ASYMPTOTICALLY FAST TRIANGULARIZATION OF MATRICES OVER RINGS*

JAMES L. HAFNER† AND KEVIN S. MCCURLEY‡

Abstract. We consider the problems of triangularizing and diagonalizing matrices over rings, with particular emphasis on the integral case. We begin by describing fast algorithms for the computation of Hermite and Smith normal forms of integer matrices. We then show how to apply fast matrix multiplication techniques to the problem of triangularizing a matrix over a ring using elementary column operations. These general results lead to an algorithm for triangularizing integer matrices that has a faster running time than the known Hermite normal form algorithms. The triangular matrix that is computed has small entries like the Hermite normal form, and will suffice for many applications.

AMS (MOS) Subject Classification: 68Q40, 11Y16, 15A23

Key words: Hermite normal form, factorization of matrices, algorithms.

Abbreviated title: Triangularization of Matrices.

1. Introduction. A common theme when performing computations involving matrices is to use a triangular or diagonal matrix that is equivalent for the computational task at hand. For example, Hermite and Smith normal forms of integer matrices (see Section 2 for definitions) have applications to a number of areas including solving systems of linear Diophantine equations (see [13]), integer programming (see [28]), algorithmic problems in lattices (see [14] for an example) and the structure theorem for finitely generated Abelian groups (see [15, Chapters 7–10]).

In this paper, we consider the problem of computing such special forms of matrices over rings, with particular emphasis on the integers. In Section 2 we give deterministic algorithms for computing the Hermite and Smith normal forms of integer matrices which have fast and rigorously proved running times. While the emphasis of our paper is on the asymptotic analysis of algorithms, we believe these algorithms are quite practical.

* An extended abstract of this work appeared in The Proceedings of the First Annual ACM/SIAM Symposium on Discrete Algorithms, 1990

† IBM Almaden Research Center, K53/802, 650 Harry Road, San Jose, California 95120.

‡ Work done while visiting IBM Almaden Research Center. Current address: Division 1423, Sandia National Laboratories, Albuquerque, NM 87185.

The main results of this paper appear in Sections 3 and 4. In Section 3, we consider the problem of matrix triangularization over very general rings, employing the techniques of fast matrix multiplication. In Section 4, these methods are then applied to the special case of matrices over the integers to reduce the computational complexity of triangularization. Note that in many applications, a triangular matrix can be used in place of the Hermite normal form. In the last section, we discuss some open problems.

We of course require some notation. If A is a matrix over a commutative ring R, we let $\mathcal{L}(A)$ denote the R-module consisting of all R-linear combinations of the columns of A. In the case $R = \mathbb{Z}$, $\mathcal{L}(A)$ is the lattice generated by the columns of A. In this case, we let $\mathrm{Det}(\mathcal{L}(A))$ denote the determinant of this lattice. We will say that A is right unimodularly equivalent over R to a matrix B if there exists a unimodular matrix K such that $AK = B$. (A unimodular matrix K satisfies $\det(K) = \pm 1$.) Finally, we use the notation log to denote logarithms to base 2.

We would like to thank Don Coppersmith and Robert Guralnick for helpful conversations during the course of this research.

2. Hermite and Smith Normal Forms. Let A denote an $m \times n$ matrix with integer entries. A classical result says that if A has rank m, then there exists an $m \times n$ lower triangular matrix H and an $n \times n$ unimodular matrix K such that $A = HK$, and

$$\begin{aligned} h_{ii} &> 0, && 1 \le i \le m, \\ 0 \le h_{ij} &< h_{ii}, && 1 \le i \le m, 1 \le j < i. \end{aligned}$$

This was first proved for the case of square matrices by Hermite [17]. The matrix H is unique, and is known as the Hermite normal form of A. Note that $\mathrm{Det}(\mathcal{L}(A)) = \prod_{i=1}^{m} h_{ii}$.

A related construction is the Smith normal form. Smith [29], proved that for every $m \times n$ matrix A of rank r, there exist unimodular matrices J, K and a diagonal matrix S with $A = JSK$, $S = \mathrm{diag}(s_1, \ldots, s_r, 0, \ldots, 0)$, and $s_1 \mid s_2 \mid \ldots \mid s_r$, with $s_i > 0$. The matrix S is unique, and is known as the Smith normal form of A. The integers $s_1, s_2, \ldots, s_r$ are called the invariant factors of A. If A has rank m, then the product of the invariant factors equals $\mathrm{Det}(\mathcal{L}(A))$.

In this section we shall describe some algorithms for computing the Hermite and Smith normal forms of integer matrices, and analyze the performance of the algorithms. Numerous researchers have previously given algorithms for the calculation

of Hermite and Smith normal forms, as well as the related problem of solving systems of linear Diophantine equations. These include Hermite [17], Smith [29], Blankinship [2], [3], Bradley [5], Havas and Sterling [16], Frumkin [12] [11], Domich [9], Kannan and Bachem [23], Domich, Kannan, and Trotter [10], Kaminski and Paz [22], Hu [18, Appendix A], Schrijver [28, Chapters 4–5], McClellan [24], Chou and Collins [7], and Iliopolous [20].

The simplest algorithms for computing the Hermite normal form use a procedure similar to the Euclidean algorithm, replacing the diagonal element in a given row by the gcd of the elements in the same row to the right. This procedure takes $O(m^2 n)$ operations on rational numbers, but such algorithms are well known to suffer from "intermediate expression swell", since the numbers that are encountered during the calculation can become enormous, as was observed in [2] and [12] among others. As an example of the kind of intermediate expression swell that can occur, the first author used the software package Mathematica to run some trials on random 20×20 matrices with integer entries between 0 and 10. After reduction to triangular form using the standard Hermite procedure, they almost always gave at least one entry exceeding 10^{500}, and after several such examples the following rather spectacular example was found. Consider the matrix

$$\begin{bmatrix}
10 & 8 & 10 & 5 & 5 & 10 & 0 & 8 & 5 & 9 & 8 & 3 & 0 & 9 & 9 & 2 & 7 & 6 & 0 & 4 \\
4 & 10 & 8 & 5 & 2 & 6 & 5 & 1 & 2 & 6 & 9 & 1 & 4 & 6 & 2 & 7 & 7 & 1 & 4 & 10 \\
2 & 4 & 6 & 4 & 3 & 0 & 3 & 0 & 10 & 2 & 3 & 0 & 6 & 7 & 5 & 7 & 7 & 3 & 3 & 2 \\
1 & 5 & 0 & 4 & 8 & 0 & 10 & 10 & 2 & 0 & 5 & 3 & 6 & 10 & 2 & 10 & 2 & 10 & 10 & 2 \\
9 & 8 & 5 & 4 & 10 & 6 & 3 & 10 & 3 & 8 & 7 & 4 & 7 & 1 & 5 & 8 & 6 & 8 & 3 & 1 \\
10 & 3 & 1 & 0 & 0 & 9 & 8 & 2 & 1 & 6 & 0 & 10 & 7 & 4 & 9 & 2 & 3 & 10 & 2 & 9 \\
8 & 2 & 8 & 0 & 0 & 9 & 3 & 0 & 7 & 8 & 0 & 3 & 4 & 5 & 3 & 8 & 1 & 4 & 4 & 9 \\
0 & 4 & 8 & 6 & 0 & 1 & 5 & 0 & 0 & 3 & 4 & 6 & 9 & 5 & 10 & 10 & 3 & 5 & 9 & 9 \\
4 & 6 & 4 & 8 & 9 & 2 & 5 & 6 & 9 & 8 & 9 & 3 & 6 & 7 & 3 & 8 & 5 & 10 & 1 & 7 \\
2 & 0 & 7 & 10 & 8 & 4 & 7 & 5 & 3 & 7 & 1 & 10 & 6 & 2 & 2 & 9 & 9 & 9 & 1 & 10 \\
2 & 2 & 1 & 5 & 4 & 1 & 4 & 4 & 6 & 7 & 0 & 10 & 0 & 2 & 5 & 1 & 9 & 10 & 2 & 5 \\
4 & 1 & 10 & 2 & 4 & 0 & 9 & 7 & 8 & 9 & 9 & 3 & 3 & 9 & 10 & 1 & 7 & 1 & 3 & 1 \\
5 & 3 & 3 & 9 & 2 & 1 & 4 & 2 & 7 & 8 & 5 & 7 & 5 & 5 & 9 & 4 & 5 & 4 & 10 & 6 \\
4 & 8 & 10 & 1 & 0 & 8 & 10 & 4 & 4 & 10 & 3 & 10 & 7 & 4 & 5 & 7 & 3 & 2 & 1 & 2 \\
6 & 9 & 4 & 7 & 10 & 3 & 6 & 9 & 5 & 8 & 4 & 6 & 9 & 8 & 7 & 5 & 9 & 8 & 9 & 7 \\
5 & 10 & 5 & 6 & 10 & 0 & 9 & 5 & 1 & 4 & 0 & 9 & 8 & 9 & 3 & 5 & 2 & 2 & 0 & 8 \\
1 & 3 & 10 & 2 & 5 & 5 & 1 & 1 & 10 & 6 & 10 & 4 & 3 & 8 & 1 & 6 & 1 & 2 & 6 & 9 \\
6 & 4 & 1 & 5 & 8 & 2 & 3 & 1 & 3 & 7 & 4 & 10 & 2 & 9 & 10 & 3 & 3 & 6 & 5 & 8 \\
3 & 7 & 5 & 2 & 10 & 0 & 10 & 4 & 0 & 0 & 10 & 3 & 2 & 9 & 5 & 1 & 7 & 10 & 5 & 3 \\
10 & 2 & 10 & 6 & 2 & 9 & 3 & 2 & 0 & 2 & 5 & 3 & 3 & 1 & 5 & 5 & 10 & 0 & 7 & 3
\end{bmatrix}.$$

After reducing this matrix to triangular form, there is an entry exceeding 10^{5011}. By contrast, the Hadamard bound for the determinant of this matrix is approximately 10^{33}, and the algorithm that we present here would only require modular arithmetic on integers of this size. It should be mentioned however that the bad reputation of the classical Hermite algorithm is based on computational experience and heuristic arguments, and that it remains an open question whether the length of the entries remain bounded by a polynomial in the length of the input. See [12] for a more thorough discussion of this.

Apparently the first published algorithm for Hermite normal form that substantially overcame this problem of large integer entries is due to Frumkin [12]. Frumkin's algorithm has a running time that is bounded by a polynomial of degree 6 in the length of the input and uses an algorithm for the computation of solutions of linear Diophantine equations. Shortly thereafter, Kannan and Bachem [23] published another polynomial time algorithm for the computation of Hermite and Smith normal forms. The Kannan-Bachem method uses a rearrangement in the order of operations of the classical Hermite algorithm, and in so doing they were able to prove a polynomial time and space bound for the algorithm. Chou and Collins [7] modified the Kannan-Bachem procedure and gave a better space bound.

Frumkin's approach uses modular arithmetic to control the size of the entries that arise. Others that have used modular arithmetic in various forms include Hu [18, Appendix A], Schrijver [28, Chapter 4], Domich [9], Domich, Kannan, and Trotter [10], and Iliopolous [20]. In Section 2.1 below we present and analyze an algorithm for computing the Hermite normal form that is an extension of the Domich, Domich-Kannan-Trotter, and Iliopolous algorithms to nonsquare matrices, where the determinant is replaced by a suitable modulus. We also give a different method for calculating the modulus. The extension is relatively straightforward, but we present it here for completeness and to illustrate the improvements given later in Section 3.

In Section 2.2, we extend the methods of Domich, Kannan, and Trotter to get a method for calculation of Smith normal forms of integer matrices. This was first done for square matrices of full rank by Domich [9], and independently by Schrijver [28, Chapter 4]. A similar construction was given in [18, Appendix A] but it does not seem

to be complete. More recently, Iliopolous [20] has described a very similar algorithm (at least for matrices of full rank).

2.1. An Algorithm for Computing the Hermite Normal Form. We shall express our results in terms of a function $B(t)$ which will bound the number of bit operations required to do each of two different operations. The first operation is to carry out the extended Euclidean algorithm on two $\lceil t \rceil$ bit integers. The second operation is the application of the Chinese remainder theorem with moduli being all primes less than t. By Theorems 8.20 and 8.21 of [1], we can take $B(t) \ll M(t) \log t$, where $M(t)$ is a monotonic upper bound for the number of bit operations required to multiply two $\lceil t \rceil$ bit integers. By a result of Schönhage and Strassen [27], $M(t) \ll t \log t \log\log t$ so that

$$B(t) \ll t \log^2 t \log\log t.$$

We also assume that $B(t)/t$ is nondecreasing.

Our first result is the following.

THEOREM 2.1. *There exists a deterministic algorithm that receives as input an $m \times n$ integral matrix A of rank m and a positive integer h that is a multiple of* $\mathrm{Det}(\mathcal{L}(A))$, *and produces as output the Hermite normal form of A. If the entries of A are bounded in absolute value by T, then the running time of the algorithm is $O(mnB(\log T) + m^2 nB(\log h))$ bit operations.*

If only the matrix A is given, but not h, then the following corollary gives a bound for the running time to compute the Hermite normal form of A.

COROLLARY 2.2. *There exists a deterministic algorithm that receives as input an $m \times n$ integral matrix A of rank m, and produces as output the Hermite normal form of A. If the entries of A are bounded in absolute value by T, then the running time of the algorithm is $O(m^2 nB(m \log(mT)))$ bit operations.*

The corollary follows immediately from Theorem 2.1 and a special case of the following proposition (the general case will be used in the next section).

PROPOSITION 2.3. *There exists a deterministic algorithm that receives as input an $m \times n$ $(m \leq n)$ integral matrix A and produces as output the rank r of A over $\mathbb{Q}$ and a positive integer h such that h is a multiple of the product of the invariant factors*

of A. If the entries of A are bounded in absolute value by T, then $h \leq r^{r/2}T^r$, and the running time of the algorithm is $O(rmnB(m\log(mT)))$ bit operations.

We now give the proof of Proposition 2.3. Our goal will be to find a nonsingular submatrix of dimension r, where r is the rank of A over $\mathbb{Q}$. The number h will be the determinant of this submatrix, which by the Hadamard inequality is bounded by $r^{r/2}T^r$. The first step in finding such a submatrix is to calculate a number $z = O(m\log(mT))$ such that

$$m^{m/2}T^m < \prod_{\substack{p \leq z \\ p \text{ prime}}} p < m^m T^{2m},$$

and to calculate all of the primes up to z. Finding z can be done using the estimates of [26] and the calculation of the primes can be done in time $O(z)$ by the method of [25]. The number r is the largest dimension of a nonzero subdeterminant of A. For every prime p, the rank of A modulo p does not exceed r, and it equals r if and only if p does not divide some $r \times r$ subdeterminant. Since every nonzero subdeterminant of A is less in absolute value than $\prod_{p \leq z} p$, at least one of the primes $p \leq z$ will have the property that the rank of A modulo p is equal to r.

We now proceed as follows. For each prime $p \leq z$, we use row reduction modulo p to find the rank of A modulo p, and we keep track of the prime p for which this rank is maximal. We also keep track of the indices of a maximal set of linearly independent columns. After exhausting all of the primes $p \leq z$, we will have a maximal set of linearly independent columns of A over $\mathbb{Q}$, the rank r, and a prime p not dividing some $r \times r$ subdeterminant of the matrix formed by this set of columns. We can then apply column reduction modulo p to find a set of r rows which are linearly independent modulo p. The selected rows and columns from the matrix A will form an $r \times r$ nonsingular (over $\mathbb{Q}$) submatrix. Up to this point the number of bit operations is easily seen to be

$$(1) \qquad \ll rmn \sum_{p \leq z} B(\log p) \ll rmnB(c\log z)\frac{z}{\log z},$$

by the prime number theorem. Since $B(t)/t$ is nondecreasing, this gives at most $O(rmnB(z))$ bit operations.

The final step in this algorithm is to compute h, the absolute value of the determinant of this submatrix. For this we simply compute the determinant modulo p for

each prime $p \leq z$, (actually we could use a smaller value for z if $r < m$ but this is unimportant) and use the Chinese remainder theorem to find the determinant modulo $\prod_{p \leq z} p$. (We note that similar use of the Chinese remainder theorem can be found in [4] and [24].) Since the absolute value of the determinant does not exceed $\prod_{p \leq z} p$, this gives the correct integer value. The determinant can be computed modulo p in $O(r^3)$ operations modulo p, giving a total of at most $O(r^3 B(z))$ bit operations as in (1). We have already noted at the beginning of this section that the Chinese remainder theorem takes only $O(B(z))$ bit operations. The following lemma (Lemma 2.4) implies that the product of the invariant factors of A is the gcd of *all* $r \times r$ subdeterminants of A, so that it divides h. This will complete the proof of Proposition 2.3.

LEMMA 2.4. *Let A be an $m \times n$ integral matrix of rank r, and let $S = \mathrm{diag}(s_1, \ldots, s_r, 0, \ldots, 0)$ be the Smith normal form of A. For $1 \leq i \leq r$, let $\gamma_i(A)$ be the* gcd *of all $i \times i$ subdeterminants of A. Then*

$$\begin{aligned} s_1 &= \gamma_1(A) \\ s_i &= \gamma_i(A)/\gamma_{i-1}(A), \quad 2 \leq i \leq r \end{aligned}$$

The lemma follows immediately from the observations that the γ_is are invariant under unimodular row and column operations, and that S can be computed from A by such operations (see [15, section 7.7]).

We note that in the application of Proposition 2.3 to Corollary 2.2, we are assuming that the matrix has full rank m. Thus we need only do the first search until we find a prime p and m linearly independent columns modulo p. This will then form the necessary submatrix whose determinant we calculate as above. The running time of this procedure will differ from that stated in the proposition only by a constant factor.

We now turn to the proof of Theorem 2.1. The first step is to reduce the entries of A modulo h, and then to convert A into a lower triangular matrix L using unimodular column operations followed by reduction modulo h. This proceeds in the standard order, moving from top to bottom and left to right. In order to introduce a zero in the i,j location (where $i < j$), we use the extended Euclidean algorithm to calculate integers r and t such that $ra_{ii} + ta_{ij} = g$, where $g = \gcd(a_{ii}, a_{ij})$ and $|r|, |t| \leq h$. We then replace column i by $r \cdot (\text{column } i) + t \cdot (\text{column } j)$, and we replace column j by $-a_{ij}/g \cdot (\text{column } i) + a_{ii}/g \cdot (\text{column } j)$. This is equivalent to postmultiplication by the

$n \times n$ unimodular matrix constructed by embedding the matrix

$$\begin{bmatrix} r & -a_{ij}/g \\ t & a_{ii}/g \end{bmatrix}$$

into the $n \times n$ identity matrix, and will replace a_{ii} by $\gcd(a_{ii}, a_{ij})$ and a_{ij} by 0. After each such column operation we reduce the entries in the two columns modulo h.

This clearly produces a lower triangular matrix L. The following lemma shows how to reconstruct the diagonal of the Hermite normal form of A from L.

LEMMA 2.5. *Let A be an integral matrix of size $m \times n$ and rank m, $H = (h_{ij})$ its Hermite normal form, and h a positive integer multiple of* $\mathrm{Det}(\mathcal{L}(A))$. *Let $d_1 = h$, and $d_{i+1} = d_i/h_{ii}$, $i = 1, \ldots, m-1$. If $L = (l_{ij})$ is any lower triangular matrix obtained from A by unimodular column operations followed by reduction modulo h, then $h_{ii} = \gcd(d_i, l_{ii})$.*

The proof of this lemma can be found in [14], but it also follows closely the arguments in [10].

To complete the construction of the Hermite normal form $H = (h_{ij})$, we proceed as follows. First we reconstruct the diagonal of H. Let l_{ij}, $i, j = 1, \ldots, m$ be the entries of L. Set $d_1 = h$. Then for $i = 1, \ldots, m$, use the extended Euclidean algorithm to find integers r_i and t_i such that $r_i d_i + t_i l_{ii} = h_{ii}$ where $h_{ii} = \gcd(d_i, l_{ii})$, and put $d_{i+1} = d_i/h_{ii}$. For $i = 1, \ldots, m$, multiply the ith column of L by t_i and reduce modulo d_i. This produces a matrix whose diagonal is the diagonal of H. It follows from [10, Corollary 2.6] that the columns of the resulting matrix generate $\mathcal{L}(A)$. The final step in computing H is to use unimodular column operations to reduce each of the entries below the diagonal modulo the diagonal entry in its corresponding row. The order of operations is to work from top to bottom and from left to right, and after each column operation we reduce the column entries modulo h.

The running time of this algorithm can be estimated as follows. First, the triangularization modulo h takes at most $O(m^2 n B(\log h))$ bit operations since we have $O(mn)$ entries to zero out, and each requires at most $O(mB(\log h))$ operations. The reconstruction of the diagonal takes at most $O(m^2 B(\log h))$ operations. Finally, the remaining steps to reduce the entries below the diagonal takes at most $O(m^3 B(\log h))$ operations. The total number of operations is thus $O(m^2 n B(\log h))$ bit operations as

claimed.

2.2. An Algorithm for Computing the Smith Normal Form. In this section we shall present and analyze an algorithm for computing the Smith normal form of an integer matrix. Without loss of generality we may assume that our matrix is $m \times n$ with $m \leq n$, since the Smith normal form of A is the same as the transpose of the Smith normal form of the transpose of A.

THEOREM 2.6. *There exists a deterministic algorithm that receives as input an $m \times n$ integral matrix A and a number h that is a positive multiple of the products of the invariant factors of A, and produces as output the Smith normal form of A. If A has rank r and entries bounded in absolute value by T, then the running time of the algorithm is $O(mnB(\log T) + rmnB(\log h)\log h)$ bit operations.*

Once again we can state a result for the case when the number h and the rank r are not provided as part of the input.

COROLLARY 2.7. *There exists a deterministic algorithm that receives as input an $m \times n$ integral matrix A, and produces as output the Smith normal form of A. If A has entries bounded in absolute value by T, then the running time of the algorithm is $O\left(rmn\{B(r\log(rT))r\log(rT) + B(m\log(mT))\}\right)$ bit operations.*

The proof of this corollary is immediate from the theorem above and Proposition 2.3. Since $r \leq m$, a simpler but less accurate bound on the running time is $O\left(m^3 nB(m\log(mT))\log(mT)\right)$.

For the proof of Theorem 2.6, we begin by describing the algorithm to compute the Smith normal form S of A. As in the Hermite normal form algorithm, there are two parts to the algorithm. In the first part, we reduce to a diagonal matrix, proceeding inductively down the diagonal, and in the second part we reconstruct from this diagonal matrix the Smith normal form. To begin the diagonalization procedure, we perform unimodular row and column operations on A followed by reduction modulo h to construct a matrix of the form

$$B = \left[\begin{array}{c|ccc} b_{11} & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & B^* & \\ 0 & & & \end{array}\right], \tag{2}$$

with the further condition that b_{11} divides every entry of the matrix B^*. To accomplish this, we first set $B = A$, and then perform unimodular column operations and reduction modulo h in order to replace b_{11} by the gcd of the entries in the first row, and to introduce zeros into the rest of the first row. We then perform unimodular *row* operations followed by reduction modulo h so as to replace the new b_{11} by the gcd of the entries in the first column. This may destroy the zeros in the first row, but if so then it reduces b_{11} by a multiplicative factor of at least 2. We repeat this process until all entries (except b_{11}) of the first row and column are zero. If now there exists an entry b^*_{ij} of B^* for which $b_{11} \not| b^*_{ij}$, then we add row i back to row 1 and repeat the procedure. Once again this will reduce b_{11} by a multiplicative factor of at least 2. Clearly b_{11} is replaced by a new entry at most $O(\log h)$ times. Hence after $O(mnB(\log h)\log h)$ bit operations, we arrive at a matrix of the form (2) with b_{11} dividing every entry of B^*. We now apply essentially the same procedure to B^* (note that this will not change the first row or column of B).

This procedure could break down if at some point we encounter a first row with all zero entries. In this case we interchange the first row with some nonzero row below it, if such a row exists. If no such row exists, which will happen after at most r steps, then we terminate the first part of the algorithm. The matrix is now a diagonal matrix $\tilde{S} = \mathrm{diag}(\tilde{s}_1, \ldots, \tilde{s}_m)$ with

$$\tilde{s}_1 | \tilde{s}_2 | \cdots | \tilde{s}_r,$$

and

$$\tilde{s}_{r+1} = \tilde{s}_{r+2} = \cdots = \tilde{s}_m = 0.$$

The final part of the algorithm is to reconstruct the Smith normal form S from $\tilde{S}$. To do this, we set $d_1 = h$, and then for $i = 1, \ldots, r$, we set $s_i = \gcd(d_i, \tilde{s}_i)$, and $d_{i+1} = d_i/s_i$ and $s_i = 0$ for $r+1 \le i \le m$. The matrix $S = \mathrm{diag}(s_1, \ldots, s_m)$ is now the Smith normal form of A.

The running time of the diagonalization part of the algorithm requires at most $O(rmnB(\log h)\log h)$ bit operations. Clearly, time for the reconstruction phase is much less, justifying our claim in the theorem for the running time of the complete algorithm.

It remains to prove the correctness of this algorithm, in particular, the reconstruction procedure. But Lemma 2.4 and the proof of Lemma 2.5 can easily be adapted to

show this.

3. Triangularization over Rings. From the previous section it is clear that the computation of the Hermite normal form of an $m \times n$ matrix can be accomplished in at most mn applications of the extended Euclidean algorithm, combined with at most $O(m^2 n)$ operations on integers. The major portion of the work goes into bounding the size of the integers that are involved. In this section we shall take a more theoretical approach to the problem of computing right unimodularly equivalent triangular forms of matrices, with the goal of reducing the total number of ring operations. This is analogous to the results that relate the complexity of matrix inversion [6] and LU factorization [19] to matrix multiplication over fields.

In Section 3.1, we will assume our ring is a commutative ring with identity in which every ideal is principal (PIR). Under these assumptions we give a construction and running time bound for computing matrices K and L with $AK = L$ where L is lower triangular and K is unimodular. This particular form, with the unimodular matrix on the left, has the advantage that we do not need to resort to matrix inversions (a more subtle problem over PIRs than over fields). In Section 3.2 we add additional assumptions on our ring and show how to get the factorization $A = LK$, using matrix inversions. Of course it is obvious that one can get to this formulation by inverting K and multiplying, but the construction we give here yields a better running time.

3.1. Triangularization Over Principal Ideal Rings. We take the following model. Let R be a commutative ring with identity in which every ideal is principal (a PIR), and let $M(m,n,k) = M_R(m,n,k)$ be a bound for the number of ring operations required to multiply an $m \times n$ matrix times an $n \times k$ matrix over R. Here by ring operation we mean an assignment, or an addition, subtraction, or multiplication (but later in Section 3.2 we shall include division as a ring operation). Over a general PIR, the Hermite normal form of a matrix need not exist since there is no notion of size. It is still possible however to obtain a (not necessarily unique) triangularization $AK = L$ over the ring R, where L is lower triangular and K is unimodular.

Let A be $m \times n$ and let t be the number of columns of zeros of A which we may assume are the last t columns. We define a *primary triangularization* of A as

two matrices L and K (over R) such that $AK = L$, L is lower triangular and K is unimodular and of the form

$$K = \begin{bmatrix} K_1 & 0 \\ 0 & I_t \end{bmatrix} \tag{3}$$

Let $T(m,n) = T_R(m,n)$ be the number of ring operations required to compute a primary triangularization for an $m \times n$ matrix A. The major result of this section is to give a bound for $T(m,n)$ in terms of $M(m,n,k)$ and $T(1,2)$. We have written the time bound in terms of the quantity $T(1,2)$ since that is the most primitive operation, not a primary ring operation, which is required to do this triangularization.

The quantity $T(1,2)$ can be described simply as the number of ring operations required, given ring elements a and b, to find ring elements x, y, l, k, g such that g is a gcd of a and b, with

$$\begin{aligned} g &= ax + by \\ 1 &= kx + ly \\ 0 &= -al + bk, \end{aligned} \tag{4}$$

or in other words, to find a unimodular matrix $\begin{bmatrix} x & -l \\ y & k \end{bmatrix}$ such that

$$[a\ b] \begin{bmatrix} x & -l \\ y & k \end{bmatrix} = [g\ 0]\,.$$

The assumptions of a PIR guarantee that the equations (4) are solvable (see [30, Chapter IV, Theorem 33]) but under these general assumptions it is not clear that there is an algorithm in terms of basic ring operations for finding a solution. (Thus it might be the case that $T(1,2)$ is infinite.) However, if the ring R is the homomorphic image of a Euclidean ring E then an algorithm for solving (4) over the ring R would be to take any preimages of a and b in E, solve the equation there and then project back to R. The most important examples of this situation (for our purposes) are the rings $\mathbb{Z}/h\mathbb{Z}$. We refer the reader to Section 4 where this is used.

We assume throughout that

$$M(m,m,m) \ll m^{\theta}. \tag{5}$$

From (5) and block decompositions, we may easily deduce the following consequences:

$$M(m,n,n) \ll \begin{cases} mn^{\theta-1} & \text{if } m \geq n \\ n^2 m^{\theta-2} & \text{if } m < n. \end{cases} \tag{6}$$

The current record on the exponent θ is held by Coppersmith and Winograd [8], who proved that $\theta < 2.376$ holds for all fields, and an inspection of their proof reveals that in fact it holds for all commutative rings with identity. It is also clear that $\theta \geq 2$.

We can now state the primary result of this section.

THEOREM 3.1. *If R is a PIR, then*

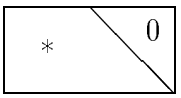
$$T(m,n) \ll \begin{cases} n^2 T(1,2) + mn^{\theta-1} & \text{if } m \geq n, \\ mnT(1,2) + m^{\theta-1} n \log(2n/m) & \text{if } m < n. \end{cases}$$

For the sake of comparison, note that the approach of Section 2.1 implies a bound of the form

$$T(m,n) \ll mnT(1,2) + mn^2, \quad m \leq n.$$

The proof of Theorem 3.1 uses a block decomposition to get an improvement on the second term. It is perhaps paradoxical that the output data includes a matrix K that is $n \times n$, but that the running time is almost linear in n. A careful examination of the proof reveals that the matrix K that is produced is sparse.

We will prove Theorem 3.1 by reducing the problem to a special case for which we require some notation. Let $\mathcal{S}_m$ be the set of $m \times 2m$ matrices A over R with $a_{ij} = 0$, for $1 \leq i \leq m-1$ and $m+i+1 \leq j \leq 2m$, that is matrices of the form



Let $T^*(m)$ be the number of ring operations required to produce for a given $A \in \mathcal{S}_m$ a primary triangularization. We now have the following lemma.

LEMMA 3.2. *We have*

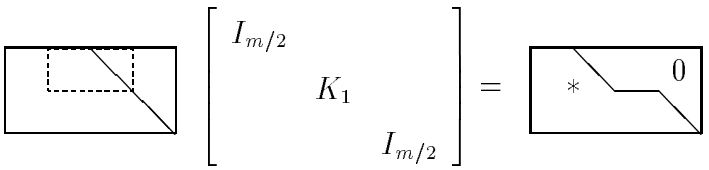
$$T^*(m) \ll m^2 T(1,2) + m^\theta.$$

Proof. By adjoining at most m rows and columns of zeros, we can assume that m is a power of 2 (this is because of the special form of the primary triangularization).

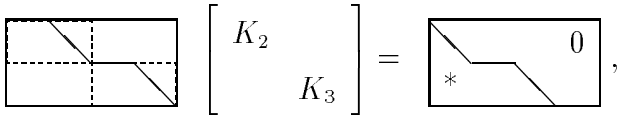
The proof proceeds inductively, by reducing the problem to several of half the size. We claim

$$\begin{aligned} T^*(m) &\leq 4T^*(m/2) + 4M(2m, 2m, 2m) \\ &\leq 4T^*(m/2) + cm^\theta, \end{aligned} \tag{7}$$

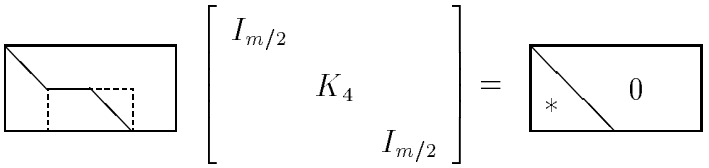
for some absolute constant c. The second part of this inequality is immediate from (5). To prove the inequality (7), we begin with a matrix $A \in \mathcal{S}_m$. The first step is to put the top center part (in the dashed box below) of the matrix into triangular form. This transforms the matrix as follows:



where K_1 is $m \times m$ unimodular. After this we triangularize the top left part and the lower right part, giving



where K_2 and K_3 are $m \times m$ unimodular. Finally we triangularize the bottom center part, which puts the entire matrix into triangular form:



where K_4 is unimodular.

To compute the product of the unimodular factors on the left requires at most an extra $4M(2m, 2m, 2m)$ operations. Totalling up all these operations we get

$$T^*(m) \leq 4T^*(m/2) + 4M(2m, 2m, 2m),$$

which proves the claimed inequality (7).

To complete the proof of the lemma, we iterate (7) to obtain

$$T^*(m) \leq 4T^*(m/2) + cm^\theta$$

$$
\begin{aligned}
&\leq 16T^*(m/4) + cm^\theta + 4c(m/2)^\theta \\
&\vdots \\
&\leq 4^{\log_2 m} T^*(1) + cm^\theta \sum_{j=0}^{\log_2 m} (\frac{4}{2^\theta})^j \\
&\ll m^2 T^*(1) + m^\theta .
\end{aligned}
$$

But trivially, $T^*(1) = T(1,2)$ and this proves the lemma. □

We now return to the proof of Theorem 3.1, starting with the case $m = n$. By adjoining m columns of zeros, we have the obvious inequality $T(m,m) \leq T^*(m)$, which by the lemma completes this case. Next, for the case $n < m$, we first put the top $n \times n$ submatrix of A in triangular form. This gives the obvious inequality,

$$T(m,n) \leq T(n,n) + M(m-n,n,n),$$

and by (6) this completes the case $n < m$.

It remains to consider the case $m < n$. By adding at most n columns of zeros, we may assume that $n = 2^k m$ for some positive integer k. Our goal will be to prove the inequality

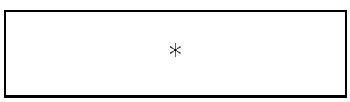
$$T(m,n) \leq 2T(m,n/2) + P(m,n) + T^*(m) + M(n,2m,2m), \tag{8}$$

where $P(m,n)$ denotes the time required to interchange two $m \times n$ blocks in an $n \times n$ matrix, so that clearly $P(m,n) \ll mn$. Before proving (8), we first deduce the theorem from it. From Lemma 3.2 and (6) we obtain

$$
\begin{aligned}
T(m,n) &\leq 2T(m,n/2) + c_1 m^2 T(1,2) + c_2 m^{\theta-1} n \\
&\vdots \\
&\leq 2^k T(m,m) + \sum_{j=0}^{k} 2^j \left(c_1 m^2 T(1,2) + c_2 m^{\theta-1} \frac{n}{2^j} \right) \\
&\ll mnT(1,2) + km^{\theta-1} n,
\end{aligned}
$$

which is the conclusion of the theorem.

The proof of inequality (8) should be clear from the following diagram:

A = 

$$A\begin{bmatrix} K_1 & \\ & K_2 \end{bmatrix} = \begin{bmatrix} * & 0 & * & 0 \end{bmatrix}$$

$$A\begin{bmatrix} K_1 & \\ & K_2 \end{bmatrix}P = \begin{bmatrix} * & 0 & * & 0 \end{bmatrix}$$

$$A\begin{bmatrix} K_1 & \\ & K_2 \end{bmatrix}P\begin{bmatrix} K_3 & \\ & I \end{bmatrix} = \begin{bmatrix} * & 0 \end{bmatrix},$$

where P is an $n \times n$ unimodular matrix which interchanges two $m \times n$ blocks of an $n \times n$ matrix, K_1 and K_2 are $n/2 \times n/2$ unimodular, and K_3 is $2m \times 2m$ unimodular. The term $M(n, 2m, 2m)$ in (8) comes from computing the products of the unimodular factors to the right of A. This completes the proof of (8), and therefore of Theorem 3.1.

3.2. Triangular Factorization Over Principal Ideal Domains. In this section we impose an additional assumption on our ring and give two results about matrices over the ring. More precisely, we shall assume throughout this section that the ring R is also an integral domain, hence a Principal Ideal Domain (PID). In our operation counts, we include division as a ring operation, with the following stipulation. If $a, b \in R$ and $a \mid b$, then we allow the "operation" b/a to return the $k \in R$ with $b = ka$.

Let $F(m, n) = F_R(m, n)$ be the number of ring operations (including divisions when possible) required to compute for an $m \times n$ matrix A a factorization $A = LK$, where L is lower triangular and K is unimodular of the form (3). Note that $F(1, 2)$ differs very little from $T(1, 2)$, except that the k and l in (4) can be recovered from g, x, and y via $k = a/g$ and $l = b/g$. Finding g, x, and y requires solving the "extended Euclidean problem" $\gcd(a, b) = g = ax + by$ (but of course we do not have a Euclidean algorithm unless R is Euclidean).

We will show that a factorization $A = LK$ is possible with essentially the same running time as in Theorem 3.1. Some result of this type would be obvious if one could reduce the problem of matrix inversion to that of matrix multiplication. In the case when R is a field, this was proved by Bunch and Hopcroft [6]. The following extends that result to integral domains.

THEOREM 3.3. *Let R be an integral domain, and let $I(m) = I_R(m)$ denote the number of ring operations required to compute the inverse of an $m \times m$ invertible matrix*

A over R. If (5) holds, then

$$I(m) \ll m^{\theta}.$$

The proof of this result over a field given by Bunch and Hopcroft [6] does not appear to extend directly to integral domains. Specifically, their proof uses the fact that if a matrix over a field is invertible, then every row of the matrix contains a unit. This is not the case for an arbitrary ring, as can be seen from the simple example $\left[\begin{smallmatrix} 2 & 3 \\ 3 & 5 \end{smallmatrix}\right]$ over $\mathbb{Z}$.

Proof. In order to prove Theorem 3.3, we shall pass to the field of fractions of R (see [15, pp. 51]). Briefly, this field can be defined as follows. Define an equivalence relation $\sim$ on the set of pairs (r, s) of elements of R with $s \neq 0$ by

$$(r, s) \sim (t, u) \text{ if and only if } ru = st.$$

The field of fractions then consists of the set of equivalence classes, with addition and multiplication defined by

$$\begin{aligned} [(r, s)] + [(t, u)] &= [(ru + st, su)] \\ [(r, s)] \cdot [(t, u)] &= [(rt, su)], \end{aligned}$$

so that the operations of the field are expressed directly in terms of R.

The first thing we need to show is that the exponent for matrix multiplication θ_F over the field of fractions F is the same as that for matrix multiplication over the ring R, namely θ in our notation. Let A and B be two $m \times m$ matrices over F and let a and b be two ring elements such that aA and bB are matrices over R (for example take a to be the product of the denominators of all entries in A). Then the product AB can be computed by first computing $(aA)(bB)$ as matrices over R and then dividing every entry by ab. This requires at most $O(m^2)$ multiplications and divisions to compute aA and bB and AB from $(aA)(bB)$. This is of lower order than the computation of $(aA)(bB)$ over R and shows that $\theta_F \leq \theta$. Conversely, let A and B be matrices over R. Interpret these matrices as defined over F by taking each entry x and replacing it by $[(x, 1)]$ in F. Then the product AB can be computed in F. The final step is to reinterpret the entries of the product as elements of R, by dividing each numerator by its corresponding denominator. This requires an extra $O(m^2)$ operations (actually divisions) and proves the inequality $\theta \leq \theta_F$.

Let F be the field of fractions of R, and let A be an invertible $m \times m$ matrix over R. There is a natural way of viewing the matrices A and A^{-1} as matrices over F, and according to the result of Bunch and Hopcroft, we can compute A^{-1} in $O(m^\theta)$ operations over F, so $O(m^\theta)$ operations in R. Once A^{-1} is computed over F, it remains only to interpret the result as a matrix over R, but this can easily be done since we know that the entries in A^{-1} must be of the form $[(r,s)]$ where $s \mid r$. Hence it suffices to take the elements $[(r,s)]$ and replace them by the corresponding value r/s in R. This completes the proof. □

We now state our result on triangular factorization over PID's.

THEOREM 3.4. *If R is a PID, then*

$$F(m,n) \ll \begin{cases} n^2 F(1,2) + mn^{\theta-1} & \text{if } m \geq n, \\ mnF(1,2) + m^{\theta-1} n \log(2n/m) & \text{if } m < n. \end{cases}$$

Note that a straight application of Theorems 3.1 and 3.3 would add an extra factor of n^θ to the above bound.

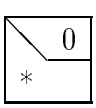
The proof of this is not much different than that of Theorem 3.1. We first define a quantity $F^*(m)$ to be the number of ring operations required to compute the factorization of a matrix in $\mathcal{S}_m$. Next we claim

$$F^*(m) \leq 4F^*(m/2) + 2I(m) + 4M(2m, 2m, 2m). \tag{9}$$

To see that the claim holds, we need an intermediate observation. Let $X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ be $m \times m$, with $X_1 \in \mathcal{S}_{m/2}$. If $X_1 = L_1 K_1$ is a factorization into lower triangular times unimodular, then

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} L_1 K_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} L_1 \\ X_2 K_1^{-1} \end{bmatrix} K_1, \tag{10}$$

where now $\begin{bmatrix} L_1 \\ X_2 K_1^{-1} \end{bmatrix}$ has the shape



.

This procedure requires $F^*(m/2) + I(m) + M(m/2, m, m)$ operations.

With this in hand, the proof of (9) is essentially the same as (7) except that the unimodular factors are on the right sides of the equations.

We deduce from (9) and Theorem 3.3 the simple bound

$$F^*(m) \ll m^2 F(1,2) + m^\theta.$$

This and arguments similar to those in the proof of Theorem 3.1 are all that is needed to complete the proof of Theorem 3.4. The only differences are that the unimodular factors are again on the right sides of the equations and the case $n < m$ requires an inversion of an $n \times n$ unimodular matrix.

4. Triangularization over $\mathbb{Z}$.. One application of the Hermite normal form is to give a particular triangular basis for the lattice generated by the columns of an integral matrix. In this section we prove a result that produces a triangular basis with nice properties. It is stronger than the result of Theorem 2.1 in one sense since the running time is smaller, but weaker since it does not produce the unique Hermite normal form. The output still has the property that the entries are moderate in size, so it is useful in some lattice problems. The algorithm ties together methods similar to Corollary 2.2 and the results of Section 3.1.

In this section we let θ denote an exponent for which (5) holds for $\mathbb{Z}/N\mathbb{Z}$ for every integer $N > 1$. In particular, $\theta < 2.376$ by [8]. Our result is the following.

THEOREM 4.1. *There exists a deterministic algorithm that receives as input an $m \times n$ integral matrix A of rank m, and produces as output a triangular matrix L that is right unimodularly equivalent to A. If A has entries bounded in absolute value by T, then the entries of L satisfy $0 \le l_{ij} < m^{m/2}T^m$, and the running time of the algorithm is $O(m^{\theta-1}n\log(2n/m)B(m\log(mT)))$ bit operations.*

Proof. The algorithm has three stages. In the first stage, we find the determinant h of a nonsingular submatrix of the matrix A in a manner similar to Section 2.1, but using fast matrix multiplication techniques. In the second stage, we use the algorithm of Theorem 3.1 over $\mathbb{Z}/h\mathbb{Z}$ to find a triangular matrix L' that is right equivalent to A over $\mathbb{Z}/h\mathbb{Z}$. Finally in the third stage we reconstruct from L' a triangular matrix L that is right equivalent to A over $\mathbb{Z}$.

To begin the first stage of the algorithm, we proceed as in the algorithm of Theorem 2.1 to calculate the number $z = O(m\log(mT))$ and all of the primes $p \le z$. Now we search for a prime $p \le z$ for which the rank of A over $\mathbb{Z}/p\mathbb{Z}$ is m. For this, we use an

algorithm of Ibarra, Moran, and Hui [19]. Their algorithm gives (among other things) the rank of A and a maximal set of linearly independent columns over $\mathbb{Z}/p\mathbb{Z}$. As soon as we find such a prime p, the set of m linearly independent columns over $\mathbb{Z}/p\mathbb{Z}$ provide our $m \times m$ nonsingular submatrix of A when viewed over $\mathbb{Z}$. The running time of their algorithm is $O(m^{\theta-1}n)$ operations in $\mathbb{Z}/p\mathbb{Z}$, for a total of at most

$$\ll m^{\theta-1}n \sum_{p \leq z} B(\log p) \ll m^{\theta-1}nB(z)$$

bit operations.

We now calculate the determinant of this submatrix using the Chinese remainder theorem and the calculation of the determinant modulo all of the primes $p \leq z$. The calculation of the determinants modulo p can again be accomplished with the algorithm of [19], with a total running time of $O(m^{\theta}B(z))$ bit operations, and the Chinese remainder calculation takes $O(B(z))$ bit operations as before.

After this first stage, we have a number $h \leq m^{m/2}T^m$ that is a multiple of $\mathrm{Det}(\mathcal{L}(A))$. In the second stage we now apply the algorithm of Theorem 3.1 over the ring $\mathbb{Z}/h\mathbb{Z}$ (which is a PIR) to find matrices K' and L' over $\mathbb{Z}/h\mathbb{Z}$ such that $AK' = L'$, K' is unimodular, and L' is lower triangular. The only thing we need to describe is how many bit operations are required to do $T(1,2)$. Since this can be done by the extended Euclidean algorithm in $\mathbb{Z}$ on nonnegative integers bounded by h, this is $B(\log h)$. The number of bit operations required for this stage is $O(m^{\theta-1}n \log(2n/m)B(\log h))$.

Finally in the third stage we proceed as in the proof of Theorem 2.1 to reconstruct from L' a matrix L whose diagonal is the diagonal of the Hermite normal form of A, and whose columns generate $\mathcal{L}(A)$. Note that in this procedure, the entries are reduced modulo some divisor of h (the d_is) and so will be bounded between 0 and h. The number of operations for this stage is $O(m^2B(\log h))$, since we need m applications of the extended Euclidean algorithm on integers $\leq h$, combined with $O(m^2)$ operations modulo h. This completes the proof of Theorem 4.1. □

5. A Fast Probabilistic Method for Smith Forms over $\mathbb{Z}$, and Some Open Questions.. The results of the previous section suggest an interesting problem, namely whether algorithms for constructing the Smith normal form can be constructed that make use of fast matrix multiplication techniques, achieving a speedup over the methods

presented in Section 2.2. Our results in Section 4 come close to achieving this for the Hermite normal form, although we are only able to triangularize the matrix. We were unable to find a block decomposition of the Smith normal form that would allow us to incorporate fast matrix multiplication methods, but if one is willing to consider probabilistic methods, then the method of Kaltofen, Krishnamoorthy, and Saunders [21] might be adapted to the problem at hand.

We will briefly describe such a probabilistic algorithm for computing the Smith normal form, based on the method of Kaltofen, Krishnamoorthy, and Saunders. Let A be an $m \times m$ nonsingular integer matrix for which we wish to compute the Smith normal form, and for a matrix B, let B_i denote the upper left $i \times i$ subdeterminant of B. The algorithm uses a parameter k, and can be described as follows.

1. For $i = 1, \ldots, m$, set $g_i = A_i$.
2. For $i = 1, \ldots, k$, do
   (a) choose random $m \times m$ unimodular matrices X and Y.
   (b) calculate $G = XAY$.
   (c) set $g_i = \gcd(G_i, g_i)$, $1 \leq i \leq m-1$,
3. output the matrix $T = \mathrm{diag}(t_1, \ldots, t_m)$, where $t_1 = g_1$ and $t_i = g_i/g_{i-1}$, $2 \leq i \leq m$.

We should mention that in practice we can take X and Y to be triangular, or the product of an upper and a lower triangular matrix. The principle upon which the algorithm is based is the following. In Lemma 2.4, each γ_i is a gcd of all $i \times i$ subdeterminants of the matrix A. We should be able to calculate this gcd by taking the gcd of a few "random" linear combinations of the $i \times i$ subdeterminants, and it turns out that the G_is are good candidates for such random linear combinations.

The method of Kaltofen, Krishnamoorthy, and Saunders was originally stated for the problem of computing Smith normal forms of matrices over the polynomial ring $F[x]$, where F is a field. In this case they were able to prove that the algorithm has a good probability of producing the correct output, and thus were able to prove that the problem of computing the Smith form over $\mathbb{Q}[x]$ or $GF(p)[x]$ is in RNC^2. The method of proof that they use does not carry over to the integer case, but we suspect that the probability of failure in the above algorithm decreases expo-

nentially with k. If this can be proved, then we could probably achieve an expected running time of $O(m^3 B(m \log(mT)))$ bit operations using standard matrix multiplication. Using fast matrix multiplication methods, we might even be able to lower it to $O(m^\theta B(m \log(mT)))$.

It is perhaps interesting to note that even though we are not able to prove it, the method given above seems to work very well in practice for integer matrices. It does however sometimes produce a wrong answer which may be difficult to detect. In some cases, incorrect outputs can be detected immediately because they may not have the correct divisibility property or even be integral. There seems to be no immediate way to *always* detect a wrong answer other than to run some independent algorithm such as that in Theorem 2.6 and compare the answers.

We note that our algorithms for computing the Hermite and Smith normal forms of matrices over $\mathbb{Z}$ do not produce the unimodular multipliers. For square matrices of full rank the multipliers can be recovered with the same asymptotic running time bounds (see for example [20]). For matrices that are not square or not of full rank, the multipliers are not unique. The algorithms that are presented here can be modified to produce these multipliers, but at the expense of increased running time. One technique for doing this that was pointed out to us by H.W. Lenstra, Jr., is to permute the columns of A so that $A = [A_1 : A_2]$ where A_1 is $m \times m$ and nonsingular. We then apply the algorithm to the augmented matrix

$$A^* = \begin{bmatrix} A_1 & A_2 \\ 0 & I_{n-m} \end{bmatrix}.$$

This matrix is clearly square and nonsingular, and any unimodular multiplier that triangularizes A^* will also work for A.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] W. A. Blankinship, *Algorithm 287, matrix triangulation with integer arithmetic*, Communications of the ACM, 9 (1966), p. 513.

[3] ——, *Algorithm 288, solution of simultaneous linear Diophantine equations*, Communications of the ACM, 9 (1966), p. 514.

[4] I. Borosh and A. S. Fraenkel, *Exact solutions of linear equations with rational coefficients by congruence techniques*, Mathematics of Computation, 20 (1966), pp. 107–112.

[5] G. H. Bradley, *Algorithms for Hermite and Smith normal form matrices and linear Diophantine equations*, Mathematics of Computation, 25 (1971), pp. 897–907.

[6] J. R. Bunch and J. E. Hopcroft, *Triangular factorization and inversion by fast matrix multiplication*, Mathematics of Computation, 28 (1974), pp. 231–236.

[7] T.-W. J. Chou and G. E. Collins, *Algorithms for the solution of systems of linear Diophantine equations*, SIAM Journal of Computing, 11 (1982), pp. 687–708.

[8] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions.* preprint, May, 1987, Extended abstract in *Proc. 19th ACM Symposium on Theory of Computing* (1987), 1–6.

[9] P. D. Domich, *Residual Methods for Computing Hermite and Smith Normal Forms*, PhD thesis, Cornell University, 1985.

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

[11] M. A. Frumkin, *An application of modular arithmetic to the construction of algorithms for solving systems of linear equations*, Soviet Mathematics Doklady, 17 (1976), pp. 1165–1168.

[12] ——, *Polynomial time algorithms in the theory of linear Diophantine equations*, in Fundamentals of Computation Theory, Springer-Verlag, New York, 1977, pp. 386–392. Lecture Notes in Computer Science, volume 56.

[13] ——, *Complexity questions in number theory*, Journal of Soviet Mathematics, 29 (1985), pp. 1502–1517.

[14] J. L. Hafner and K. S. McCurley, *A rigorous subexponential algorithm for computation of class groups*, Journal of the American Mathematical Society, (1989). to appear.

[15] B. Hartley and T. O. Hawkes, *Rings, Modules, and Linear Algebra*, Chapman and Hall, London, 1970.

[16] G. Havas and L. S. Sterling, *Integer matrices and abelian groups*, in Lecture Notes in Computer Science, vol. 72, Springer-Verlag, New York, 1979, pp. 431–451.

[17] C. Hermite, *Sur l'introduction des variables continues dans la théorie des nombres*, Journal für die Reine und Angewandte Mathematik, 41 (1851), pp. 191–216.

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

[19] O. H. Ibarra, S. Moran, and R. Hui, *A generalization of the fast LUP matrix decomposition algorithm and applications*, Journal of Algorithms, 3 (1982), pp. 45–56.

[20] C. S. Iliopolous, *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 (1989), pp. 658–669.

[21] E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders, *Fast parallel computation of Hermite and Smith forms of polynomial matrices*, SIAM Journal of Algebraic and Discrete Methods, 8 (1987), pp. 683–690.

[22] M. Kaminski and A. Paz, *Computing the Hermite normal form on an integral matrix*, Technical Report 417, Technion Department of Computer Science, 1986.

[23] R. Kannan and A. Bachem, *Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix*, SIAM Journal of Computing, 8 (1979), pp. 499–507.

[24] M. T. McClellan, *The exact solution of systems of linear equations with polynomial coefficients*, Journal of the ACM, 20 (1973), pp. 563–588.

[25] P. Pritchard, *Fast compact prime number sieves (among others)*, Journal of Algorithms, 4 (1983), pp. 332–344.

[26] J. B. Rosser and L. Schoenfeld, *Sharper bounds for the Chebyshev functions $\theta(x)$ and $\psi(x)$*, Mathematics of Computation, 29 (1975), pp. 243–269.

[27] A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing, 7 (1971), pp. 281–292.

[28] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley, New York, 1985.

[29] H. J. S. Smith, *On systems of linear indeterminate equations and congruences*, Philosophical Transactions of the Royal Society of London, 151 (1861), pp. 293–326. See also pages 367–409 in *The Collected Papers of Henry John Steven Smith*, Volume I, Chelsea, New York, 1965.

[30] O. Zariski and P. Samuel, *Commutative Algebra*, vol. I, Van Nostrand, Princeton, 1958.