Residual Hermite Normal Form Computations

PAUL D. DOMICH
National Institute of Standards and Technology

This paper extends the class of Hermite normal form solution procedures that use modulo determinant arithmetic. Given any relatively prime factorization of the determinant value, integral congruence relations are used to compute the Hermite normal form. A polynomial-time complexity bound that is a function of the length of the input string exists for this class of procedures. Computational results for this new approach are given.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations on matrices*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*determinants*; I.1.2 [**Algebraic Manipulation**]: Algorithms

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Hermite normal form, integral congruence techniques, matrices with integer entries, modulo determinant arithmetic

1. INTRODUCTION

Consider an $n \times n$ matrix $A = [a_{ij}]$ with integer-valued entries. The problem addressed here can be stated as that of finding an "equivalent" lower triangular matrix with integer-valued entries that determines all integer points $b \in Z^n$ such that $Ax = b$ for $x \in Z^n$. One well-known "equivalent" lower triangular form related to A by a unimodular matrix K is called the *Hermite normal form* of A, denoted as $H = [h_{ij}]$; a matrix K with integer entries is unimodular if $|\det(K)| = 1$. For this work, A is assumed square and $d = |\det(A)| > 0$. The generalization of these results for an arbitrary matrix with integer-valued entries is straightforward. The reader may wish to refer to an earlier paper by Domich et al. [7] and perhaps [5] and [6] for a more complete discussion of the preliminary algebraic development.

THEOREM 1.1 (Hermite). *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

This research was supported in part by National Science Foundation grant ECS8113534 and was conducted at CORE, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, and the National Institute of Standards and Technology, Boulder, Colorado.
Author's current address: National Institute of Standards and Technology, Applied and Computational Mathematics Division (719), Boulder, CO 80303-3328.

normal form *of A, whose entries satisfy*

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

PROOF. A constructive proof is outlined later in this section, or see [11]. □

THEOREM 1.2. *There is a unique matrix H satisfying Theorem* 1.1.

PROOF. For an easy proof of this last result see Domich et al. [7]. □

The following elementary column operations can be used to determine H:

interchange two columns; (1.1)

multiply a column by -1; (1.2)

add an integral multiple of one column to another. (1.3)

These three elementary column operations, or composites of operations (1.1)–(1.3), applied to the A matrix correspond to a unimodular transformation of A. Two types of unimodular transformations are used.

$$\begin{matrix} \textit{Type 1} \\ \begin{bmatrix} i & 0 & 1 \\ j & 1 & -s \end{bmatrix} \end{matrix} \qquad \begin{matrix} \textit{Type 2} \\ \begin{bmatrix} p & -\dfrac{a_{ij}}{r} \\ q & \dfrac{a_{ii}}{r} \end{bmatrix} \end{matrix} \tag{1.4}$$

In the most simplistic procedure, iteration i begins with all superdiagonal entries in the first $(i-1)$ rows of A equal to zero. Both transformations operate on columns i and j, $i < j$, of the matrix A. The unimodular transformation of Type 1 performs operation (1.3) where traditionally the factor s in (1.4) is set equal to the largest integer no greater than a_{ii}/a_{ij}, that is, $s = \lfloor a_{ii}/a_{ij} \rfloor$. This choice for s results in reducing a_{ii} modulo a_{ij}, denoted as $a_{ii} \bmod (a_{ij})$. Other choices for s using the "golden ratio" try to limit the magnitude of the factors s for an entire row iteration [6].

The Type 2 "composite" transformation is a product of Type 1 transformations. This transformation replaces a_{ii} with $gcd(a_{ii}, a_{ij})$ and zeros a_{ij}. Here integers p, q, and r such that $pa_{ii} + qa_{ij} = gcd(a_{ii}, a_{ij}) = r$ are found by the Euclidean algorithm (see, for example, Knuth [14, p. 293–316]).

One well-known procedure outlined by Rosser [19] uses only Type 1 transformations. Applying these transformations in row order and from the diagonal to the right within each row, A is transformed into a lower triangular form in a finite number of steps. Additional transformations of the first type applied in row order reduce subdiagonal entries modulo the corresponding diagonal element. Since the product of unimodular matrices remains unimodular, this procedure suggests a constructive proof of Theorem 1.1.

A second procedure by Bradley [3] applies both Type 1 and Type 2 transformations using an elimination order similar to that of Rosser. Other proposed solution procedures include that of Blankenship [2], Sims [20], and Smith [21].

More recently, a new order of computation using Type 2 transformations was suggested by Kannan and Bachem [13] for which they prove a polynomial bound both for the number of iterations and the required storage to solve for H as a function of the input data string. However, all of these procedures empirically exhibit a rapid rate of coefficient growth. This "entry explosion" [9, 10, 17] has in some cases prevented the solution of test problems of dimension 10 with initial entries bounded by 2^{16} allowing 3200 bits for representing each (integer) entry (see [5] and also [4]).

The results presented in this paper extend those of [7] in which polynomial-time algorithms were developed for solving for the Hermite normal form using modulo d arithmetic. A general algebraic development for the modulo d approaches along with the fundamental results from the earlier work are reproduced in Section 2. A new procedure is developed in Section 3 that performs modulo d arithmetic using a general integral congruence relation. Computational experience with this new type of Hermite normal form procedure and two of the earlier modulo d procedures are presented in Section 4.

2. ALGEBRAIC PRELIMINARIES

Several of the important results for the general modulo d Hermite normal form procedures are summarized below. The interested reader should refer to [7] or [5] and [6] for a full algebraic development of these results.

THEOREM 2.1. *Suppose A is a nonsingular $n \times n$ matrix with integer entries and $I * d$ is an identity matrix scaled by $d = |\det(A)|$. Define matrices*

$$A' = \begin{bmatrix} A & I * d \\ 0 & I \end{bmatrix} \quad \text{and} \quad H' = \begin{bmatrix} H & O \\ 0 & I \end{bmatrix}$$

Then H' is the Hermite normal form of A', and conversely, H' determines H, the Hermite normal form of A.

PROOF. Since A^{-1} exists and is equal to A^+/d, where A^+ is the integer adjoint matrix of A, then $I * d = AA^+$. Thus $I * d$ is an integral combination of the columns of A. The result follows from the uniqueness of H (Theorem 1.2). □

With this theorem the validity of a general modulo d Hermite normal form procedure is apparent. Standard transformations of Type 1 and 2 are first applied to the A portion of A' as seen in Section 1. Entries in A can then be altered by integral multiples of d using operation 1.3 and columns of $I * d$. Thus modulo d arithmetic on A is reduced to a simple column operation. Define $G = [g_{ij}]$ as the lower triangular form found from the matrix A applying the operations specified above.

The Hermite normal form of A is found after removing $I * d$ in A' using columns of G in a similar fashion. Note that removing $I * d$ in column order permits modulo d arithmetic as before on the "fill" entries introduced into $I * d$ using the untouched columns of $I * d$. Further, rows $n + 1$ to $2n$ have no affect on H and can be ignored. This result provides the basis for a modulo d approach. An analogous result exists for a second matrix D.

COROLLARY 2.2. *Let* $d_1 = d$, *and define* $d_{i+1} = d_i/h_{ii}$ *for* $i = 1, 2, \ldots, n-1$, *where* h_{ii} *is the* i*th diagonal element of* H, *the Hermite normal form of* A. *Then the diagonal matrix* $D = diag(d_1, d_2, \ldots, d_n) = AN$ *for some* $n \times n$ *integer matrix* N.

PROOF. See [7] or [5] and [6]. □

The matrix D is used in a similar fashion as $I * d$. Suppose D replaces $I * d$ in A', and let G be a lower triangular matrix computed from the A matrix after applying the standard unimodular transformations and modulo arithmetic using columns of D, that is, $a_{ij} \equiv a_{ij} \bmod (d_i)$.

PROPOSITION 2.3. *For* $1 \le i \le n$, $h_{ii} = gcd(g_{ii}, d_i)$, *where* d_i *is as defined in Corollary* 2.2.

PROOF. A constructive proof of Proposition 2.3 demonstrates that the resulting diagonal elements of G remain unaltered after eliminating the diagonal entries in D. See [6] or [7]. □

By the uniqueness of H, it can be shown that eliminating the diagonal entries from D using the columns of G is sufficient to determine H. This last result significantly reduces the computational effort required to find the Hermite normal form of the A matrix. This type of procedure is called a *decreasing modulus approach.*

An extension to these basic developments requires two classical results, the Chinese Remainder Theorem and the Unique Prime Factorization Theorem. Both are characterizations of unique representation properties for the integers. Using these results, an algorithm is developed that performs the modulo arithmetic operation implicitly. This is done using congruence relations which further reduce the size of the largest operand encountered by the procedure. This modification extends the class of problems that can be solved using standard working precision while simplifying the modulo operation.

THEOREM 2.4 (Chinese Remainder Theorem). *Suppose* $0 < p_1 < p_2 < p_3 \cdots < p_k$ *are pairwise relatively prime integers, and let* $p = \prod_{i=1}^{k} p_i$. *For some integer* u, $0 \le u < p$, *let* $u_i \equiv u \bmod (p_i)$, $\forall i$. *Then* u *is uniquely determined by* $(u_1, u_2, \ldots, u_k)$, *where* $0 \le u_i < p_i$, $\forall i$.

PROOF. See, for example, [1]. □

Corollary 2.5 extends Theorem 2.4 to the field of all integers.

COROLLARY 2.5. *Suppose* $u \ge p$ *or* $u < 0$. *Then* $(w_1, w_2, \ldots, w_k)$, *where* $w_i \equiv u \bmod (p_i)$ *determines a unique integer* w *such that* $0 \le w < p$ *and* $w \equiv u \bmod (p)$.

PROOF. Since p is divisible by each p_i, writing u as $w + w'p$ for integers w and w', $0 \le w < p$, then for all i

$$\begin{aligned} u \bmod(p_i) &\equiv (w + w'p) \bmod(p_i) \\ &= w \bmod(p_i) \\ &= w_i. \end{aligned}$$

□

THEOREM 2.6 (Unique Prime Factorization Theorem). *Any integer $\alpha > 1$ can be uniquely factored in one way as $p_1^{\sigma_1} p_2^{\sigma_2} \cdots p_k^{\sigma_k}$ where the σ_i's are positive integers and $p_1 < p_2 < \cdots < p_k$ are positive prime numbers.*

PROOF. See, for example, [12]. □

Hence, any relatively prime factorization of an integer $p > 0$ as in Theorem 2.6 corresponds to a particular partition of the factors above. Further, any such relatively prime factorization of p may be used to uniquely represent a second integer u, $0 \le u < p$, using the residual vector as in Theorem 2.4.

Thus, for the Hermite normal form development, let $p = d = |\det(A)| > 0$. Then any entry a_{ij} in matrix A has a residual representation using the factors defined above that uniquely determines an integer w_{ij}, $0 \le w_{ij} < d$. Further, w_{ij} is equal to $a_{ij} \bmod(d)$. Often the residual form of an integer is used in conjunction with its mixed radix representation (see, e.g., Knuth [14, p. 175]).

PROPOSITION 2.7. *Let $0 < p_1 < p_2 < p_3 \cdots < p_k$ be pairwise relatively prime integers, and let $p = \prod_{i=1}^{k} p_i$. Then an integer u, $0 \le u < p$, is uniquely determined by $(v_i, v_2, \ldots, v_k)$ for $0 \le v_i < p_i$ where*

$$u = v_1 + v_2 p_1 + v_3 p_1 p_2 + \cdots + v_k p_1 p_2 \cdots p_{k-1}.$$

PROOF. See, for example, [16]. □

The homomorphic transformation from residual to radix representation is as follows:

$$\begin{aligned}
v_1 &= u_1 \bmod(p_1), \\
v_2 &= (u_2 - v_1) c_{12} \bmod(p_2), \\
v_3 &= ((u_3 - v_1) c_{13} - v_2) c_{23} \bmod(p_3), \\
&\vdots \\
v_k &= (((u_k - v_1) c_{1k} - v_2) c_{2k} \cdots - v_{k-1}) c_{k-1,k} \bmod(p_k),
\end{aligned}$$

where $(u_1, \ldots, u_k)$ is the vector of residual values associated with u, and integers c_{ij} are selected such that $p_i c_{ij} \equiv 1 \bmod(p_j)$, where $p_i < p_j$. The scalars c_{ij} are easily found by the Euclidean algorithm.

3. MODULO *D* CONGRUENCE PROCEDURES

The procedures developed in this section use a pairwise relatively prime factorization of $d = |\det(A)|$ as the moduli for an integral congruence relation. These factors further reduce the size of the operands in the procedure, thus permitting the solution of larger problems in standard working precision than any of those procedures previously described. The implementation of these approaches is more efficient since the operation of reducing an entry modulo d is performed implicitly.

Given any relatively prime factorization of d, a modulo d Hermite normal form procedure is defined that uses standard congruence relations to represent matrix entries modulo d. The size of the operands encountered during computation is then bounded by the largest factor of d. The validity of this type of procedure follows from Theorem 2.1 and the Chinese Remainder Theorem 2.4.

These procedures are robust in that any relatively prime factorization of d can be used. When no nontrivial factorization of d is known, the procedure is identical to a general modulo d Hermite normal form approach seen in [7], and a polynomial-time bound for this procedure follows directly from those general modulo d results. Otherwise, a polynomial-time bound exists as a function of these factors and the problem description.

3.1 Factoring the Determinant Value d

Suppose an $n \times n$ nonsingular matrix A with integer entries is given. As a result of the determinant value computation, say by the polynomial-time bounded Gaussian elimination procedure of [8], a partial factorization of d may be available. This partial factorization can be easily adjusted into a relatively prime factorization of d using the Euclidean algorithm. These new factors are used as moduli in an integral congruence relation as defined in Theorem 2.4.

Other routines to factor d directly can be found in [14, p. 339–359] or [18], for example. It remains a well-known open question whether a polynomial-time algorithm exists for factoring an integer, as well as deciding in polynomial time whether an integer is composite (see [15]). This work does not address the problem of factoring d; the trivial factors 1 and d are used when no nontrivial factorization of d is known.

3.2 Development of the Modulo D Congruence Procedures

We assume throughout the remainder of the paper that positive integers p_1, $p_2, \ldots, p_k$ are known such that $d = \prod_{i=1}^{k} p_i$, where $gcd(p_i, p_j) = 1$ for all $i \neq j$. Using an integral congruence relation as in Theorem 2.4, each entry (i, j) in the A matrix is represented as

$$(a_{ij1}, a_{ij2}, \ldots, a_{ijk})$$

where $a_{ijt} \equiv a_{ij} \bmod(p_t)$. Thus, if column l of A is scaled by s and added to column j, then the tth residual term of the (i, j) entry is set equal to

$$(a_{ijt} + sa_{ilt}) \bmod(p_t).$$

This can be viewed in matrix format as k separate matrices, $A^{(p_t)} = [a_{ijt}]$, where $A^{(p_t)}$ is maintained modulo p_t, for $t = 1, \ldots, k$. Applying unimodular transformations K_ℓ found during row iteration ℓ, the updated residual matrices at the start of row iteration i are

$$A^{(p_t)} \equiv AK_1K_2 \cdots K_{i-1} \bmod(p_t), \qquad \text{for} \quad 1 \leq t \leq k.$$

With these k residual matrices, row i can be computed explicitly using the mixed radix representation for each entry. Hence, row entries are known up to integral multiples of d. The unimodular transformation matrix K_i is then determined from row i and applied to each of the residual matrices, $A'^{(p_t)} \equiv A^{(p_t)}K_i \bmod(p_t)$.

THEOREM 3.1. *Matrices $A'^{(p_1)}$, $A'^{(p_2)}$, ... $A'^{(p_k)}$ found at the end of row iteration i as defined above determine the unique $n \times n$ matrix A' with entries $a'_{st} \in [0, d)$ and with $A' \equiv AK_1K_2 \cdots K_i mod(d)$.*

PROOF. For $i = 1$, $A'^{(p_t)} \equiv A \bmod(p_t) K_1 \bmod(p_t) = AK_1 \bmod(p_t)$, for $1 \le t \le k$, and by Corollary 2.5 the theorem holds. For row iteration $i - 1$, assume residual matrices $A^{(p_1)}, A^{(p_2)}, \ldots A^{(p_k)}$ that satisfy the conditions of the theorem are known. Hence, for any unimodular matrix K_i found during row iteration i

$$A'^{(p_t)} \equiv A^{(p_t)} K_i \bmod(p_t) = (AK_1K_2 \cdots K_{i-1}) \bmod(p_t) K_i \bmod(p_t)$$
$$= AK_1K_2 \cdots K_i \bmod(p_t),$$

for $t = 1, \ldots, k$, and once again by Corollary 2.5 the theorem holds. □

An alternative solution procedure using congruence techniques updates a single row of A in residual form for each row iteration. For this type of *product form approach*, the unimodular transformations are saved for later row iterations. The updated form of the ith row at the start of the ith iteration is found by evaluating in residual form the matrix vector product

$$A_i K_1 K_2 \cdots K_{i-1},$$

where A_i represents the ith row of the original A matrix and K_ℓ is the unimodular matrix determined in row iteration ℓ. This matrix-vector product is computed modulo p_t, for $t = 1, \ldots, k$, to find the residual form for row i. From the residual vectors, the radix form of the row is determined, and unimodular transformations are found and applied to row i as before.

THEOREM 3.2. *The matrix* $K^{(d)} \equiv K_1K_2 \cdots K_{i-1} mod(d)$ *is sufficient to determine the updated form of the ith row.*

PROOF. Suppose $d = |\det(A)|$, and let $K_1K_2 \cdots K_{i-1} = K^{(d)} + Sd$ for some $n \times n$ matrix S with integer entries (j, ℓ) such that $0 \le K^{(d)}_{j\ell} < d$, $\forall j, \ell$. Then the updated form of row i is of the form

$$A_i K_1 K_2 \cdots K_{i-1} = A_i(K^{(d)} + Sd) \equiv A_i K^{(d)} \bmod(d).$$

Since this row is known to within integral multiples of d, the result follows. □

With the transformation matrix $K^{(d)}$ maintained modulo d, the Hermite normal form can be determined while bounding the storage requirements for the entire procedure as a polynomial function of the input. Note that $K^{(d)}$ is not necessarily unimodular as specified in Theorem 1.1; the matrix K with integer entries such that $AK = H$ and $|\det(K)| = 1$ can be determined from this linear relation.

In practice, the matrices K_ℓ in the product form of K have not been found to have excessively large coefficients, and explicit representation of each K_ℓ is possible, as seen in Section 4. It is also possible to maintain $K^{(d)}$ in residual form, that is, to define matrices $K^{(p_1)}, K^{(p_2)}, \ldots K^{(p_k)}$ at the start of iteration i such that $K^{(p_t)} = K_1K_2 \cdots K_{i-1} \bmod(p_t)$, $\forall t$.

The procedure begins with each $K^{(p_t)}$ initialized to an identity matrix. As iterations progress, each $K^{(p_t)}$ is modified by the unimodular operations determined from the current row of A. The updated residual form of the (i, j) entry

of A at the start of the ith iteration is, therefore,

$$(A_i K_{\cdot j}^{(p_1)} \bmod(p_1),\ A_i K_{\cdot j}^{(p_2)} \bmod(p_2),\ \ldots,\ A_i K_{\cdot j}^{(p_k)} \bmod(p_k)),$$

where $K_{\cdot j}^{(p_t)}$ is the jth column of $K^{(p_t)}$. The unimodular operations applied to the ith row of A are applied to each of the k residual matrices $K^{(p_t)}$ for use in the next iteration and for determining the final K' defined above.

4. COMPUTATIONAL RESULTS

In this section four Hermite normal form procedures are examined empirically. Two procedures (KANBAC and ROSSER) have been previously examined in an earlier work [7] and include modulo d arithmetic and a reconstruction phase. A single congruence relation Hermite normal form procedure (RESROS) was developed as a variant on ROSSER which uses modulo d and an implicit reconstruction step (see Section 3). Owing to the slower growth of coefficients for the method of Rosser, RESROS was then altered so as to not require determinant value information. For this procedure, the size of the largest matrix entry is monitored at each iteration and an appropriately large set of moduli for the congruence relation is determined. This guarantees that no entry exceeds in magnitude the product of the prime factors and that the correct Hermite normal form can be reconstructed from the residual representation of the matrix entries.

All of the traditional Hermite normal form methods are easily adapted to include congruence relation representation of the matrix entries and modulo d arithmetic. These other methods, though, for example, KANBAC, have been found to experience faster coefficient growth. This may indicate that using congruence relations without determinant value information would be inappropriate since the number and size of prime factors could become prohibitively large. Hence, only the procedure of Rosser is examined in this setting. The procedures reported in this section are described in more detail below.

At the start of iteration i for the procedure of Kannan and Bachem [13] implemented in KANBAC, the $(i-1)$st principal submatrix of A is in lower triangular form. To zero the first $i-1$ entries in column i, unimodular transformations of Type 2 are applied sequentially to column pairs $(1, i)$, $(2, i)$, $\ldots$ $(i-1, i)$. The procedure then iterates for the $(i+1)$st principal submatrix. For KANBAC implementation, the subsequent reconstruction step and the reduction of subdiagonal entries is postponed until the entire matrix is in lower triangular form.

The ROSSER procedure uses operations (1.1)–(1.3) in a row by row elimination scheme. In row iteration i, the entry with the largest magnitude on and to the right of the diagonal is reduced modulo the second largest entry using operation (1.3). Here, entries to the left of the diagonal are reduced iteratively so as to lower the size of the multipliers. That is, whenever an entry in the current row, say a_{ij} where $j < i$, has a magnitude that exceeds the largest entry a_{ik}, for $k \geq i$, operation (1.3) is performed reducing $a_{ij} \bmod(a_{ik})$.

Modifying this last procedure to include a residual representation of the matrix entries using an integral congruence relation (RESROS) requires a nontrivial, relatively prime, factorization of d. Here it is assumed that d has been determined

using a stable procedure such as lower/upper triangular unit (LU) decomposition [8], and the factors of d are adjusted to be relatively prime using the Euclidean algorithm. Hence, relatively prime integers p_i, $0 < p_1 < p_2 < \cdots < p_k$ exist such that $\prod_{i=1}^{k} p_1 = d$ and $gcd(p_i, p_j) = 1$ for $i \neq j$. The residual form of a_{ij} is, therefore,

$$(a_{ij} \bmod(p_1),\ a_{ij} \bmod(p_2),\ \ldots,\ a_{ij} \bmod(p_k)).$$

A product form procedure was implemented for the RESROS approach (see Section 3). At the start of iteration i, the residual form of row i is $A_i K_1 K_2 \cdots K_{i-1} \bmod(p_t)$ for $t = 1, \ldots, k$, and K_i is initialized to an identity matrix. The radix form for each entry in the row is determined, and operations (1.1)–(1.3) are applied to the radix form and the corresponding columns of K_i. At the end of the iteration, the final form of row i and K_i are stored and the procedure iterates.

If K_i is maintained exactly, coefficients in K_i may become excessively large, and a product form of K_i can be used. In this situation, a second K_i matrix is initialized and operations continue with this new matrix as before. Additional unimodular matrices found in row iteration i are applied in order of appearance in computing the updated form of later rows of A. For notational convenience, let K_i denote the product of all transformation matrices found in iteration i. As seen in Section 3, it is possible to implement the product form procedure maintaining the residual form of $K^{(d)}$, thus bounding the storage requirements for the procedure.

All the traditional elimination schemes mentioned in Section 1 can be used on the residual form of A with the single requirement that all operations are applied to the residual vector of each entry a_{ij}. The result is a lower triangular form of A found using modulo arithmetic and operations (1.1)–(1.3). Note that the scaling factor in (1.4), $s = \lfloor a_{ij}/a_{ij} \rfloor$, is computed either explicitly using the mixed radix form of each entry or by a real-valued approximation:

$$\left\lfloor \frac{b_{ij}}{b_{ii}} \right\rfloor$$

where $b_{i\cdot} p_1 p_2 \cdots p_{k-1} \approx a_{i\cdot}$, and $b_{i\cdot} \in (0, p_k)$. If the real-valued approximation is used and s is incorrectly specified, the resulting entry may become negative. The corresponding column is then scaled by -1, the residual values are adjusted appropriately, and operations continue.

The transformations needed to locate the diagonal entries of H can be performed either explicitly or in residual form with only slight modifications. Recall that the reconstruction requires the elimination of $I * d$ using columns of A and unimodular transformations. To represent the entries of magnitude equal to d, the radix vector

$$(0, 0, \ldots, 0, p)$$

is used. Once this reconstruction is completed, H is known up to the final reduction of subdiagonal entries modulo the corresponding diagonal elements.

The RESROS procedure also uses the decreasing modulus approach (see Section 2). As diagonal entries of H become known, the modulus and its relatively prime factors are reduced. Since d_j divides d_i, for $j \geq i$ (see Corollary 2.2), the

procedure sets $d_j = d_i$ as the modulus in row j, $\forall j \geq i$, until the diagonal entry of H in that row is found.

Applying congruence techniques when determinant information is not used requires knowledge of a bound, say b, on the magnitude of the largest entry in the updated form of the current row. For row iteration i, a set of relatively prime integers $p_1, p_2, \ldots, p_k$ is selected such that $\prod_{t=1}^{k} p_t > 2b$. (Note that the factor 2 is included so as to properly represent negative entries.) Using the entry of maximum magnitude in each K_j, say b_j, one such bound is

$$\max(a_{ij}) \prod_{j=1}^{i-1} b_j n < \prod_{\ell=1}^{k} p_\ell .$$

Additional primes p_t are added to the current set as determined by the above equation where for convenience the factors p_t are selected from the set of prime integers. The number and magnitude of these factors can be varied and can affect the performance of the procedure. Heuristic methods to determine a better bound b are also possible [6], although a verification of the final solution may be required.

The updated residual vector for each entry in row i is then computed by applying the unimodular transformation matrices $K_1, K_2, \ldots, K_{i-1}$ in the indicated order to row i of A. The corresponding radix vector is determined, and operations continue as before. The validity of the congruence relation follows directly from Theorem 2.4.

Each of these four procedures have been applied to 50 randomly generated test problems with known solutions. Initially, all problems have entries bounded by 2^{16}. The real-valued statistics in Table I represent the average value for 10 problems of the stated size. The Solution time and the time used in a linked-integer Division routine are in CPU seconds for an IBM 3081. Length is the number of 16-bit integer words used to represent the determinant value of A. For the two RESROS routines, IR1 indicates the number of vectors of length n required to store the K_i matrices. Iterations specifies the number of column operations performed.

As previously demonstrated in [7], the general modulo d procedures are capable of solving larger problems than the traditional methods with a significantly improved solution time. The versions of KANBAC and ROSSER without modulo d arithmetic require in excess of 3200 bits per entry or more than 60 CPU seconds for problems of size 20 or larger. The introduction of determinant information improves these procedures considerably as shown in Table I. The addition of congruence techniques permits solution of the same problems in standard working precision.

The statistics in Table I demonstrate that the division procedure that explicitly performs the modulo d operation consumes a large portion of the solution time for both KANBAC and ROSSER. The congruence relation procedure RESROS significantly reduces the solution time by implicitly performing the modulo d operation via congruence techniques. These factors were representable in a single integer word so that system routines could perform the modulo operation resulting in a decreased solution time.

Table I. Test Problem Results

Size	Solution	Division	Length	Iterations
		KANBAC		
8	0.26	0.10	1.1	84.0
10	0.70	0.31	1.7	134.0
16	3.38	1.81	2.2	360.0
20	6.64	3.77	2.7	570.0
30	22.74	13.62	2.9	1,305.0
		ROSSER		
8	0.25	0.02	1.1	114.0
10	0.55	0.07	1.7	211.0
16	2.51	0.27	2.2	735.0
20	5.23	0.43	2.7	1,311.0
30	21.60	3.63	2.9	3,578.5
		RESROS		
Size	Solution		IR1	Iterations
8	0.03		43.0	100.4
10	0.07		64.0	233.6
16	0.38		151.0	872.5
20	0.90		231.0	1,740.8
30	3.35		501.4	4,369.8
	RESROS (without determinant information)			
8	0.04		43.0	132.3
10	0.10		64.0	244.6
16	0.55		151.0	981.2
20	1.38		229.0	2,017.4
30	7.71		500.1	7,548.3

Note that the transformation matrices are stored explicitly, and this additional storage for the RESROS approach may be further reduced by storing $K_1K_2 \cdots K_i \bmod(d)$ (see Section 3). Note also that the RESROS routine without determinant information performs considerably better than the first two routines even without determinant information. These results suggest that congruence techniques are effective in determining the Hermite normal form of a matrix with integer entries offering savings in both storage and solution time.

ACKNOWLEDGMENTS

This work is a direct result of my graduate work at Cornell University while under the supervision of L. E. Trotter, Jr. I gratefully acknowledge his contribution in the development and review of this work.

REFERENCES

1. AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass., 1974, 290–291.
2. BLANKENSHIP, W. A. Algorithm 287, Matrix triangularization with integer arithmetic. *Commun. ACM 9* (Sept. 1966), 513.

3. BRADLEY, G. H. Algorithms for Hermite and Smith normal form matrices and linear diophantine equations. *Math. Comput. 25* (1971), 897–907.
4. CHOU, J. T. Algorithms for the solution of systems of linear diophantine equations. Ph.D. dissertation, Dept. of Computer Science, Univ. of Wisconsin, Madison, 1979.
5. DOMICH, P. D. Three new polynomially-time bounded Hermite normal form algorithms. Master's thesis, School of Operations Research and Industrial Engineering, Cornell Univ., Ithaca, N.Y., 1983.
6. DOMICH, P. D. Residual methods for computing Hermite and Smith normal forms. Ph.D. dissertation, School of Operations Research and Industrial Engineering, Cornell Univ., Ithaca, N.Y., 1985.
7. DOMICH, P. D., KANNAN, R., AND TROTTER, L. E., JR. Hermite normal form computation using modulo determinant arithmetic. *Math. Oper. Res. 12* (1987), 50–59.
8. EDMONDS, J. Systems of distinct representatives and linear algebra. *J. Res. N.B.S. 71B* (1967), 241–245.
9. FRUMKIN, M. A. Polynomial time algorithms in the theory of linear diophantine equations. In *Fundamentals of Computation Theory*, M. Karpinski, Ed. *Lecture Notes in Computer Science 56* Springer, New York, 1977, pp. 386–392.
10. HAVAS, G., AND STERLING, L. Integer matrices and Abelian groups, symbolic and algebraic computation. In *Lecture Notes in Computer Science EUROSAM'79, An International Symposium on Symbolic and Algebraic Manipulation* (Marseille, 1979). Springer Verlag, Berlin, 1979, pp. 431–451.
11. HERMITE, C. Sur l'introduction des variables continues dans la theorie des nombres. *J. Reine Angew. Math. 41* (1951), 191–216.
12. HERSTEIN, I. N. *Topics in Algebra.* J. Wiley, New York, 1975, p. 20.
13. KANNAN, R., AND BACHEM, A. Polynomial algorithms for computing and the Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput. 9* (1979), 499–507.
14. KNUTH, D. E. *The Art of Computer Programming.* Vol. 2, *Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 1973.
15. LENSTRA, J. K., RINNOOY KAN, A. H. G., AND VAN EMDE BOAS, P. An appraisal of computational complexity for operations researchers. Mathematisch Centrum BW Tech. Rep. 159/82, Amsterdam, The Netherlands, 1982.
16. LINDAMOND, G. E. Numerical analysis in residue number systems. Computer Science Rep. TR-64-7, Univ. of Maryland, College Park, Md., 1964.
17. MCCLELLAN, M. T. The exact solution of systems of linear equations with polynomial coefficients. *J. ACM 20*, 4 (Oct. 1973), 563–588.
18. POLLARD, S. M. Theorems on factorization and primality testing. *Proc. Camb. Phil. Soc. 76* (1974), 251–258.
19. ROSSER, J. B. A method of computing exact inverse of matrices with integer coefficients. *J. Res. N.B.S. 49* (1952), 349–358.
20. SIMS, C. C. The influence of computers in algebra. In *Proceedings of the Symposium on Applied Mathematics*, 20 (Missoula, Mont., Aug. 1973). American Mathematical Society, Providence, R.I., 1973, pp. 13–30.
21. SMITH, D. A. A basis algorithm for finitely generated Abelian groups. *Math. Algorithms 1* (1966), 13–26.

Received October 1986; revised October 1988; accepted October 1988