Computing Hermite and Smith normal forms of triangular integer matrices

Technical Report No. 256, Department of Computer Science, ETH Zurich, December 1996.

Computing Hermite and Smith Normal Forms of Triangular Integer Matrices*

Arne Storjohann
Institut für Wissenschaftliches Rechnen
ETH Zürich, Switzerland
storjoha@inf.ethz.ch

Abstract

This paper considers the problem of transforming a *triangular* integer input matrix to canonical Hermite and Smith normal form. We provide algorithms and prove deterministic running times for both transformation problems that are linear (hence optimal) in the matrix dimension. The algorithms are easily implemented, assume standard integer multiplication, and admit excellent performance in practice. The results presented here lead to faster practical algorithms for computing the Hermite and Smith normal form of an arbitrary (non triangular) integer input matrix.

1 Introduction

It follows from Hermite [Her51] that any $m \times n$ rank n integer matrix A can be transformed using a sequence of integer row operations to an upper triangular matrix H that has j-th diagonal entry h_j positive for $1 \leq j \leq n$ and off-diagonal entries $\bar{h}_{ij}$ satisfying $0 \leq \bar{h}_{ij} < h_j$ for $1 \leq i < j \leq n$. The matrix H — called the Hermite normal form of A — always exists and is unique. In this paper we consider the problem of computing the Hermite normal form of an $n \times n$ nonsingular integer input matrix T that is already upper triangular and has off-diagonal entries bounded in magnitude by the product $D = |h_1 h_2 \cdots h_n|$ of the diagonal entries. We get the following transformation diagram.

$$\overset{T}{\begin{bmatrix} h_1 & t_{12} & t_{13} & & t_{1n} \\ & h_2 & t_{23} & \cdots & t_{2n} \\ & & h_3 & & t_{3n} \\ & & & \ddots & \vdots \\ & & & & h_n \end{bmatrix}} \quad \rightsquigarrow \quad \overset{H}{\begin{bmatrix} h_1 & \bar{h}_{12} & \bar{h}_{13} & & \bar{h}_{1n} \\ & h_2 & \bar{h}_{23} & \cdots & \bar{h}_{2n} \\ & & h_3 & & \bar{h}_{3n} \\ & & & \ddots & \vdots \\ & & & & h_n \end{bmatrix}}$$

The problem is to transform T to Hermite normal form H using a sequence of integer row operations where an integer row operation is one of: adding an integer multiple of one row to a different row; switching two rows; negating a row. The Hermite normal form has some important advantages over a general triangularization. First, the total size of H (the sum of the bit lengths of the individual entries) is bounded by $O(n \log D)$ bits whereas T

*This work has been supported by grants from the Swiss Federal Office for Education and Science in conjunction with partial support by ESPRIT LTR Project no. 20244 — ALCOM–IT.

requires $O(n^2 \log D)$ bits to write down. Second, to determine if two integer matrices are row equivalent (each is transformable to the other by applying a sequence of integer row operations) it is sufficient to compare their canonical Hermite normal forms. This check for row equivalence is not possible with a general (non unique) triangularization.

Previous Hermite normal form algorithms of Domich, Kannan & Trotter [DKT87], Iliopoulos [Ili89] and Hafner & McCurley [HM91] all require $O(n^3)$ arithmetic operations with integers bounded in length by $O(\log D)$ bits to reduce T to H. A single arithmetic operation with integers bounded in length by $O(\log D)$ bits costs $O((\log D)^{1+\epsilon})$ bit operations where $\epsilon \in (0, 1]$ depending on the algorithm used for integer multiplication. Standard integer multiplication has $\epsilon = 1$ whereas the asymptotically fast (but currently impractical) algorithm of Schönhage-Strassen [SS71] allows any fixed ϵ with $\epsilon > 0$. More recently, Storjohann & Labahn [SL96] have given a fast matrix multiplication decomposition which reduces the complexity of computing H from T to $O(n^\theta)$ arithmetic operations with integers of the same length. Here, θ is the exponent for matrix multiplication: two $n \times n$ matrices over a ring can be multiplied in $O(n^\theta)$ ring operations. The standard algorithm has $\theta = 3$ and the current record is $\theta < 2.376$ due to Coppersmith & Winograd [CW90]. Thus, the previously fastest algorithm to compute H from T requires $O(n^{2.376}(\log D)^{1+\epsilon})$ bit operations assuming asymptotically fast (but currently impractical) matrix multiplication. The algorithm we give here computes H from T in only $O(n^2 \log^2 D)$ bit operations — this is now absolutely optimal in n being a factor of only $O(\log D)$ more than the number of bits required to write down the input matrix. Moreover, the algorithm assumes the standard, practical algorithms for integer and matrix multiplication and admits excellent performance in practice.

The second problem we consider is integer matrix diagonalization. It follows from Smith [Smi61] that any $n \times m$ integer matrix A can be transformed using a sequence of integer row *and* column operations to the unique diagonal matrix $S = \mathrm{diag}(s_1, \ldots, s_r, 0, \ldots, 0)$ where r is the rank of A, each s_i positive, and s_i divides s_{i+1} for $1 \leq i \leq r-1$. The matrix S — called the Smith normal form of A — always exists and is unique. In this paper we consider the problem of transforming to Smith normal form an $n \times m$ integer matrix T which has principal $n \times n$ submatrix nonsingular upper triangular and off-diagonal entries bounded in magnitude by the product D of the diagonal entries. We get the following transformation diagram.

$$\overset{T}{\begin{bmatrix} t_1 & t_{12} & t_{13} & & t_{1n} & t_{1\,n+1} & & t_{1m} \\ & t_2 & t_{23} & \cdots & t_{2n} & t_{2\,n+1} & \cdots & t_{2m} \\ & & t_3 & & t_{3n} & t_{3\,n+1} & & t_{3m} \\ & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & t_n & t_{n\,n+1} & & t_{nm} \end{bmatrix}} \rightsquigarrow \overset{S}{\begin{bmatrix} s_1 & & & & & \\ & s_2 & & & & \\ & & s_3 & & & \\ & & & \ddots & & \\ & & & & s_n & \end{bmatrix}} \quad (1)$$

In this case the problem is to transform T to Smith normal form S using a sequence of integer row and column operations. Note that the total size of T is $O(nm \log D)$ bits. The problem of computing the Smith normal form of an already triangular input matrix T is interesting in it's own right but we wish to emphasize that this problem is not contrived. For example, both the asymptotically fast Smith normal form algorithm given in [Sto96b] and the fast practical algorithm given in Section 4 of this paper assume that the input matrix can be written as T of (1); the case of an arbitrary (i.e. non triangular) integer input matrix A is handled by first triangularizing.

For brevity in summarizing previous complexity results let us consider here the case of an input matrix A that is square nonsingular, that is, with $n = m$. Let us also assume that

have the quantity D (which will be $|\det A|$) and that entries in A are bounded in magnitude by D. Hafner & McCurley's [HM91] asymptotically fast triangularization algorithm requires $O(n^\theta)$ arithmetic operations with integers bounded in length by $O(\log D)$ bits to produce a triangularization T of A but their algorithm for computing S from T (or directly from A) requires $O(n^3 \log D)$ arithmetic operations with integers of the same length — note the extra $O(\log D)$ factor in this complexity result. Giesbrecht [Gie95] has given a Las Vegas probabilistic algorithm that requires an expected number of $O(n^3)$ arithmetic operations with integers of the same length — in essence removing this offending $O(\log D)$ factor. Most recently, Storjohann [Sto96b] has given a deterministic algorithm that requires $O(n^\theta)$ arithmetic operations with integers bounded in length by $O(\log D)$ bits. This last complexity result may appear to be near-optimal for the problem of transforming T to S as well as for the more general problem of transforming A to S but this assessment is false. In this paper we prove the surprising result that the Smith normal form of T can be computed in only $O(n^2 \log^2 D)$ bit operations using standard integer arithmetic — this is now optimal in the matrix dimension, being a factor of only $O(\log D)$ more than the size of the input matrix. In the case where $m > n$ the complexity becomes $O(nm \log^2 D)$ — also optimal in the matrix dimension. This algorithm is easy to implement in a Computer Algebra system and runs extremely fast in practice.

We also apply our triangular Smith normal form algorithm to get a fast+practical+ deterministic algorithm for computing the Smith normal form of an arbitrary (i.e. non triangular) $n \times m$ input matrix A. The cost of the algorithm is $O^\sim(nm^3 \log^2 m||A|| + m^4 \log^3 m||A||)$ bit operations where $||A|| = \max|A_{ij}|$. For comparison, a direct application of the near-optimal algorithm in [Sto96b] under the assumption of standard integer and matrix arithmetic calls for $O(nm^4 \log^2 m||A||)$ bit operations. We have improved this in a practical sense by about a factor of $O(m)$. The only other algorithm that also provides a factor of $O(m)$ improvement in practice is the Monte Carlo probabilistic algorithm of Giesbrecht [Gie95]. Giesbrecht's algorithm requires $O^\sim(nm^3 \log||A|| + m^3 \log^2||A||)$ bit operations using standard[1] matrix and integer arithmetic to return the Smith normal form with an exponentially small probability of error. Giesbrecht has presented an even faster Monte Carlo probabilistic algorithm that works for sparse input matrices [Gie96]. The disadvantage of the algorithms in [Gie95, Gie96] is that they requires randomization and may return an incorrect result which is difficult to detect. An important advantage of the algorithms in [Gie95, Gie96], though, is that they admit near-optimal space complexity. The Smith normal form algorithm we present here, like previous algorithms [Ili89, HM91, Sto96b], requires about about a factor of $O(m)$ more space than is required to write down the input matrix. Since many input matrices arising in practice are large, sparse and with small entries, a significant open problem is to find a fast deterministic[2] algorithm with near-optimal space complexity for computing the Smith normal form of a sparse integer matrix.

The rest of this paper is organised as follows. In Sections 2 and 3 we we present our algorithm for transforming an upper triangular input matrix to Hermite and Smith normal form respectively. In Section 4 we give an algorithm for computing the Smith normal form of an arbitrary integer input matrix. Before continuing we define the complexity model used for the analysis of algorithms in this paper.

[1] Giesbrecht also shows how to incorporate fast matrix multiplication techniques into his algorithm.

[2] A Las Vegas probabilistic (i.e. correctness is guaranteed) algorithm for sparse integer matrices would already be a significant breakthrough.

Complexity Model The number of bits in the binary representation of an integer a is given by

$$\lg a = \begin{cases} 1, & \text{if } a = 0; \\ 1 + \lfloor \log_2 |a| \rfloor, & \text{if } a > 0. \end{cases}$$

Using standard arithmetic, a and b can be multiplied in $O((\lg a)(\lg b))$ bit operations, and we can express $a = qb + r$, with $0 \leq |r| < |b|$, in $O((\lg a/b)(\lg b))$ bit operations. We can compute $g = \gcd(a, b)$ in $O((\lg a)(\lg b))$ bit operations; in the same running time we can recover s and t satisfying $sa + tb = g$ with $s \leq |b/g|$ and $|t| \leq |a/b|$. Finally, each direction of the isomorphism implied by the Chinese remainder algorithm can be computed in $O((\lg N)^2)$ bit operations if N is the product of all the moduli. This complexity model was popularized by Collins [Col68] and has been dubbed "naive bit complexity" (see, for example, Bach & Shallit [BS96]).

2 Hermite Normal Forms of Triangular Matrices

Work on this problem was motivated by an asymptotically fast triangularization algorithm of Hafner & McCurley [HM91] that takes as input an $m \times n$ rank n input matrix and returns as output a row equivalent upper triangular matrix with all entries nonnegative and with off-diagonal entries bounded in magnitude by the product D of the diagonal entries — the matrix T of (2) is an example of the principal $n \times n$ submatrix of such a triangularization.

$$\underset{T}{\begin{bmatrix} 8 & 11286 & 4555 & 46515 & 83732 \\ & 1 & 66359 & 153094 & 47580 \\ & & 9 & 43651 & 201929 \\ & & & 77 & 129116 \\ & & & & 50 \end{bmatrix}} \rightsquigarrow \underset{H}{\begin{bmatrix} 8 & 0 & 1 & 51 & 42 \\ & 1 & 2 & 20 & 5 \\ & & 9 & 69 & 23 \\ & & & 77 & 16 \\ & & & & 50 \end{bmatrix}} \quad (2)$$

The classical algorithm for transforming T to Hermite normal form works by placing, for $r = 1, 2, \ldots, n$, the principal r-th minor of T into correct form. At stage r appropriate multiples of the r-th row of T are added to rows $1, 2, \ldots, r-1$ to reduce the entries above the diagonal in column r modulo the diagonal entry.

Classical Triangular Reduction

for $r = 1$ to n do
  for $i = 1$ to $r - 1$ do
    $\text{row}(T, i) = \text{row}(T, i) - \lfloor T_{i,r}/T_{r,r} \rfloor \text{row}(T, r)$;

The classical approach requires $O(n^3)$ arithmetic operations but does not properly bound the magnitudes of intermediate integer entries. The Hermite normal form algorithms given in [DKT87, Ili89, HM91] essentially follow the classical approach but perform all integer operations modulo D; this leads directly to a complexity of $O(n^3)$ arithmetic operations with integers bounded in length by $O(\log D)$ bits. Fast matrix multiplication techniques can be used to reduce the complexity to $O(n^\theta)$ ring operations with integers of the same length [SL96]. At first sight this appears to be near-optimal in the parameter n since

the problem seems tantamount to performing a "matrix operation"[3] on an $n \times n$ matrix. This assessment is false. The algorithm we give here computes H in only $O(n^2 \log^2 D)$ bit operations using standard arithmetic.

The algorithm we propose differs from the classical approach primarily with respect to the order of operations. Our algorithm works by putting, for $r = n, n-1, \ldots, 1$, the *trailing* $(n-r)$-th minor of T into correct form. At stage r, appropriate multiples of rows $r+1, r+2, \ldots, n$ of T are added to row r to reduce the off-diagonal entries in row r modulo the diagonal entry in the same column.

New Triangular Reduction

for $r = n$ by -1 to 1 do
  for $j = r+1$ to n do
    $\mathrm{row}(T, r) = \mathrm{row}(T, r) - \lfloor T_{r,j}/T_{j,j} \rfloor \mathrm{row}(T, j)$;

This new reduction order, coupled with a careful analysis of the integer operations occurring during the reduction, enables us to prove a complexity of only $O(n^2 \log^2 D)$ bit operations using standard integer arithmetic. To get this complexity result, we need two refinements to the algorithm. First, all integer operations must be performed modulo d for some integer d bounded by D.

Lemma 1 *Let T be a $k \times k$ nonsingular upper triangular integral matrix and let $d = \prod_{2 \le i \le k} |T_{i,i}|$. For j with $2 \le j \le k$, the matrix T_d obtained from T by reducing the entry in row 1 column j modulo d is row equivalent to T.*

Proof. Let T_1 be the trailing $(k-1) \times (k-1)$ submatrix of T. We claim that the k-dimensional row vector with all entries zero except with j-th entry equal to $\det T_1$ can be expressed as an integer linear combination of the last $k-1$ rows of T — to see this, note that $T_1^{\mathsf{adj}} T_1 = \det(T_1) I_{k-1}$ where T_1^{adj}, the adjoint of T_1, is an integer matrix. Since $d = \pm \det T_1$, there exists some integer linear combination of the last $k-1$ rows of T which, when added to row 1 of T, have the desired effect of reducing the entry in column j modulo d but leaves the other entries unchanged. ∎

For the second refinement, we note that some row operations may be sparse. At stage r of the algorithm, rows $r+1, r+2, \ldots, n$ of T are already in Hermite normal form. Thus, for $r+1 \le i < j \le n$, entry $T_{i,j}$ will satisfy $0 \le T_{i,j} < T_{j,j}$. When $T_{j,j} = 1$ the entry $T_{i,j}$ will be zero and a row operation that adds a multiple of row i to another row can ignore column j. The following algorithm for one stage of the reduction includes both refinements.

Algorithm: `TriangularReduction`
Input: A $k \times k$ upper triangular rank k integral matrix T with trailing $(k-1)$-th minor in Hermite normal form.
Output: The Hermit normal form of T. [T is transformed in place.]

(1) [Initialize:]
  $d \leftarrow \prod_{2 \le i \le k} T_{i,i}$;
  if $T_{1,1} < 0$ then $\mathrm{row}(T, 1) \leftarrow -\mathrm{row}(T, 1)$;
  for $j = 2$ to k do $T_{1,j} \leftarrow \mathrm{mod}(T_{1,j}, d)$;

[3] Note that the classical reduction algorithm performs $O(n^2)$ row operations on an $n \times n$ matrix — the same as required by the classical algorithms for Gaussian elimination, determinant computation, inversion, etc.

(2) [Reduce off-diagonal entries in row 1:]
$L \leftarrow \{j \mid 2 \leq j \leq k, T_{i,i} > 1\}$;
for $j = 2$ to k do
$L \leftarrow L \backslash \{j\}$;
(a) $(q, r) \leftarrow$ a solution to $T_{1,j} = qT_{j,j} + r$ with $0 \leq r < T_{j,j}$;
$T_{1,j} \leftarrow r$;
for $l \in L$ do
(b) $T_{1,l} \leftarrow \mathrm{mod}(T_{1,l} - qT_{j,l}, d)$;

Lemma 2 *Algorithm* `TriangularReduction` *is correct. If both d and $||T||$ are bounded by D, then the running time of the algorithm is $O(n \log^2 D)$ bit operations.*

Proof. We first prove correctness. First note that the only entries of T which are modified during the algorithm are the off-diagonal entries of row 1. Step (1) applies an integer row operation to T, if required, that ensures that entry $T_{1,1}$ is positive. By the choice of (q, r) in step (2), the output matrix will be in Hermite normal form. The fact that line (b) is essentially performing integer row operations on T follows from Lemma 1. We now bound the running time. The cost of step (1) is bounded by $O(k \log^2 D)$ bit operations. Note that throughout step (2) all entries in the work matrix satisfy $0 \leq T_{i,j} \leq d$. A single execution of statement (a) requires $O((\lg T_{1,j}/T_{j,j})(\lg T_{j,j}))$ bit operations. Since $\lg T_{1,j}/T_{j,j} < \lg d$, we can bound this more simply by $O((\lg d)(\lg T_{j,j}))$ bit operations. The cost of a single execution of statement (b) is bounded by $O((\lg q)(\lg T_{j,l}) + (\lg qT_{j,l}/d)(\lg d))$ bit operations. Again, replacing q by d leads to the simpler bound $O((\lg d)(\lg T_{j,l}))$ bit operations. This shows that the cost of a single execution of statement (a) and (b) is bounded by $c(\lg d)(\lg T_{j,j})$ and $c(\lg d)(\lg T_{j,l})$ respectively for some absolute constant c. The total cost for one pass of the outer loop in step (3) is given by

$$\begin{aligned}
c(\lg d)(\lg T_{j,j}) + \sum_{l \in L} c(\lg d)(\lg T_{j,l}) &\leq c(\lg d)(\lg T_{j,j} + \sum_{l \in L} \lg T_{j,l}) \\
&\leq c(1 + \log_2 d)(1 + \log_2 T_{j,j} + \sum_{l \in L} (1 + \log_2 T_{j,l})) \\
&\leq c(1 + \log_2 d)(1 + \log_2 T_{j,j} + 2 \sum_{l \in L} \log_2 T_{j,l}) \\
&\leq c(1 + \log_2 d)(1 + 2 \log_2 d)
\end{aligned}$$

Thus, one pass of the loop requires $O(\log^2 d)$ bit operations. Since the loop is repeated $k - 1$ times, and $d|D$, all of step (3) can be accomplished in $O(k \log^2 D)$ bit operations. ■

Theorem 3 *There exists a deterministic algorithm that receives as input an $n \times n$ rank n upper triangular matrix T and returns the Hermite normal form of T. If both $|\det(T)|$ and $||T||$ are bounded by D, then the running time of the algorithm is $O(n^2 \log^2 D)$ bit operations.*

Proof. Apply in place, for $k = 1, 2, \ldots, n$, algorithm `TriangularReduction` to the trailing $k \times k$ minor of T. ■

3 Smith Normal Forms of Triangular Matrices

In this section we assume we start with an $n \times m$ integer input matrix T having principal $n \times n$ minor nonsingular upper triangular and all entries bounded in magnitude by the product D of the diagonal entries. Equation (3) gives an example of a 5×7 input matrix with $D = 226944$.

$$T = \begin{bmatrix} 2 & 226940 & 226916 & 16 & 226902 & 226942 & 6 \\ & 3 & 84 & 226900 & 150 & 11 & 226932 \\ & & 2 & 210 & 102 & 226900 & 226684 \\ & & & 2 & 222912 & 223924 & 220452 \\ & & & & 9456 & 7080 & 15216 \end{bmatrix} \quad \rightsquigarrow \quad S = \begin{bmatrix} 1 & & & & & & \\ & 2 & & & & & \\ & & 2 & & & & \\ & & & 6 & & & \\ & & & & 24 & & \end{bmatrix} \tag{3}$$

The goal is to transform T to Smith normal form S by applying a sequence of integer row and column operations. The classical approach for diagonalizing integer matrices using alternating row and column operations does not exploit the triangular structure during the reduction when applied to a triangular input matrix. Algorithms based on the classical approach but which perform arithmetic modulo D still require (worst case) on the order of $O(nm^2 \log D)$ arithmetic operations with integers bounded in length by $O(\log D)$ bits to compute S from T [Ili89, HM91]. [Sto96b] The near-optimal algorithm given in [Sto96b] for diagonalizing matrices over the ring of integers modulo D can be applied to the problem at hand and requires $O(nm^{\theta-1})$ arithmetic operations with integers bounded in length by $O(\log D)$ bits; even assuming the current record on θ this becomes $O(nm^{1.376}(\log D)^{1+\epsilon})$ bit operations. The algorithm we propose here requires only $O(nm \log^2 D)$ bit operations using standard integer and matrix multiplication.

Before presenting the general algorithm in Subsection 3.3, we present two key subroutines separately in Subsections 3.1 and 3.2. We will require the following simple number theoretic algorithm.

Theorem 4 (Bach [Bac92]) *Let a, b, N be integers with N positive and $\gcd(a, b, N) = 1$. There exists a deterministic algorithm that takes as input a, b and N and returns as output an integer c with $0 \leq c < N$ and such that $\gcd(a + cb, N) = 1$. If $|a|, |b| \leq N$ then the cost of the algorithm is $O(\log^2 N)$ bit operations assuming standard integer arithmetic.*

Proof. The algorithm and proof are due to Bach [Bac92]. The algorithm can be expressed as follows:

1. if $\gcd(a, N) = 1$ then
2. $\quad c \leftarrow 0$
3. elif $\gcd(a + b, N) = 1$ then
4. $\quad c \leftarrow 1$
5. else
6. $\quad g \leftarrow \gcd(a, N)$;
7. $\quad (N', N'') \leftarrow$ a factorization of N with $\gcd(N', N'') = 1$ and $\gcd(g, N'') = 1$;
8. $\quad c \leftarrow$ an integer with $0 < c < N$ and $c \equiv 1 \bmod N'$ and $c \equiv 0 \bmod N''$;
9. fi;

The values returned for c in lines 2 and 4 are clearly correct. If $\gcd(a, N) > 1$ then g must contain at least one prime divisor of N. If $\gcd(a + b, N) > 1$ as well, then some

prime divisor of N must be excluded from a (otherwise $\gcd(b, N) = 1$ and we would have $\gcd(a + b, N) = 1$). The factorization $N'N''$ is obtained by applying a factor refinement algorithm to $g \cdot (N/g)$. The factor N' will contain those prime of N that are common to a while the factor N'' those primes of N that are not common to a — this reduces the problem to the two cases in lines 2 and 4. Line 7 can accomplished in $O(\log^2 N)$ bit operations (see Bach, Driscoll & Shallit [BDS93] or Bach & Shallit [BS96]) and line 8 is accomplished in the same time using the Chinese remainder algorithm. ■

3.1 Phase One Subroutines

Let T be a $k \times m$ rank k upper triangular matrix with first $k-1$ columns in Smith normal form. We can write T as

$$T = \left[\begin{array}{ccccc|cccc} a_1 & & & & t_1 & * & * & \cdots & * \\ & a_2 & & & t_2 & * & * & \cdots & * \\ & & \ddots & & & \vdots & & & \\ & & & a_{k-1} & t_{k-1} & & & & \\ & & & & t_k & * & * & \cdots & * \end{array}\right] \tag{4}$$

The purpose of this section is to prove the following result.

Theorem 5 *Let T be as in (4) and satisfy the conditions*

1. *first k columns of T have rank k,*
2. *first $k-1$ columns of T are in Smith normal form,*
3. *off-diagonal entries in rows $1, 2, \ldots, k-1$ are reduced modulo the diagonal entry in the same column,*
4. *off-diagonal entries in row k are bounded in magnitude by D, a positive multiple of the determinant of the principal k-th minor of T.*

There exists a deterministic algorithm that applies unimodular row and column operations to transform the k-th principal minor of T to Smith normal form. If the input matrix satisfies $k = 1$ or $a_1 > 1$, then the running time of the algorithm is $O(m \log^2 D)$ bit operations.

We first prove some intermediate results.

Lemma 6 *Let T be as in (4) with $k > 1$ and satisfy the 4 conditions of Theorem 5. There exists a deterministic algorithm that transforms T to an equivalent matrix that satisfies the same conditions but with the addition of*

5. $\gcd(a_i, t_i) = \gcd(a_i, t_i, t_{i+1}, \ldots, t_k)$ *for* $1 \le i \le k-1$,

If the input matrix satisfies $a_1 > 1$, then the running time of the subroutine is bounded by $O(m \log^2 D)$ bit operations.

Proof. The algorithm is inductive. Note that the input matrix T satisfies

$$\gcd(T_{r,r}, T_{r,k}) = \gcd(T_{r,r}, T_{r,k}, T_{r+1,k}, \ldots, T_{k,k}) \tag{5}$$

for $r = k$. For some i, $1 \le i \le k$, assume that T satisfies (5) for $r = k, k-1, \ldots, i-1$. We show how to apply unimodular row and column operations to transform T to an equivalent matrix that satisfies conditions 1 through 4 and (5) for $r = k, k-1, \ldots, i$. Let c be a solution to $\gcd(t_i + ct_{i+1}, a_i) = \gcd(t_i, t_{i+1}, a_i)$ with $0 \le c < a_i$. Set $\mathrm{row}(T, i) = c\ \mathrm{row}(T, i+1)$ to produce the matrix

$$T' = \left[\begin{array}{ccccccc|cccc}
a_1 & & & & & & t_1 & * & * & \cdots & * \\
& a_2 & & & & & t_2 & * & * & \cdots & * \\
& & \ddots & & & & \vdots & \vdots & & & \\
& & & a_i & ca_{i+1} & & t_i + ct_{i+1} & * & & & \\
& & & & a_{i+1} & & t_{i+1} & * & & & \\
& & & & & \ddots & \vdots & \vdots & & & \\
& & & & & a_{k-1} & t_{k-1} & * & * & \cdots & * \\
& & & & & & t_k & * & * & \cdots & *
\end{array}\right]$$

Now,

$$\begin{aligned}
\gcd(T'_{i,i}, T'_{i,k}) &= \gcd(a_i, t_i + ct_{i+1}) \\
&= \gcd(a_i, t_i, t_{i+1}) \\
&= \gcd(a_i, t_i, a_{i+1}, t_{i+1}) \\
&= \gcd(a_i, t_i, t_{i+1}, t_{i+2}, \ldots, t_k)
\end{aligned}$$

Here, the second last equality follow from the fact that $a_i | a_{i+1}$, and the last equality follows by the induction hypothesis. Finally, reduce off-diagonal entries in row i of T' modulo the diagonal entry a_i. Since $a_i | a_{i+1}$, the entry in the i-th row $i+1$'st column will be zeroed out, leaving the the first $k-1$ columns unchanged. The following code implements the above construction.

1. for $i = k-1$ by -1 to 1 do
2. $\quad c \leftarrow$ a solution to $\gcd(T_{i,k} + cT_{i+1,k}, a_i) = \gcd(T_{i,k}, T_{i+1,k}, a_i)$ with $0 \le c < a_i$;
3. $\quad \mathrm{row}(T, i) \leftarrow \mathrm{row}(T, i) + c\ \mathrm{row}(T, i+1)$;
4. $\quad$ for $j = i+1$ to m do
5. $\quad\quad T_{i,j} \leftarrow T_{i,j} \pmod{a_i}$;

For convenience, denote D by a_k. Then the loop at line 4 requires $O(m(\lg a_{i+1})(\lg a_i))$ bit operations. This bounds the cost of line 2, which is accomplished using the algorithm of Lemma 4. Thus, one pass of the outer loop is bounded by $cm \lg^2 a_{i+1}$ bit operations for some absolute constant c. The total cost of the outer loop is

$$\begin{aligned}
\sum_{1 \le i \le k-1} cm \lg^2 a_{i+1} &\le cm \lg^2 a_k + cm \sum_{1 \le i \le k-2} \lg^2 a_{i+1} \\
&\le cm(1 + \log^2 D) + cm \sum_{1 \le i \le k-2} (1 + \log^2 a_{i+1}) \\
&\le cm(2 \log^2 D) + cm \sum_{1 \le i \le k-2} (2 \log^2 a_{i+1}) \\
&\le cm(2 \log^2 D) + cm(2 \log^2 D)
\end{aligned}$$

Thus, the running time of the algorithm is bounded by $O(m \log^2 D)$ bit operations. ■

Lemma 7 *Let T be as in (4) with $k > 1$ and satisfy the 5 conditions of Lemma 6. There exists a deterministic algorithm that transforms T to an equivalent matrix that satisfies the same conditions but with the addition of*

6. *$T_{1,1}$ divides all other entries in the principal k-th minor of T,*

7. *If $k > 1$ then $T_{1,k} = 0$.*

If the input matrix satisfies $a_1 > 1$, then the running time of the algorithm is $O(m(\log D)(\log a_1))$ bit operations.

Proof. We show how to transform T to the equivalent matrix

$$T' = \left[\begin{array}{ccccc|cccc} s_1 & & & & & * & * & \cdots & * \\ & a_2 & & & t_2a_1/s_1 & * & * & \cdots & * \\ & & \ddots & & \vdots & \vdots & & & \\ & & & a_{k-1} & t_{k-1}a_1/s_1 & & & & \\ & & & & t_ka_1/s_1 & * & * & \cdots & * \end{array}\right],$$

where $s_1 = \gcd(a_i, t_i)$, using only unimodular row and column operations. Let (s,t) be a solution to the extended gcd equation $sa_1 + tt_1 = s_1$ and let V be the $n \times n$ identity matrix except with $V_{1,1} = s$, $V_{1,k} = t$, $V_{k,1} = a_1/s_1$ and $V_{k,k} = -t_1/s_1$. Then V is unimodular and

$$TV = \left[\begin{array}{ccccc|cccc} s_1 & & & & 0 & * & * & \cdots & * \\ tt_2 & a_2 & & & t_2a_1/s_1 & * & * & \cdots & * \\ tt_3 & & \ddots & & \vdots & \vdots & & & \\ \vdots & & & a_{k-1} & t_{k-1}a_1/s_1 & & & & \\ tt_k & & & & t_ka_1/s_1 & * & * & \cdots & * \end{array}\right].$$

By assumption, a_1 divides t_i for $i = 1, 2, 3, \ldots, k$. Since $s_1 | a_1$, we can add appropriate multiples of row 1 in matrix TV to rows $2, 3, \ldots, k$ to zero out off-diagonal entries in column 1 and produce the matrix T'. Finally, reduce off-diagonal entries in the first $k-1$ rows of T' modulo the diagonal entry in the same column, and reduce off-diagonal entries in row k modulo D. The following code implements the above construction. For convenience, a_k is taken to be D.

1. $(s_1, s, t) \leftarrow$ a solution to $sa_1 + tt_1 = s_1$ with $s_1 = \gcd(a_1, t_1)$;
2. $T_{1,1} \leftarrow s_1$;
3. $T_{1,k} \leftarrow 0$;
4. for $i = 2$ to k do
5. $\quad q \leftarrow tt_i/s_1 \pmod{a_i}$;
6. $\quad \mathrm{row}(T, i) \leftarrow \mathrm{row}(T, i) - q\ \mathrm{row}(T, 1)$;
7. $\quad T_{i,k} \leftarrow T_{i,k}a_1/s_1$;
8. $\quad$ for $j = k$ to m do
9. $\quad\quad T_{i,j} \leftarrow T_{i,j} \pmod{a_i}$;
10. for $j = k$ to m do
11. $\quad T_{1,k} \leftarrow T_{1,k} \pmod{s_1}$;

Line 1 can be accomplished in $O((\lg a_1)^2)$ bit operations to yield a solution (s, t, s_1) satisfying $|s|, |t|, s_1 \le a_1$. Since magnitudes of off-diagonal entries in rows 1 and i are bounded by a_1 and a_i respectively, line 6 requires $O(m(\lg a_i)(\lg a_1))$ bit operations. This bounds

the cost of lines 5 and 7 as well. After lines 5, 6 and 7 are completed, row i of T has entries bounded in magnitude by $2a_ia_1$, leading to a cost of $O(m(\lg a_i)(\lg a_1))$ for the loop at line 8. The total cost for the loop in line 4 is given by

$$\begin{aligned}
\sum_{i=2}^{k} c(\lg a_i)(\lg a_1) &= c(\lg a_k)(\lg a_1) + c(\lg a_1)\sum_{i=2}^{k-1}(\lg a_i) \\
&\leq c(1+\log D)(1+\log a_1) + c(1+\log a_1)\sum_{i=2}^{k-1}(1+\log a_k) \\
&\leq c(2\log D)(2\log a_1) + c(2\log a_1)\sum_{i=2}^{k-1}(2\log a_k) \\
&\leq c(2\log D)(2\log a_1) + c(2\log a_1)(2\log D)
\end{aligned}$$

Thus, the loop in line 4 is bounded by $O(m(\log D)(\log a_1))$ bit operations. This bounds the cost of line 1 and the loop in line 10. ■

We can now prove Theorem 5. Let T be a $k \times m$ matrix which satisfies the conditions of the Theorem. If $k > 1$, apply the algorithm of Lemma 6 at a cost of $O(m\log^2 D)$ bit operations to to "condition" T. Next, apply the algorithm of Lemma 7 to the trailing $i \times i$ minor of T for $i = k, k-1, \ldots, 2$ at a cost of

$$\begin{aligned}
\sum_{1\leq i\leq k} cm(\log D)(\log a_i) &= cm(\log D)\sum_{1\leq i\leq k}(\log a_i) \\
&\leq cm(\log^2 D)
\end{aligned}$$

So far the total cost is bounded by $O(m\log^2 D)$ bit operations. It remains to replace $T_{k,k}$ by $|T_{k,k}|$ and reduce off-diagonal entries in row k modulo $T_{k,k}$. Since $T_{k,k} \leq D$, the cost of this is also bounded $O(m\log^2 D)$ bit operations. ■

3.2 Phase Two Subroutines

Let T be a $k \times m$ rank k upper triangular matrix with first k columns in Smith normal form. We can write T as

$$T = \left[\begin{array}{cccc|ccccc} a_1 & & & & b_{1,1} & b_{1,2} & \cdots & b_{1,m-k} \\ & a_2 & & & b_{2,1} & b_{2,2} & \cdots & b_{2,m-k} \\ & & \ddots & & \vdots & & & \\ & & & a_k & b_{k,1} & b_{k,2} & \cdots & b_{k,m-k} \end{array}\right] \tag{6}$$

The purpose of this section is to prove the following result.

Theorem 8 *Let T be a $k \times m$ integral matrix with first k columns rank k and in Smith normal form and off-diagonal entries in each row reduced modulo the diagonal entry in the same row. There exists a deterministic algorithm that applies unimodular column operations to transform T to an equivalent matrix with principal k-th minor lower triangular and last $m-k$ columns zero. If $T_{1,1} > 1$, and the determinant of the principal k-th minor of T is bounded in magnitude by D, then the output matrix will have all entries bounded in magnitude by D and the running time of the algorithm is $O(m\log^2 D)$ bit operations.*

Proof. Let (s,t) be a solution to the extended Euclidean problem $sa_1 + tb_{1,j} = s_1$ where $s_1 = \gcd(a_1, b_{1,j})$. Let V be the $k \times k$ identity matrix except with $V_{1,1} = s$, $V_{1,j} = t$, $V_{j,1} = -b_{1,j}/s_1$ and $V_{k,k} = a_1/s_1$. Then V is unimodular and

$$TV = \left[\begin{array}{cccc|cccc} s_1 & & & & 0 & b_{1,2} & \cdots & b_{1,m-k} \\ tb_{2,1} & a_2 & & & b_{2,1}a_1/s_1 & b_{2,2} & \cdots & b_{2,m-k} \\ \vdots & & \ddots & & \vdots & & & \\ tb_{k,1} & & & a_k & b_{k,1}a_1/s_1 & b_{k,2} & \cdots & b_{k,m-k} \end{array}\right],$$

Now, reduce off-diagonal entries in columns 1 and j in TV modulo the diagonal entry in the same row. Repeating this process, zero out all off-diagonal entries in row 1. The following code implements the above construction.

1. for $j = 1$ to $m - k$ do
2. $\quad (s_1, s, t) \leftarrow$ a solution to $sT_{1,1} + tb_{1,j} = s_1$ with $s_1 = \gcd(T_{1,1}, b_{1,j})$;
3. $\quad T_{1,1} \leftarrow s_1$;
4. $\quad T_{1,j} \leftarrow 0$;
5. $\quad a \leftarrow -b_{1,j}/s_1$;
6. $\quad b \leftarrow a_1/s_1$;
7. $\quad$ for $i = 2$ to k do
8. $\quad\quad C \leftarrow sT_{i,1} + tb_{i,j} \pmod{a_i}$;
9. $\quad\quad T_{i,j} \leftarrow aT_{i,1} + bb_{i,j} \pmod{a_i}$;
10. $\quad\quad T_{i,1} \leftarrow C$;

Since all entries in row i of T are reduced modulo a_i, the quantities s_1, s, t, a and b computed in lines 2,5 and 6 will have magnitude bounded by a_1 and the cost of one pass of these lines is bounded by $O((\lg^2 a_1))$ bit operations. The cost of a single pass of the loop at line 7 is $O((\lg a_1)(\lg a_i))$ bit operations. The total cost of the loop at line 7 for a single value of j is is given by

$$\begin{aligned} \sum_{2 \le i \le k} c(\lg a_1)(\lg a_i) &= c(1 + \log a_1) \sum_{2 \le i \le i} (1 + \log a_i) \\ &\le c(2 \log a_1) \sum_{2 \le i \le k} (2 \log a_i) \\ &\le c(2 \log a_1)(\log D) \end{aligned}$$

Thus, the total cost of the loop at line 7 for a single value of j is bounded by $O((\log a_1)(\log D))$ bit operations, and this bounds the cost of lines 2,5 and 6 as well. The outer loop at line 1 is repeated $m - k < m$ times so the total cost of the code fragment is $O(m(\log a_1)(\log D))$ bit operations.

After the code completes the work matrix has the form

$$\left[\begin{array}{cccc|cccc} s_1 & & & & 0 & 0 & \cdots & 0 \\ * & a_2 & & & * & * & \cdots & * \\ \vdots & & \ddots & & \vdots & & & \\ * & & & a_k & * & * & \cdots & * \end{array}\right].$$

To complete the reducetion, apply a similar procedure to the trailing $(k-i+1) \times (m-i+1)$ submatrix of the work matrix to zero out entries to the right of the diagonal in row i for

$i = 2, 3, \ldots, k$ to complete the reduction. The total cost to reduce T to lower triangular form is

$$\sum_{1 \le i \le k} cm(\log a_i)(\log D) \le cm \log^2 D$$

bit operations for some absolute constant c. ■

3.3 The Triangular Smith Normal Form Algorithm

Theorem 9 *There exists a deterministic algorithm that receives as input an $n \times m$ integral matrix A with principal $n \times n$ minor nonsingular and upper triangular, and returns the Smith normal form of A. If both the magnitude of all entries of A and the magnitude of the determinant of the principal $n \times n$ minor of A is bounded by D, then the running time of the algorithm is $O(nm \log^2 D)$ bit operations using standard integer arithmetic.*

Phase one of the algorithm transforms, for $r = 1, 2, \ldots, n$ in succession, the principal $r \times r$ minor of A into Smith normal form. The algorithm is inductive and it is sufficient to consider a single stage. Let $A^{(r)}$ be the work matrix at the beginning of stage r. Off-diagonal entries in rows $1, 2, \ldots, r-1$ of $A^{(r)}$ will have magnitude bounded by the diagonal entry in the same row, and using a block decomposition, the $n \times m$ matrix $A^{(r)}$ can be written as

$$A^{(r)} = \left[\begin{array}{c|c|c} I_{r-k} & & \\ \hline & T_1 & T_2 \\ \hline & & B \end{array}\right]. \tag{7}$$

T_1 is $k \times k$ upper triangular with rank k and with first $k-1$ columns in Smith normal form. T_2 is $k \times (m-r)$ with entries in rows i, $1 \le i \le k-1$, reduced modulo the i-th diagonal entry in T_1 and entries in row k reduced modulo D. B is comprised of the last $(n-r)$ rows and $(m-r)$ columns of the input matrix A. Furthermore, if $k \ge 2$, then the leading diagonal entry in T_1 is greater than 1. For the initial case of the reduction, note that $A^{(1)} = A$ can be written as in (7) with $k = 1$.

At stage r, row operations on $A^{(r)}$ are limited to rows occupied by T_1 and the only column operations involving columns occupied by B are limited to those which add multiples of columns occupied by T_1 to these last $m-r$ columns. Hence, the last $n-r$ rows of the work matrix (those occupied by B) remain unchanged during stage r. Furthermore, since the principal $(r-k)$-th submatrix of $A^{(r)}$ is the identity, we can limit our attention the submatrix of $A^{(r)}$ comprised of T_1 and T_2, namely the $k \times (m-r)$ matrix

$$\begin{aligned} T &= \left[\begin{array}{c|c} T_1 & T_2 \end{array}\right] \\ &= \left[\begin{array}{ccccc|cccc} a_1 & & & & t_1 & * & * & \cdots & * \\ & a_2 & & & t_2 & * & * & \cdots & * \\ & & \ddots & & \vdots & * & * & \cdots & * \\ & & & a_{k-1} & t_{k-1} & \vdots & & & \\ & & & & t_k & * & * & \cdots & * \end{array}\right] \end{aligned}$$

Use the algorithm of Theorem 5 to transform T to an equivalent matrix with principal $k \times k$ minor in Smith normal form and off-diagonals in each row reduced modulo the diagonal entry in each column. By Theorem 5, the cost of one stage is $O(m \log^2 D)$ bit operations. This stage is repeated for $r = 1, 2, \ldots, n$, leading to a total cost for phase one of $O(nm \log^2 D)$ bit operations.

After phase one, the work matrix either has principal n-th minor the identity and all other entries zero (in which case we are finished) or has the form

$$\left[\begin{array}{c|c|c} I_{n-k} & & \\ \hline & S_1 & B \end{array}\right]$$

where S_1 is a $k \times k$ matrix in Smith normal form with leading diagonal entry greater than 1. Furthermore, off-diagonal entries in each row are reduced modulo the diagonal entry in the same row. Use the algorithm Theorem 8, to transform the submatrix $\left[\begin{array}{c|c} S_1 & B \end{array}\right]$ to an equivalent matrix with first k columns lower triangular and all other columns zero. By Theorem 8, the cost of the transformation is $O(m \log^2 D)$ bit operations. After phase 2 the work matrix has the form

$$\left[\begin{array}{c|c|c} I_{n-k} & & \\ \hline & A_1 & O \end{array}\right]$$

where A_1 is a $k \times k$ lower triangular with magnitude of determinant and all entries bounded by D. Finally, use the reduction of phase 1 to reduce the square nonsingular matrix A_1 to Smith normal form at a cost of $O(n^2 \log^2 D)$ bit operations. Combining these phases, the total cost for the reduction is seen to be bounded by $O(nm \log^2 D)$ bit operations. ■

4 A Fast+Practical+Deterministic Algorithm for the Smith Normal Form

Let A be an $n \times m$ input matrix. The first step in computing the Smith normal form is transform A to a row equivalent matrix H which satisfies the following conditions:

($e1$) Let r be the rank of H. Then the first r rows of H are nonzero.

($e2$) For $1 \leq i \leq r$ let $H[i, j_i]$ be the first nonzero entry in row i. Then $j_1 < j_2 < \ldots < j_r$.

A matrix H which satisfies ($e1$) and ($e2$) and is row equivalent to A is said to a row echelon form of A. In [Sto96a] we give a fast+practical+deterministic algorithm for recovering r and a row echelon form H of A in $O(nmr^2 \log^2 r||A|| + r^4 \log^3 r||A||)$ bit operations. Furthermore, both $||H||$ and $D = |\prod_{i=1}^{r} H[i, j_i]|$ will be bounded in length by $O(r \log r||A||)$ bits. Let P be a permutation matrix such that column i of HP is column j_i of H for $1 \leq i \leq r$. The Smith normal form of HP (which is equivalent to A) can be computed in $O(rm \log^2 D)$ bit operations by applying the algorithm of Theorem 9 to the submatrix comprised of the first r rows of HP. Noting the $O(rm \log^2 D)$ is bounded by $O(nmr^2 \log^2 r||A||)$ yields the following result.

Theorem 10 *There exists a deterministic algorithm that takes as input an $n \times m$ integer input matrix A and recovers the rank r and Smith normal form S of A. The cost of the algorithm is $O(nmr^2 \log^2 r||A|| + r^4 \log^3 r||A||)$ bit operations assuming standard integer and matrix arithmetic.*

Proof. Follows from Theorem 9 and [Sto96a, Theorem 16]. ■

References

[Bac92] Eric Bach. Linear algebra modulo N. Unpublished manuscript., December 1992.

[BDS93] E. Bach, J. Driscoll, and J. O. Shallit. Factor refinement. *Journal of Algorithms*, 15:199—222, 1993.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory*, volume 1 : Efficient Algorithms. MIT Press, 1996.

[Col68] G. E. Collins. Computing time analyses for some arithmetic and algebraic algorithms. Technical Report 36, University of Wisconsin, Madison; Computer Sciences, July 1968. Appeared in *Proc. 1968 Summer Institute on Symbolic Mathematical Computations*, IBM Federal Systems Center, 1968, pp. 195–231.

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[DKT87] P. D. Domich, R. Kannan, and L. E. Trotter, Jr. Hermite normal form computation using modulo determinant arithmetic. *Mathematics of Operations Research*, 12(1):50–59, February 1987.

[Gie95] Mark Giesbrecht. Fast computation of the Smith normal form of an integer matrix. In A. H. M. Levelt, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '95*, pages 110–118, 1995.

[Gie96] Mark Giesbrecht. Probabalistic computation of the Smith normal form of a sparse integer matrix. In H. Cohen, editor, *Algorithmic Number Theory: Second International Symposium*, pages 175–188, 1996. Proceedings to appear in Springer's Lecture Notes in Computer Science.

[Her51] C. Hermite. Sur l'introduction des variables continues dans la théorie des nombres. *J. Reine Angew. Math.*, 41:191–216, 1851.

[HM91] James L. Hafner and Kevin S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM Journal of Computing*, 20(6):1068–1083, December 1991.

[Ili89] Costas 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 Journal of Computing*, 18(4):658–669, August 1989.

[SL96] Arne Storjohann and George Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. In Y. N. Lakshman, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '96*, pages 259–266. ACM Press, 1996.

[Smi61] H. J. S. Smith. On systems of linear indeterminate equations and congruences. *Phil. Trans. Roy. Soc. London*, 151:293–326, 1861.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[Sto96a] Arne Storjohann. A fast+practical+deterministic algorithm for triangularizing integer matrices. Technical Report 255, Departement Informatik, ETH Zürich, December 1996.

[Sto96b] Arne Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In Y. N. Lakshman, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '96*, pages 267–274. ACM Press, 1996.