(\mathbb{Z})$. Hence:

$$WC = CZ^{-1}. \tag{1}$$

Partition $W \in GL_m(\mathbb{Z})$, $Z^{-1} \in GL_n(\mathbb{Z})$ with submatrices $F \in \mathbb{Z}^{r,r}$, $X \in \mathbb{Z}^{r,m-r}$, $Y \in \mathbb{Z}^{m-r,r}$, $G \in \mathbb{Z}^{m-r,m-r}$ and $K \in \mathbb{Z}^{r,r}$, $S \in \mathbb{Z}^{r,n-r}$, $T \in \mathbb{Z}^{n-r,r}$, $L \in \mathbb{Z}^{n-r,n-r}$:

$$W = \begin{pmatrix} F & X \\ Y & G \end{pmatrix}, \quad Z^{-1} = \begin{pmatrix} K & S \\ T & L \end{pmatrix}$$

With these notations, (1) is equivalent to

$$\begin{pmatrix} FD & 0 \\ YD & 0 \end{pmatrix} = \begin{pmatrix} DK & DS \\ 0 & 0 \end{pmatrix}$$

Since D is invertible, $Y = 0$. So W has the form $W = \begin{pmatrix} F & X \\ 0 & G \end{pmatrix}$. From $W \in GL_m(\mathbb{Z})$ we get $F \in GL_r(\mathbb{Z})$ and $G \in GL_{m-r}(\mathbb{Z})$. From $FD = DK$ and $F \in GL_r(\mathbb{Z})$ it follows $D^{-1}FD \in GL_r(\mathbb{Z})$ and we obtain $W \in M$. $\underline{M \subseteq L(C)}$: Let $W \in M$. Because of $F \in GL_r(\mathbb{Z})$ and $G \in GL_{m-r}(\mathbb{Z})$ we have:

$$W \in GL_m(\mathbb{Z}). \tag{2}$$

Let $K := D^{-1}F^{-1}D = (D^{-1}FD)^{-1}$. Because of $D^{-1}FD \in GL_r(\mathbb{Z})$ it follows $K^{-1} \in GL_r(\mathbb{Z})$ and therefore $K \in GL_r(\mathbb{Z})$. With $L \in GL_{n-r}(\mathbb{Z})$ arbitrary we have for $\begin{pmatrix} K & 0 \\ * & L \end{pmatrix} \in GL_n(\mathbb{Z})$:

$$\begin{aligned} \begin{pmatrix} F & * \\ 0 & G \end{pmatrix}\begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} K & 0 \\ * & L \end{pmatrix} &= \begin{pmatrix} FD & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} K & 0 \\ * & L \end{pmatrix} \\ &= \begin{pmatrix} FDK & 0 \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \end{aligned} \tag{3}$$

From the definition of $L(C)$ and because of (2) and (3) we obtain $W \in L(C)$.

(b) Follows from (a) by transposition. □

The following theorem shows how to get corresponding transformation matrices.

Theorem 1: *Let $A \in \mathbb{Z}^{m,n}$ have rank r and SNF $C = UAV = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}$ with $D = Diag(d_1, \ldots, d_r)$ and transformation matrices $U \in GL_m(\mathbb{Z})$, $V \in GL_n(\mathbb{Z})$. Then arbitrary transformation matrices $U' \in GL_m(\mathbb{Z})$ and $V' \in GL_n(\mathbb{Z})$ can uniquely be written as WU with $W \in L(C)$ and VZ with $Z \in R(C)$, respectively. The transformation matrices WU and VZ are corresponding, if and only if $FDK = D$ for $F := (W_{i,j})_{1 \le i,j \le r}$ and $K := (Z_{i,j})_{1 \le i,j \le r}$.*

Proof: If $C = U'AV'$ for $U' \in GL_m(\mathbb{Z})$ and $V' \in GL_n(\mathbb{Z})$, then U' and V' have unique factorizations $U' = WU$ with $W := U'U^{-1}$ and $V' = V\,Z$ with $Z := V^{-1}V'$. Furthermore $WCZ = U'U^{-1}CV^{-1}V' = U'AV' = C$. Hence by Lemma 1, $W \in L(C)$, $Z \in R(C)$ and $FDK = D$. □

If we take one transformation matrix as fixed, we obtain with $Z = E_n$ and $W = E_m$:

Corollary 1: *Let $A \in \mathbb{Z}^{m,n}$ have rank r and SNF $C = UAV$ with transformation matrices $U \in GL_m(\mathbb{Z})$, $V \in GL_n(\mathbb{Z})$. Then*

(a) $U' \in GL_m(\mathbb{Z})$ *and* $C = U'AV$, *if and only if* $U' = \begin{pmatrix} E_r & * \\ 0 & G \end{pmatrix} \cdot U$ *for some* $G \in GL_{m-r}(\mathbb{Z})$.

(b) $V' \in GL_n(\mathbb{Z})$ *and* $C = UAV'$, *if and only if* $V' = V \cdot \begin{pmatrix} E_r & 0 \\ * & L \end{pmatrix}$ *for some* $L \in GL_{n-r}(\mathbb{Z})$.

Remark 2: *Corollary 1 shows the "degrees of freedom" we have in the choice of the corresponding transformation matrices.*

4. Reduction Algorithm for the Transformation Matrices

4.1. Simultaneous Reduction

The following lemma shows, how to obtain from corresponding transformation matrices U, V new corresponding transformation matrices, dependent on three parameters $k, s, t \in \mathbb{Z}$ with $1 \leq s \neq t \leq r$.

Lemma 2: *Let* $A \in \mathbb{Z}^{m,n}$ *have rank* r *and SNF* $\begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} = UAV$ *with* $D = Diag(d_1, \ldots, d_r)$ *and corresponding transformation matrices* $U \in GL_m(\mathbb{Z})$, $V \in GL_n(\mathbb{Z})$. *For integers* k, s, t *with* $1 \leq s \neq t \leq r$ *define*

$$q_{s,t}(k) := \begin{cases} k & \text{if } s < t \\ -k \cdot \frac{d_s}{d_t} & \text{if } s > t \end{cases}.$$

Then $(E_m + q_{s,t}(k) \cdot E_{s,t}) \cdot U \in GL_m(\mathbb{Z})$ *and* $V \cdot (E_n + q_{t,s}(k) \cdot E_{s,t}) \in GL_n(\mathbb{Z})$ *are corresponding transformation matrices for the SNF of* A.

Proof: Let $1 \leq s \neq t \leq r$. Clearly, $E_m + q_{s,t}(k) \cdot E_{s,t} \in GL_m(\mathbb{Z})$, $E_n + q_{t,s}(k) \cdot E_{s,t} \in GL_n(\mathbb{Z})$. Further we have:

$$\begin{aligned} &(E_m + q_{s,t}(k) \cdot E_{s,t}) \cdot U \cdot A \cdot V \cdot (E_n + q_{t,s}(k) \cdot E_{s,t}) \\ &\quad = (E_m + q_{s,t}(k) \cdot E_{s,t}) \cdot \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \cdot (E_n + q_{t,s}(k) \cdot E_{s,t}) \\ &\quad = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} + (q_{s,t}(k) \cdot d_t + q_{t,s}(k) \cdot d_s) \cdot E_{s,t} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\ &\quad = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

□

We want to reduce $\|U\|^2 + \|V\|^2$. With the assumption of Lemma 2 let $U = [u_1, \ldots, u_m]^T$, $V = [v_1, \ldots, v_n]$ and w.l.o.g. $s < t$. Left multiplication of U by $E_m + q_{s,t}(k) \cdot E_{s,t}$ and right multiplication of V by $E_n + q_{t,s}(k) \cdot E_{s,t}$ yield

$u_s = u_s + k \cdot u_t$ and $v_t = v_t - k \cdot \frac{d_t}{d_s} \cdot v_s$ and all other rows of U and all other columns of V remain unchanged.

Let s, t be fixed with $1 \leq s \neq t \leq r$. For reducing $\|U\|^2 + \|V\|^2$ we have to minimize the following function in the parameter k:

$$\begin{aligned} f(k) &= \|u_s + k \cdot u_t\|^2 + \|v_t - k \cdot \frac{d_t}{d_s} \cdot v_s\|^2 \\ &= \|u_s\|^2 + k^2\|u_t\|^2 + 2k < u_s, u_t > + \|v_t\|^2 + k^2 \frac{d_t^2}{d_s^2}\|v_s\|^2 - 2k \frac{d_t}{d_s} < v_s, v_t > \\ &= k^2\left(\|u_t\|^2 + \frac{d_t^2}{d_s^2}\|v_s\|^2\right) + k\left(2 < u_s, u_t > -2\frac{d_t}{d_s} < v_s, v_t >\right) + \|u_s\|^2 + \|v_t\|^2 \\ f'(k) &= \left(2\|u_t\|^2 + 2\frac{d_t^2}{d_s^2}\|v_s\|^2\right)k + 2 < u_s, u_t > -2\frac{d_t}{d_s} < v_s, v_t > \\ f''(k) &= 2\|u_t\|^2 + 2\frac{d_t^2}{d_s^2}\|v_s\|^2 > 0. \end{aligned}$$

So the absolute integral minimum is

$$k_1 = \left\lceil \frac{\frac{d_t}{d_s} < v_s, v_t > - < u_s, u_t >}{\|u_t\|^2 + \frac{d_t^2}{d_s^2}\|v_s\|^2} \right\rfloor.$$

In the case $s > t$ we get:

$$k_2 = \left\lceil \frac{\frac{d_s}{d_t} < u_s, u_t > - < v_s, v_t >}{\|v_s\|^2 + \frac{d_s^2}{d_t^2}\|u_t\|^2} \right\rfloor.$$

4.2. Reduction Algorithm

The algorithm displayed below reduces (already computed) corresponding transformation matrices U, V of a matrix $A \in \mathbb{Z}^{m,n}$ with rank r. It consists of two main parts: Steps 1 to 5 and steps 6 to 19. The first steps use lattice basis reduction techniques.

According to Corollary 1, for the first r rows of U and the first r columns of V we use *size reduction* [11], because for a lattice basis $b_1, \ldots, b_n$ and $i = 2, \ldots, n$, size reduction of b_i uses a unimodular transformation of the form $b_i^{\text{new}} := b_i^{\text{old}} + \sum_{j=1}^{i-1} t_j b_j$ with suitable $t_j \in \mathbb{Z}$. For the remaining $m - r$ rows of U and the remaining $n - r$ columns of V we use the more efficient *LLL reduction* [9] with parameter α ($\frac{1}{4} < \alpha \leq 1$) which includes size reduction steps and allows that the vectors are interchanged. We always use the parameter $\alpha = 1$.

The first steps of our algorithm are applicable only if $r < \max\{m, n\}$, compare Corollary 1. The remaining steps refer to the simultaneous reduction from Sect. 4.1.

Input $A \in \mathbb{Z}^{m,n}$ with rank r, $U = [u_1, \ldots, u_m]^T \in GL_m(\mathbb{Z})$,
$V = [v_1, \ldots, v_n] \in GL_n(\mathbb{Z})$ with $UAV = \mathrm{SNF}(A)$,
$d_1, \ldots, d_r$ elementary divisors

1 LLL reduce $u_{r+1}, \ldots, u_m$
2 LLL reduce $v_{r+1}, \ldots, v_n$
3 FOR $s = 1, \ldots, r$
4 Size reduce u_s in the lattice basis $u_{r+1}, \ldots, u_m, u_s$
5 Size reduce v_s in the lattice basis $v_{r+1}, \ldots, v_n, v_s$
6 smaller = FALSE
7 FOR $s = 1, \ldots, r$
8 FOR $t = 1, \ldots, r$, $t \neq s$
9 IF $s < t$
10 THEN $k = k_1$
11 $u_s = u_s + k \cdot u_t$
12 $v_t = v_t - k \cdot \dfrac{d_t}{d_s} \cdot v_s$
13 ELSE $k = k_2$
14 $u_s = u_s - k \cdot \dfrac{d_s}{d_t} \cdot u_t$
15 $v_t = v_t + k \cdot v_s$
16 IF $k \neq 0$
17 THEN smaller = TRUE
18 IF smaller
19 THEN GOTO 6

Output $U \in GL_m(\mathbb{Z}), V \in GL_n(\mathbb{Z})$ with $UAV = \mathrm{SNF}(A)$
and small $\|U\|^2 + \|V\|^2$

This algorithm reduces $\|U\|^2 + \|V\|^2$ in a finite number of steps, as we now show:

We can easily show that the steps 1, 3 and 4 correspond to left multiplication of U by a matrix of the form

$$\begin{pmatrix} E_r & * \\ 0 & G \end{pmatrix}$$

with $G \in GL_{m-r}(\mathbb{Z})$ and that the steps 2, 3 and 5 correspond to right multiplication of V by a matrix of the form

$$\begin{pmatrix} E_r & 0 \\ * & L \end{pmatrix}$$

with $L \in GL_{n-r}(\mathbb{Z})$.

The new matrices U, V after step 5 are corresponding transformation matrices for the SNF of A because of Lemma 1 and Theorem 1.

The steps 11, 12 and 14, 15 correspond to left multiplication of U by the matrix $E_m + q_{s,t}(k) \cdot E_{s,t}$ and right multiplication of V by the matrix $E_n + q_{t,s}(k) \cdot E_{s,t}$. According to Lemma 2, the new matrices U, V after step 19 are corresponding transformation matrices for the SNF of A.

If for all s, t with $1 \leq s \neq t \leq r$ k is zero, then $\|U\|^2 + \|V\|^2$ cannot be reduced any further. Since $\|U\|^2 + \|V\|^2$ is a natural number, the algorithm terminates after finitely many steps. □

Remark 3: *The main idea of steps 6 to 19 is to reduce one transformation matrix at the expense of the other in such a way that $\|U\|^2 + \|V\|^2$ decreases. This part of the algorithm should produce good results, if at the beginning one transformation matrix has considerably smaller entries than the other.*

4.3. Improvement of the Algorithm and Approximation Guarantees

Using results from lattice basis reduction we get:

Theorem 2: **(a)** *The square of the Euclidean norm of the last $m - r$ rows of the transformation matrix U is minimum up to a factor of $(\frac{4}{3})^{m-r-1}$. The same holds for the last $n - r$ columns of the transformation matrix V with a factor of $(\frac{4}{3})^{n-r-1}$.*

We can improve the algorithm so that the following holds:

(b) *The factor of a) can be improved to $\frac{m-r+3}{4}$ and $\frac{n-r+3}{4}$, respectively.*

(c) *Let the transformation matrix U be fixed. Then the square of the Euclidean norm of the first r columns of the transformation matrix V is minimum for this choice of U. The same holds for a fixed transformation matrix V and the first r rows of the transformation matrix U.*

Proof: (a) Follows directly from the fact that the square of the Euclidean norm of the i-th vector ($i = 1, \ldots, n$) of a LLL reduced lattice basis $b_1, \ldots, b_n$ with $\alpha = 1$ is minimum up to a factor of $(\frac{4}{3})^{n-1}$, see [9].

(b) Replace the LLL reduction by the HKZ reduction [7]. In [7], it is shown that the square of the Euclidean norm of the i-th vector ($i = 1, \ldots, n$) of a HKZ reduced lattice basis $b_1, \ldots, b_n$ is minimum up to a factor of $\frac{i+3}{4}$.

(c) In [11], for a lattice $L = L(b_1, \ldots, b_n)$ an algorithm SHORTEST-ENUM $(b_1, \ldots, b_n)$ is introduced which finds a shortest lattice vector $b \in L \setminus \{0\}$ by complete enumeration. This algorithm can easily be transformed to an algorithm CLOSEST-ENUM $(v, b_1, \ldots, b_n)$ for finding a closest lattice vector $b \in L$ to a vector v.

We add the following steps to the reduction algorithm:

20 FOR $s = 1, \ldots, r$

21 $u_s = u_s -$ CLOSEST–ENUM $(u_s, u_{r+1}, \ldots, u_m)$

22 $v_s = v_s -$ CLOSEST–ENUM $(v_s, v_{r+1}, \ldots, v_n)$

The assertion follows from Corollary 1. □

Remark 4: *Note that the improved algorithms of (b) and (c) are not polynomial any more but terminate in reasonable time. Numerical experiments show that steps 20 to 22 are of little practical value; they were omitted in the computations reported in Sect. 5.*

5. Experimental Results

5.1. Test Matrices

In this section, we consider the following class of matrices:

- $A_n = (a_{s,t} = s^3 \cdot t^2 + s + t) \;\; for 1 \le s, t \le n.$

These matrices have rank 3.

- $B_n = (b_{s,t} = (s-1)^{t-1} \bmod n) \;\; for 1 \le s, t \le n.$

These matrices have full rank, if n is a prime. In general the more prime factors n has the smaller is the rank.

- $C_n = \left(c_{s,t} = \begin{cases} 0 & \text{for } s < t \\ s & \text{for } s = t \\ st & \text{for } s > t \end{cases} \right) \;\; \text{for } 1 \le s, t \le n.$

  These matrices are lower triangular matrices and have full rank.

Moreover, we use the following matrices from literature:

- D_6 with rank 6 [14, p. 133].
- $E_{5,16}$ with rank 5 [3, p. 159].
- F_{18} with rank 14 [12, p. 379].
- G_{20} with rank 20 [2, p. 1069].
- $H_{26,27}$ with rank 25 [4, p. 406].
- $I_{78,77}$ with rank 77 [15, p. 71].
- J_{140} with rank 140 [15, p. 71].

5.2. Implementation

The reduction algorithm of Sect. 4.2 was implemented in C^{++} and compiler *ibmcxx*, version 3.6.4.0. The experiments were made under AIX 4.3.3 on a POWER3-II 375 MHz processor of a IBM RS/6000 SP-SMP computer with main memory 1024 MB.

With the Kannan-Bachem algorithm [6], we compute the SNF of the test matrices and corresponding transformation matrices U, V. In comparison with other SNF algorithms this algorithm produces the smallest transformation matrices, even smaller than those produced by the algorithm of Storjohann [14]. For the example matrix D_6 of [14], Storjohann's algorithm produces transformation matrices U, V with $\left\lceil \log_2\left(\|U\|^2 + \|V\|^2\right)\right\rceil = 74$, whereas the Kannan-Bachem algorithm produces such with $\left\lceil \log_2\left(\|U\|^2 + \|V\|^2\right)\right\rceil = 44$, compare the table in Sect. 5.5.

We test three classes of matrices: with small, medium and large rank. If we consider a matrix of full rank and one fixed transformation matrix, then the other transformation matrix is uniquely determined because of Corollary 1. In this case the steps 1 to 5 of the reduction algorithm are not executed.

In practice the steps 6 to 19 do not terminate in reasonable time for larger matrices with large rank. Because of that we use the following improvements. On the one hand we favor the choice of s, t for which $\|u_s\|^2$ and $\|v_t\|^2$ are large. Furthermore we leave out some reduction steps, if $\|U\|^2 + \|V\|^2$ decreases little, but $\|U\|^2$ or $\|V\|^2$ increases considerably. Finally we stop this part of the algorithm after a pre-assigned number of steps.

As the numbers $\|U\|^2 + \|V\|^2$ are very large in most cases, we give the bits of this numbers.

5.3. Tests with Small Rank

Matrix	Rank	Bits ($\|U\|^2 + \|V\|^2$)		Time
		old	new	
A_{50}	3	36	10	00:00:06
A_{100}	3	43	12	00:00:43
A_{150}	3	47	13	00:02:22
A_{200}	3	50	13	00:05:43
A_{250}	3	53	14	00:11:17
A_{300}	3	54	14	00:19:37
A_{350}	3	56	15	00:31:19
A_{400}	3	57	15	00:47:01
A_{450}	3	59	15	01:07:15
A_{500}	3	60	16	01:32:45

The entries are clearly reduced by the reduction algorithm. The bits of the input transformation matrix increase more than the bits of the output transformation matrix, i.e., the algorithm behaves better for larger matrices. The additional steps

6 to 19 are not important, as they are applied only to the first 3 row vectors of U and the first 3 column vectors of V.

5.4. Tests with Medium Rank

Matrix	Rank	Bits ($\|U\|^2 + \|V\|^2$)		Time
		old	new	
B_{50}	19	59	16	00:00:05
B_{100}	22	39	24	00:00:58
B_{148}	38	4663	147	00:06:41
B_{201}	67	18822	400	02:54:47
B_{248}	33	2260	156	00:34:45
B_{298}	149	5737	862	08:44:40
B_{345}	45	1636	142	02:33:20
B_{396}	32	419	28	03:00:15
B_{450}	57	206	21	04:00:19
B_{496}	62	4694	189	11:50:42

We get better results than for small rank and can considerably reduce the transformation matrices which have very large entries. The LLL reduction consumes most of the execution time.

5.5. Tests with Large Rank

Matrix	Rank	Bits ($\|U\|^2 + \|V\|^2$)		Time
		old	new	
D_6	6	44	039	0:00:00
$E_{5,16}$	5	125	12	00:00:00
F_{18}	14	140	30	00:00:01
G_{20}	20	243	164	00:00:05
$H_{26,27}$	25	73	35	00:00:04
$I_{78,77}$	77	1532	753	00:09:18
J_{140}	140	2008	1339	00:34:43
B_{53}	53	416	312	00:03:55
B_{101}	101	1042	843	00:17:45
B_{149}	149	1889	1503	00:39:28
B_{199}	199	2723	2569	02:20:59
B_{251}	251	3799	3246	02:34:05
B_{293}	293	4998	4964	07:59:31
B_{349}	349	4895	4149	08:59:50
B_{401}	401	5629	5345	19:46:23
C_{50}	50	1524	1379	00:16:07
C_{100}	100	4596	2686	01:11:30
C_{150}	150	9571	5176	06:14:28
C_{200}	200	16097	7843	17:43:22
C_{250}	250	22283	9592	25:38:42
C_{300}	300	31792	22934	29:23:56
C_{350}	350	40324	27382	44:11:10
C_{400}	400	51250	29698	107:12:14

We still get a reduction, but less than for medium rank matrices. This is easily explained by Corollary 1: The set of transformation matrices gets smaller as the rank gets larger.

On average the bits of the matrices C_n are nearly halved, whereas for the matrices B_n the bits are reduced less. The reason for this is that due to their structure the matrices C_n are transformed to diagonal form by the first step of the Kannan Bachem Algorithm using only unimodular column operations, and so the transformation matrix V has considerably larger entries than the transformation matrix U. For the matrices B_n the entries of the transformation matrices U and V have nearly the same size. As the steps 6 to 19 always reduce one transformation matrix at the expense of the other, the algorithm produces better results for the matrices C_n, compare Remark 3. We can drastically decrease the large execution times for the matrices C_n by terminating the algorithm prematurely; in most cases $\|U\|^2 + \|V\|^2$ is only slightly larger.

References

[1] Gruber, P. M., Lekkerkerker, C. G.: Geometry of numbers. Amsterdam: North-Holland 1987.
[2] Hafner, J. L., McCurley, K. S.: Asymptotically fast triangularization of matrices over rings. SIAM J. Comput. *20(6)*: 1068–1083 (1991).
[3] Havas, G., Holt, D. F., Rees, S.: Recognizing badly presented $\mathbb{Z}$-modules. Linear Algebra Appl. *192*, 137–163 (1993).
[4] Havas, G., Majewski, B. S.: Integer matrix diagonalization. J. Symbolic Comput. *24*, 399–408 (1997).
[5] Havas, G., Majewski, B. S., Matthews, K. R.: Extended gcd and Hermite normal form algorithms via lattice basis reduction. Exp. Math. *7*, 125–136 (1998).
[6] Kannan, R., Bachem, A.: Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. SIAM J. Comput. *8(4)*, 499–507 (1979).
[7] Lagarias, J. C., Lenstra, H. W., Schnorr, C. P.: Korkin-zolotarev bases and successive minima of a lattice and its reciprocal lattice. Combinatorica *10*, 333–348 (1990).
[8] Lazebnik, F.: On systems of linear diophantine equations. Math. Mag. *69(4)*, 261–266 (1996).
[9] Lenstra, A.K., Lenstra, H.W., Jr., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. *261*, 515–534 (1982).
[10] MAGMA HTML Help Document, V2.11 (May 2004).
Available at "http://magma.maths.usyd.edu.au/magma/htmlhelp/MAGMA.htm".
[11] Schnorr, C. P., Euchner, M.: Lattice basis reduction: Improved practical algorithms for solving subset sum problems. Math. Programming *66*, 181–199 (1994).
[12] Sims, C. C.: Computation with finitely presented groups. Cambridge University Press 1994.
[13] Smith, H. J. S.: On systems of linear indeterminate equations and congruences. Philos. Trans. Royal Soc. London *151*, 293–326 (1861).
[14] Storjohann, A.: Algorithms for matrix canonical forms. Ph.D. thesis, ETH Zürich 2000.
[15] Wagner, C.: Normalformenberechnung von Matrizen über euklidischen Ringen. Ph.D. thesis, Institut für Experimentelle Mathematik, Universität/GH Essen 1997.

G. Jäger
Mathematisches Seminar, Bereich 2
Christian-Albrechts-Platz 4
24118 Kiel
Germany
e-mail: gej@numerik.uni-kiel.de

Verleger: Springer-Verlag GmbH, Sachsenplatz 4–6, 1201 Wien, Austria. – Herausgeber: Prof. Dr. Wolfgang Hackbusch, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstraße 22–26, 04103 Leipzig, Germany. – Satz und Umbruch: Scientific Publishing Services (P) Ltd., Chennai, India. – Offsetdruck: Manz Crossmedia, 1051 Wien, Austria. – Verlagsort: Wien. – Herstellungsort: Wien. – Printed in Austria.