MATHEMATICS OF OPERATIONS RESEARCH
Vol. 12, No. 1, February 1987
Printed in U.S.A.

HERMITE NORMAL FORM COMPUTATION USING MODULO DETERMINANT ARITHMETIC*†

P. D. DOMICH,‡ R. KANNAN§ AND L. E. TROTTER, JR.**

This paper describes a new class of Hermite normal form solution procedures which perform modulo determinant arithmetic throughout the computation. This class of procedures is shown to possess a polynomial time complexity bound which is a function of the length of the input string. Computational results are also given.

1. Introduction. Throughout we assume that $A \equiv [a_{ij}]$ is a matrix with integer-valued entries. For convenience we also assume that A is $n \times n$ and nonsingular; the restatement of the results presented here for arbitrary integral matrices is straightforward. We denote $d \equiv |\det(A)| > 0$ and use $H \equiv [h_{ij}]$ to denote the Hermite normal form of A (see (1.2) below).

A matrix with integral entries and determinant of unit magnitude is *unimodular*. For example, matrices which transform A as prescribed by the following *elementary column operations* are unimodular.

1.1. *Elementary operations.* (a) *Interchange two columns.*
(b) *Multiply a column by* -1.
(c) *Add an integral multiple of one column to another.* ∎

If two elements in the same row of A are positive, operation (1.1.c) can be used to reduce the larger modulo the smaller. Thus by working in row order and from the diagonal to the right within each row, a finite number of operations (1.1.a–c) will transform A into lower triangular form. Additional operations of type (1.1.c) can then be used, again in row order, to reduce subdiagonal entries in a given row modulo the corresponding diagonal entry. Since the product of unimodular matrices remains unimodular, the process just described provides a constructive proof of the following classical theorem of Hermite [1851].

1.2. THEOREM. *Given an* $n \times n$ *nonsingular integer-valued matrix* A, *there exists an* $n \times n$ *unimodular matrix* K *such that* $AK = H$, *the Hermite normal form of* A, *whose entries satisfy*

$$\begin{aligned} &h_{ij} = 0, && \forall j > i, \\ &h_{ii} > 0, && \forall i, \\ &h_{ij} \leqslant 0 \quad \text{and} \quad |h_{ij}| < h_{ii}, && \forall j < i. \quad ∎ \end{aligned}$$

*Received March 27, 1985; revised October 29, 1985.
AMS 1980 subject classification. Primary: 65F30. Secondary: 90C10.
IAOR 1973 subject classification. Main: Numerical analysis. Cross Reference: Programming: Integer.
OR/MS Index 1978 subject classification. Primary: 437 Mathematics/Matrices. Secondary: 627 Programming/integer.
Key words. Unimodular matrix, Hermite normal form.
† The research was partially supported by NFS Grant ECS 8113534 and NSF Grant DCR 8416190.
‡ National Bureau of Standards.
§ Carnegie-Mellon University.
** Cornell University.

0364-765X/87/1104/0050$01.25

In fact the matrix H specified in (1.2) is the unique matrix with entries satisfying the stipulations of (1.2) which can be obtained from A by a unimodular transformation (see (2.1) below). One particularization of the algorithmic process described above was proposed by Rosser [1952] for computing H.

Any unimodular matrix is a product of elementary matrices corresponding to operations (1.1.a–c). This well-known fact is a corollary of the proof sketched above for Hermite's theorem: when A is unimodular, then H is the $n \times n$ identity matrix I, so there exist elementary matrices $K_1, \dots, K_t$ for which $AK_1 \cdots K_t = I$; hence $A = K_t^{-1} \cdots K_1^{-1}$ and each K_i^{-1} is also elementary.

By performing column operations which are composites of operations (1.1.a–c), one can compute H using no more than n^2 column operations. Suppose, for example, that a_{ij} and a_{ik} (where $j < k$) are nonzero with greatest common divisor g.c.d. $(a_{ij}, a_{ik}) = g$. Then there exist integers p, q such that $g = pa_{ij} + qa_{ik}$ and a unimodular transformation sending a_{ij} to g and a_{ik} to 0 is obtained by embedding the matrix

$$\begin{bmatrix} p & -a_{ik}/g \\ q & a_{ij}/g \end{bmatrix}$$

into the positions indexed by j and k in the $n \times n$ identity. An algorithm proposed by Bradley [1971] computes H using this type of transformation, again working in row order and from the diagonal to the right during the triangularization of A. Such an algorithm clearly requires no more than n^2 such column operations, each altering at most $2n$ entries of A. The indicated g.c.d. computations can be performed in polynomial time (Lamé 1844), so it is tempting to conclude (see, e.g., Sahni 1974) a polynomial time bound for the entire computational procedure.

Such an analysis, however, neglects the size of the numbers encountered during the course of the computation. Frumkin [1977] points out that certain matrix entries may square as a result of column operations as indicated above; this suggests the possibility (though no specific example is known) that a given entry, say a_{ij}, could square at each successive iteration, thereby producing entries of order $|a_{ij}|^{2^n}$. The issue is not merely theoretical, for empirical evidence indicates that growth in the size of matrix entries severely limits standard approaches to Hermite normal form computation. For example, in §4 we report that intermediate data of magnitude exceeding 2^{432} were encountered when a variant of the Bradley [1971] procedure was used to compute the Hermite normal form of an 8×8 matrix whose initial entries were all smaller than 2^{16} (see Table 4.1, Algorithm BRADLEY). This phenomenon has been termed "intermediate expression swell" by McClellan [1973] and "entry explosion" by Havas and Sterling [1979]; see also Rosser [1952], Smith [1966] and Sims [1974].

Kannan and Bachem [1979] proposed altering the flow of computation in the Bradley [1971] algorithm so that successively the $1 \times 1, 2 \times 2, \dots, n \times n$ principal submatrices of A are placed in Hermite normal form and they showed that the resulting algorithm is polynomially time bounded—*both* in the number of binary digits needed to solve the problem *and* in the number of arithmetic operations performed. But this "good" algorithm remains subject to intermediate expression swell from a practical point of view (see Table 4.1).

In this paper we present algorithms for Hermite normal form computation in which all arithmetic operations are performed modulo $d(\equiv |\det(A)|)$. Since $d \leqslant n!a^n \leqslant (na)^n$, where $a \equiv \max_{i,j}|a_{ij}|$, it follows that the number of bits needed to represent any matrix entry during the computation is bounded above by $n(\log_2 n + \log_2 a)$. The algorithms discussed here are all polynomially time bounded and the space bound indicated above represents a significant improvement over the bound of order $n^3(\log_2 n + \log_2 a)$ given for the Kannan and Bachem [1979] algorithm. In Chou [1979] a refinement of the Kannan and Bachem algorithm is reported which also has an order $n(\log_2 n + \log_2 a)$ bound, though this algorithm is entirely different from those developed below.

The present work grew out of observations by R. Kannan and H. K. Lenstra [1982] concerning the use of modulo arithmetic for Hermite normal form computation. Our development here follows that in Domich [1983, 1985]. We point out also that we have recently learned that the simple mod d algorithm which we present below in §3 has been independently discovered by A. Schrijver (see Schrijver [1985]).

2. Algebraic preliminaries. We summarize in this section several elementary facts which provide the algebraic background used to validate the algorithms presented later. The *lattice* generated by matrix A, denoted $L(A)$, is the set of all integral linear combinations of the columns of A; i.e., $L(A) \equiv \{Ax : x \text{ integral}\}$. A *basis* for $L(A)$ is any linearly independent subset of $L(A)$ which generates A (and hence $L(A)$) via integer linear combinations. Observe that since $L(A)$ has rank n and the vectors in a basis are independent, it follows that all bases for $L(A)$ are of size n. Thus we may associate with any basis of $L(A)$ an $n \times n$ nonsingular matrix whose columns are in $L(A)$. H, for example, provides such a basis, since H is obtained by a unimodular transformation of A, say $H = AK$. Consequently K^{-1} is unimodular, and so $Ax = HK^{-1}x = Hy$, with x integral if and only if $y \equiv K^{-1}x$ is integral. It follows (see (2.2) below) easily from uniqueness properties of H, that any two bases of $L(A)$ are related, as just seen with H and A, by a unimodular transformation.

2.1. PROPOSITION. *H provides the unique basis for $L(A)$ as stipulated by Theorem 1.2.*

PROOF. Suppose matrices G, H each satisfy the conditions of (1.2) and $L(G) = L(A) = L(H)$. Since $L(G) = L(H)$, g_{nn} must be an integral multiple of h_{nn}, and vice-versa; thus $g_{nn} = h_{nn}$. Assume inductively that columns $n, n-1, \ldots, j+1$ are identical in G and H. Since the jth column of G is in $L(H)$, there exist integers $t_j, \ldots, t_n$ so that

$$g_{ij} = t_j h_{ij} + \cdots + t_n h_{in}, \qquad j \leqslant i \leqslant n.$$

For $i = j$ this relation requires $g_{jj} = t_j h_{jj}$ with t_j integral and it again follows as above that $g_{jj} = h_{jj}$; i.e., $t_j = 1$. For $i = j+1$ we thus have $g_{j+1,j} = h_{j+1,j} + t_{j+1}h_{j+1,j+1}$, implying that $h_{j+1,j+1}$ divides the quantity $g_{j+1,j} - h_{j+1,j}$. But then since $|g_{j+1,j}| < g_{j+1,j+1} = h_{j+1,j+1}$ and $|h_{j+1,j}| < h_{j+1,j+1}$, we must have $g_{j+1,j} = h_{j+1,j}$; i.e., $t_{j+1} = 0$. Similarly $t_{j+2} = \cdots = t_n = 0$, showing that columns j of G and H are identical and completing the induction. ■

2.2. COROLLARY. *Any two bases of $L(A)$ are related by a unimodular transformation.*

PROOF. Suppose that the columns of matrices B, C provide bases for $L(A)$. Then (2.1) implies that $H = BU = CV$, where U and V are unimodular. Hence $B = C(VU^{-1})$, and VU^{-1} is also a unimodular matrix. ■

A further consequence of the proof of (2.1) is the following condition sufficient for determining a basis of $L(A)$.

2.3. COROLLARY. *Let G be a lower triangular $n \times n$ matrix whose columns are in $L(A)$ and whose diagonal entries satisfy $g_{jj} = h_{jj}$, $1 \leqslant j \leqslant n$. Then G provides a basis for $L(A)$.*

PROOF. By applying unimodular column operations to G we may obtain a lower triangular matrix G' whose columns are in $L(A)$ and whose entries satisfy $g'_{ij} \leqslant 0$ and $|g'_{ij}| < g'_{jj} = h_{jj}$, for $1 \leqslant j < i \leqslant n$. The argument given in the proof of (2.1) now shows that $G' = H$, which implies that G gives a basis for $L(A)$. ■

We shall also need the following consequence of (2.1).

2.4. COROLLARY. *Suppose M is an $n \times m$ matrix whose columns are in $L(A)$ and define matrices*

$$A' \equiv \left[\begin{array}{c|c} A & M \\ \hline 0 & I \end{array}\right], \quad H' \equiv \left[\begin{array}{c|c} H & 0 \\ \hline 0 & I \end{array}\right],$$

where I denotes the $m \times m$ identity. Then H' is the Hermite normal form of A' and, conversely, H' determines H, the Hermite normal form of A.

PROOF. Suppose $AK = H$, where K is unimodular, and let Q be an $n \times m$ integral matrix for which $AQ = M$. Then

$$K' \equiv \left[\begin{array}{c|c} K & -Q \\ \hline 0 & I \end{array}\right]$$

is unimodular and $A'K' = H'$. Thus (2.1) implies H' must be the Hermite normal form of A'. On the other hand, uniqueness of H' also implies that its $n \times n$ principal submatrix must be H. ■

Corollary 2.4 suggests that arbitrary vectors of $L(A)$, when taken as the column of matrix M, may be used to operate on the columns of A in determining H. Certain vectors in $L(A)$ can be used in this way to perform mod d operations on the entries of A. In particular, we note that since dA^{-1} is an integral matrix (by Cramer's rule) and $A(dA^{-1}) = dI$, the following proposition holds.

2.5. PROPOSITION. *For $1 \leqslant i \leqslant n$, $de_i \in L(A)$, where e_i denotes the ith unit vector.* ■

The result of (2.5) can be sharpened as follows.

2.6. COROLLARY. *Define $d_1 = d$ and $d_{i+1} = d_i/h_{ii}$ for $i = 1, 2, \ldots, n-1$. Then for $1 \leqslant i \leqslant n$, $d_i e_i \in L(A)$, where e_i denotes the ith unit vector.*

PROOF. For $i = 1$, Proposition 2.5 applies. Then (2.5) may be applied to the $(n-1) \times (n-1)$ submatrix of H obtained by deleting the first row and column; thus $d_2 e_2 = (d_1/h_{11})e_2 \in L(H) = L(A)$. And so on. ■

Finally we show that mod d arithmetic during Hermite normal form computation does not change certain basic quantities associated with matrix A. This invariance leads to a means for "reconstructing" H from any lower triangular matrix G obtained from A via operations (1.1.a–c) and mod d reduction. Let $\gamma_i(A)$ denote the g.c.d. of all determinants of $i \times i$ submatrices contained within rows $1, \ldots, i$ of A.

2.7. PROPOSITION. *Suppose G results from applying to A a sequence of operations (1.1.a–c), reducing arbitrary entries modulo d during the process. Then g.c.d. $(d, \gamma_i(A))$ = g.c.d. $(d, \gamma_i(G))$ for $1 \leqslant i \leqslant n$.*

PROOF. It is obvious that (1.1.a–b) have no effect on $\gamma_i(\cdot)$ and it is straightforward to verify that (1.1.c) also leaves $\gamma_i(\cdot)$ unchanged. Mod d reduction of an entry affects determinant values linearly and the result follows, since

$$\text{g.c.d.}(d, p_1 + q_1 d, \ldots, p_k + q_k d) = \text{g.c.d.}(d, p_1, \ldots, p_k). \quad ■$$

2.8. Corollary. *If matrix G of* (2.7) *is lower triangular, then* $h_{ii} = \text{g.c.d.}(d_i, g_{ii})$, $1 \leqslant i \leqslant n$, *with* d_i *as in* (2.6).

Proof. By (2.7), $\text{g.c.d.}(d, \gamma_i(G)) = \text{g.c.d.}(d, \gamma_i(A)) = \text{g.c.d.}(d, \gamma_i(H))$, $1 \leqslant i \leqslant n$. Now $\gamma_i(G) = g_{11} \cdots g_{ii}$ and, since $\gamma_i(H) = h_{11} \dots h_{ii}$ divides d, we have $\text{g.c.d.}(d, g_{11} \cdots g_{ii}) = h_{11} \dots h_{ii}$. For $i = 1$, the desired result follows and for $i > 1$ we observe that $\text{g.c.d.}(d, g_{11} \dots g_{i-1,i-1}) = h_{11} \dots h_{i-1,i-1}$ implies $\text{g.c.d.}(d_i, g_{11} \cdots g_{i-1,i-1}/h_{11} \cdots h_{i-1,i-1}) = 1$. Thus we obtain that

$$\begin{aligned} h_{ii} &= \text{g.c.d.}(d, g_{11} \cdots g_{ii})/h_{11} \cdots h_{i-1,i-1} \\ &= \text{g.c.d.}(d_i, (g_{11} \cdots g_{i-1,i-1}/h_{11} \cdots h_{i-1,i-1}) g_{ii}) \\ &= \text{g.c.d.}(d_i, g_{ii}). \quad \blacksquare \end{aligned}$$

3. Mod d algorithms. We now indicate several procedures for computing H using mod d arithmetic. The first is a particularly simple algorithm suggested by Corollary 2.4 and Proposition 2.5. Note that these results imply that the matrix $D \equiv dI$ has columns in $L(A)$ and hence that the Hermite normal form of A' is H', where

$$A' \equiv \left[\begin{array}{c|c} A & D \\ \hline 0 & I \end{array}\right] \quad \text{and} \quad H' \equiv \left[\begin{array}{c|c} H & 0 \\ \hline 0 & I \end{array}\right].$$

Thus H can be determined by computing the first n rows of H' and we focus attention on the matrix $\bar{A} \equiv [A|D]$. We now compute H by applying the "obvious" Hermite normal form procedure to $\bar{A}$ using the columns of D to perform mod d operations on the remaining matrix entries.

Recall the elementary operations (1.1.a–c) and the composite (g.c.d.) column operation discussed in the introduction. By applying these operations as determined by the entries in row 1 of $\bar{A}$, we transform $\bar{A}$ into the following form:

$$\bar{A}_1 \equiv \left[\begin{array}{cccc|cccc} h_{11} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ * & * & \cdots & * & * & d & & \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \bigcirc \\ * & * & \cdots & * & * & \bigcirc & & d \end{array}\right],$$

where $*$ denotes an arbitrary integral entry. Note that operation (1.1.c) may be used with the de_j columns to ensure that no entry in the course of this computation exceeds d in magnitude. We continue by applying column operations as determined by the $(2,2), \dots, (2, n+2)$ entries of $\bar{A}_1$ to obtain

$$\left[\begin{array}{ccccc|ccccc} h_{11} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ * & h_{22} & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ * & * & * & \cdots & * & * & * & d & & \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \ddots & \bigcirc \\ * & * & * & \cdots & * & * & * & \bigcirc & & d \end{array}\right].$$

Operation (1.1.c) may now be used to reduce the (2,1) entry modulo h_{22}, thereby producing

$$\bar{A}_2 \equiv \left[\begin{array}{ccccc|ccccc} h_{11} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ h_{21} & h_{22} & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ * & * & * & \cdots & * & * & * & d & & \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \ddots & \bigcirc \\ * & * & * & & * & * & * & \bigcirc & & d \end{array}\right].$$

Continuing in this fashion we obtain $\bar{A}_n = [H|0]$. Note that a column de_j, once destroyed in this process, is no longer needed for mod d operations.

Coupled with determination of d, the algorithm just described provides means for computing H from A in polynomial time. Edmonds [1967] has given a polynomial-time Gaussian elimination procedure which can be used to compute d. It is important to notice that each intermediate matrix entry for Edmonds' algorithm is the determinant of a submatrix of the original matrix. Hence the $n(\log_2 n + \log_2 a)$ bound on the number of bits required to represent any matrix entry encountered in the course of the computation (as discussed in §1) also applies to Edmonds' algorithm.

Further procedures for computing H using mod d arithmetic may be based on Corollaries 2.3 and 2.8. In particular, (2.8) provides means for determining the correct diagonal entries of H from a matrix G produced by applying to A any of several standard algorithms modified to perform computation mod d. These results apply to a broad class of algorithms and we investigate computational performance in conjunction with three standard Hermite normal form algorithms (ROSSER, BRADLEY, KANBACH) in the following section.

In particular, suppose G is a lower triangular matrix obtained from A by applying a sequence of column operations of types (1.1.a–c), reducing entries mod d in the process. By (2.8), $h_{ii} = \text{g.c.d.}(d_i, g_{ii})$ and we may determine integers p, q so that $pg_{ii} + qd_i = h_{ii}$. We then multiply the ith column of G by p and reduce its elements mod d_i. Since $pg_{ii} \equiv h_{ii} \pmod{d_i}$, the ith element of the resulting column vector is h_{ii}. (Note that when $pg_{ii} \pmod{d_i} \equiv 0$, we must have $h_{ii} = d_i$.) Furthermore, one easily verifies that this vector is in $L(A)$. Thus after applying the above procedure to each consecutive column of G, we may assume that G satisfies the hypothesis of Corollary 2.3. Hence the obvious unimodular column operations can be applied to G to produce a matrix whose columns satisfy the conditions of (1.2) and, by Proposition 2.1, the matrix which results must be H. We remark that this last stage of the computation, i.e., reduction of subdiagonal elements so that the requirements of (1.2) are met, is also to be done using mod d arithmetic.

In fact the above discussion suggests that one could take d_i as the modulus for arithmetic computation in row i. Of course, only $d_1 \equiv d$ is known at the start of the computation, but an elementary procedure such as that first described in this section can be easily modified to perform mod d_i arithmetic once d_i is known. With this modification, the upper bound on the number of bits required to represent matrix entries may decrease significantly over the course of the computation.

Consider again the augmented matrix $\bar{A} \equiv [A|D]$. Using the columns of D to perform mod d_1 ($\equiv d$) operations, we transform $\bar{A}$ into the following form using only unimodular column operations:

$$\bar{F}_1 \equiv \left[\begin{array}{cccc|cccc} f_{11} & 0 & \cdots & 0 & d_1 & \bigcirc & & \\ * & * & \cdots & * & & d_1 & & \\ \vdots & \vdots & & \vdots & \bigcirc & & \ddots & \\ * & * & \cdots & * & & & & d_1 \end{array}\right] \equiv [F_1|D_1].$$

Proposition 2.7 now implies (with $i = 1$) that g.c.d.$(d_1, f_{11}) = h_{11}$. Furthermore, for $i \geqslant 2$ we have g.c.d.$(d_1, \gamma_i(F_1))$ = g.c.d.$(d_1, \gamma_i(A)) = h_{11} \dots h_{ii}$. Thus, since f_{11} divides the determinant of any $i \times i$ submatrix appearing in rows $1, \dots, i$ of F_1, we may write schematically that g.c.d.$(d_1, \gamma_i(F_1))$ = g.c.d.$(d_1, f_{11}p_1, \dots, f_{11}p_{k(i)}) = h_{11} \dots h_{ii}$. We now perform a unimodular column operation on columns 1 and $n+1$ of $\bar{F}_1$ to replace f_{11} by h_{11} = g.c.d.(d_1, f_{11}), obtaining

$$\bar{G}_1 \equiv \left[\begin{array}{cccc|cccc} h_{11} & 0 & \cdots & 0 & 0 & & & \\ * & * & \cdots & * & * & d_1 & & \bigcirc \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \\ * & * & \cdots & * & * & \bigcirc & & d_1 \end{array}\right] \equiv [G_1 \mid E_1].$$

Since h_{11} = g.c.d.(d_1, f_{11}) we have that

$$\text{g.c.d.}\big(d_1, f_{11}p_1, \dots, f_{11}p_{k(i)}\big) = \text{g.c.d.}\big(h_{11}d_2, h_{11}p_1, \dots, h_{11}p_{k(i)}\big), \qquad 1 \leqslant i \leqslant n,$$

and hence

$$\text{g.c.d.}\big(d_1, \gamma_i(A)\big) = \text{g.c.d.}\big(d_1, \gamma_i(G_1)\big) = h_{11} \dots h_{ii}, \qquad 1 \leqslant i \leqslant n.$$

Since g.c.d.$(d_1, \gamma_i(G_1)) = h_{11}$ g.c.d.$(d_2, p_1, \dots, p_{k(i)})$, $2 \leqslant i \leqslant n$, we now focus attention on the matrix $[A_2 \mid D_2]$, where A_2 is the $(n-1) \times (n-1)$ submatrix of G_1 obtained by deleting the first row and column and D_2 is the $(n-1) \times (n-1)$ matrix $d_2 I$. Continuing in this fashion we ultimately obtain a matrix of the form

$$\left[\begin{array}{ccc|ccc} h_{11} & & \bigcirc & 0 & & \bigcirc \\ & \ddots & & & \ddots & \\ \bigstar & & h_{nn} & \bigstar & & 0 \end{array}\right] \equiv [G \mid E],$$

where G satisfies the hypothesis of (2.3). Thus H results from G by applying unimodular column operations to reduce off-diagonal entries modulo diagonal entries. During these final column operations, the computation in row i can be performed modulo d_i, $1 \leqslant i \leqslant n$. Note that it is unnecessary to remove subdiagonal nonzero entries of E; thus fewer column operations are required here than for the algorithm described at the beginning of this section. Computational performance with this procedure (called DECMOD) is also discussed in the following section.

4. Computational experience.

In this section we report computational experience with the Hermite normal form algorithms described earlier. The material here consists of three basic parts. First, traditional procedures similar to those proposed by Rosser [1952], Bradley [1971] and Kannan and Bachem [1979], termed ROSSER, BRADLEY and KANBACH, are examined computationally. These procedures are then altered to include determinant information; i.e., arithmetic is performed modulo d, with subsequent "reconstruction" of H. The final computational results are for the algorithm described at the end of the previous section, using a decreasing modulus for arithmetic operations and requiring no reconstruction. This latter algorithm is termed DECMOD.

The basic ROSSER algorithm uses only the elementary operations (1.1.a–c). The computation proceeds in row order, working within each row to the right of its diagonal entry. Successive column operations are used to reduce the entry of largest magnitude modulo the second largest. This process continues in a given row until all

TABLE 4.0

Problem Characteristics.

Problem	Dimension	det(A)
1,2	8×8	480, 14976
3,4	10×10	36288, 1105920
5,6	16×16	58982400, 1719926784
7,8	20×20	$O(2^{48})$
9,10	32×32	$O(2^{64})$

TABLE 4.1

Solution Statistics for Three Basic Algorithms Implemented Without mod d *Arithmetic.*

Algorithm	Problem	Solution Time	Maxword	Column Operations
ROSSER	1	0.2496	1	137
	2	0.2263	1	117
	3	0.5857	2	268
	4	0.4526	2	199
BRADLEY	1	1.5308	28	83
	2	1.7539	31	84
	3	—	> 200	—
	4	—	> 200	—
KANBACH	1	0.2828	17	83
	2	0.2695	13	84
	3	1.2945	109	134
	4	0.5026	25	134

entries to the right of the diagonal have become 0. This procedure is well known to exhibit slow coefficient growth, although no polynomial bound on the size of the coefficients occurring during the computation is known.

The BRADLEY algorithm implemented here is a slight variant of the elimination scheme developed in Bradley [1971]. Again the elimination proceeds in row order, zeroing entries to the right of the diagonal in row i by using $n - i$ composite (g.c.d.) unimodular operations as described in §1. We attempt to distribute these operations evenly over the columns involved, and thereby potentially constrain coefficient growth, by applying column operations in row i to the pairs $(i, i + 1), (i + 1, i + 2), \ldots, (n - 1, n)$, successively removing the first entry of each pair. An interchange of columns i and n then completes the elimination step for row i.

The implementation KANBACH of the procedure of Kannan and Bachem [1979] is as described in Section 1. Here, at the start of iteration i, the $(i - 1)$st principal submatrix is in Hermite normal form, and we use the composite (g.c.d.) operation on column pairs $(1, i), (2, i), \ldots, (i - 1, i)$ to reduce the first $i - 1$ entries in column i to zero. Then entries to the left of the diagonal are reduced to place the ith principal submatrix in Hermite normal form. And so on.

These three general procedures have been applied to a set of generated test problems with known solutions. All original matrix entries are of magnitude smaller than 2^{16}. Problem characteristics are summarized in Table 4.0. The results presented in Table 4.1 indicate solution statistics for the three routines discussed above on the smaller problems; solution times are in CPU seconds for an IBM3081 computer using the FORTVS compiler at optimization level 3. All computations were performed using exact linked-integer arithmetic (see Domich 1983). The parameter MAXWORD indicates for each problem instance the number of 16-bit integer words necessary to

TABLE 4.2

Solution Statistics for Three Basic Algorithms Implemented with Computation mod *d*

	ROSSER		BRADLEY		KANBACH	
Problem	Time	Col. Oper.	Time	Col. Oper.	Time	Col. Oper.
1	0.1963	122	0.1331	82	0.1397	81
2	0.1664	100	0.1664	83	0.1564	83
3	0.4126	247	0.3161	133	0.2895	133
4	0.2829	166	0.5858	134	0.5325	133
5	1.9735	704	2.5159	360	2.5592	359
6	1.7805	655	2.5392	358	2.4428	359
7	4.4395	1302	5.6675	569	5.4013	570
8	3.9803	1189	4.8622	568	4.7723	567
9	27.1564	4884	29.4128	1487	28.6474	1487
10	28.9702	5219	31.2931	1488	31.7425	1488

represent the single matrix entry of largest magnitude encountered during the computation. For the larger problems (numbered 5–10 in Table 4.0), neither BRADLEY nor KANBACH were able to determine the Hermite normal form, even though up to 200 16-bit integer words were allowed for each matrix entry—implying that some entry magnitudes exceeded 2^{3200} during the computation. The ROSSER algorithm, although capable of solving problems up to size 20×20, exceeded the maximum time limit of 60 seconds for the largest problems (32×32).

The second phase of computational testing considered the three basic procedures described above, each implemented using mod d arithmetic. That is, whenever any matrix entry exceeded the value d, it was reduced mod d. The reconstruction of H from the resulting lower triangular matrix was as described in the previous section. Table 4.2 contains these computational results. In this setting note that all procedures solved all of the test problems more quickly and with a decreased storage requirement.

The final procedure implemented uses the decreasing modulus as described at the end of §3. Such a procedure is apparently computationally advantageous for the largest problems considered here. These results are summarized in Table 4.3.

In conclusion we note that for all problems solved here a major portion of the computational effort involves the manipulation of large integers using extended precision software. In Domich [1985] a class of Hermite normal form algorithms is developed using congruence techniques to enable operation within standard working precision. These algorithms and corresponding computational experience will be the subject of a later paper.

TABLE 4.3

Solution Statistics for Algorithm DECMOD

Problem	Time	Col. Oper.
1	0.1297	81
2	0.1365	82
3	0.3028	133
4	0.3994	132
5	2.1432	358
6	1.9436	359
7	4.4728	568
8	4.2931	567
9	23.1196	1485
10	26.0483	1488

References

Birkhoff, G. and MacLane, S. (1967). *A Survey of Modern Algebra*. MacMillan, London.

Bradley, G. H., (1971). Algorithms for Hermite and Smith Normal Form Matrices and Linear Diophantine Equations. *Math. Comp.* **25** 897–907.

Chou, J. T. (1979). Algorithms for the Solution of Systems of Linear Diophantine Equations. Ph.D. Dissertation, Department of Computer Science, University of Wisconsin, Madison.

Domich, P. D. (1983). Three New Polynomially-Time Bounded Hermite Normal Form Algorithms. M.S. Thesis, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, N.Y.

______. (1985). Residual Methods for Computing Hermite and Smith Normal Forms. Ph.D. Dissertation, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, N.Y.

Edmonds, J. (1967). Systems of Distinct Representatives and Linear Algebra. *J. Res. Nat. Bur. Standards* **71B** 241–245.

Frumkin, M.A. (1977). Polynomial Time Algorithms in the Theory of Linear Diophantine Equations. in M. Karpinski. (Ed.), *Fundamentals of Computation Theory*. Springer, Berlin and New York, *Lecture Notes in Computer Sci.* **56** 386–392.

Havas, G. and Sterling, L. (1979). Integer Matrices and Abelian Groups. *Symbolic and Algebraic Computation. Lecture Notes in Computer Sci.*, EUROSAM'79, An International Symposium on Symbolic and Algebraic Manipulation, Marseille, France.

Hermite, C. (1851). Sur l'Introduction des Variables Continues dans la Théorie des Nombres, *J. Reine Angew. Math.* **41** 191–216.

Kannan, R. and Bachem, A. (1979). Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix *SIAM J. Comput.* **9** 499–507.

______ and Lenstra, H. K. (1982). Private Communication.

Lamé, G. (1844). Note sur la Limite du Nombre des Divisions dans la Recherche du Plus Grand Commun Diviseur Entre Deux Nombres Entiers, *Co. Re. Acad. Sci. Paris* **19** 867–870.

McClellan, M. T. (1973). The Exact Solution of Systems of Linear Equations with Polynomial Coefficients. *J. Assoc. Comput. Machin.* **20** 563–588.

Rosser, J. B. (1952). A Method of Computing Exact Inverse of Matrices with Integer Coefficients, *J. Res. Nat. Bur. Standards* **49** 349–358.

Sahni, S. (1974). Computationally Related Problems. *SIAM. J. Comput.* **3** 262–279.

Schrijver, A. (1985). *Theory of Linear and Integer Programming*. John Wiley, Chichester, England.

Sims, C. C. (1974). The Influence of Computers on Algebra. *Proc. Sympos. Appl. Math.* **20** 13–30.

Smith, D. A. (1966). A Basis Algorithm for Finitely Generated Abelian Groups. *Math. Algorithms* **1** 13–26.

DOMICH: CENTER FOR APPLIED MATHEMATICS, NATIONAL BUREAU OF STANDARDS, BOULDER, COLORADO 80303

KANNAN: CARNEGIE-MELLON UNIVERSITY, PITTSBURGH, PENNSYLVANIA 15213

TROTTER: SCHOOL OF OR AND IE, CORNELL UNIVERSITY, ITHACA, NEW YORK 14853