MATHEMATICS OF COMPUTATION, VOLUME 25, NUMBER 116, OCTOBER, 1971

Algorithms for Hermite and Smith Normal Matrices and Linear Diophantine Equations

By Gordon H. Bradley

Abstract. New algorithms for constructing the Hermite normal form (triangular) and Smith normal form (diagonal) of an integer matrix are presented. A new algorithm for determining the set of solutions to a system of linear diophantine equations is presented. A modification of the Hermite algorithm gives an integer-preserving algorithm for solving linear equations with real-valued variables. Rough bounds for the number of operations are cubic polynomials involving the order of the matrix and the determinant of the matrix. The algorithms are valid if the elements of the matrix are in a principal ideal domain.

1. Introduction. Hermite [16] showed that a nonsingular integer matrix could be transformed to a triangular matrix using elementary column operations over the ring of integers. Smith [26] showed that elementary row and column operations can diagonalize an integer matrix. The constructions are valid if the elements of the matrix are in a principal ideal domain, see [18, pp. 79–84]. The constructions have application to many problems in pure and applied mathematics: for example, constructive proof of the basis theorem for finitely generated Abelian groups [28, pp. 106–113], [25]; invariant polynomials and elementary divisors of polynomial matrices [13, pp. 137–139], [27, pp. 23–25], [29, pp. 222–225]; triangular bases for lattices in the geometry of numbers [10, pp. 9–13]; modules in mathematical system theory [19, pp. 325–331], [29, pp. 222–225]; integer linear programming [6], [9], [14], [17, pp. 317–354], [24]; and solving systems of linear diophantine equations [3], [6], [15, pp. 67–76].

The algorithms developed in [6], [9], [14], [17], [24] for computing optimal solutions to integer linear programs involve the construction of the Hermite normal form or the Smith normal form of an integer matrix. Since the matrices associated with integer linear programming problems are often large, there is current interest in developing more efficient algorithms for constructing the Hermite and Smith normal forms.

The calculations are basically Gaussian elimination over a Euclidean ring with greatest common divisor calculations replacing division. The new algorithms compute the gcd's by means of a new algorithm [7], [8] for computing the gcd of k integers and multipliers for which the amount of computation and storage space required is linear in k. For the new Hermite algorithm, the number of arithmetic operations is roughly bounded by the order of the matrix cubed. A bound for the Smith algorithm involves the order of the matrix and the determinant. A bound for the diophantine equation

Received June 19, 1970, revised December 21, 1970.

AMS 1970 *subject classifications.* Primary 10B05, 10C05, 10C99, 65F05; Secondary 15A06, 15A36, 90C10.

Key words and phrases. Hermite normal form, Smith normal form, linear diophantine equations, integer-preserving Gaussian elimination, integer matrix algorithm.

Copyright © 1971, American Mathematical Society

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

algorithm is a cubic polynomial in the number of variables and the number of constraints.

Related to calculations over integral domains is the problem of solving, error-free, systems of linear equations with real-valued variables. Methods for this problem include the classical technique of cross multiplying [11, pp. 82–87], [4]; methods related to greatest common divisor calculations [23]; congruence methods [5], [22]; and multistep Gaussian elimination [1]. A modification of the new Hermite algorithm presented here gives an integer-preserving Gaussian elimination method that is related to the algorithm presented in [23].

2. Hermite Normal Form. A nonsingular integer matrix with integer inverse is called a unimodular matrix. An integer matrix is unimodular if and only if its determinant is equal to plus one or minus one. Postmultiplying a matrix by a unimodular matrix is equivalent to a series of elementary column operations over the ring of integers, that is, (1) adding an integer multiple of one column to another column, (2) multiplying a column by minus one, and (3) interchanging columns.

THEOREM (HERMITE [16]). *Given a nonsingular* $(n) \times (n)$ *integer matrix* C, *there exists an* $(n) \times (n)$ *unimodular matrix* K *such that* CK *is lower triangular with positive diagonal elements. Further, each off-diagonal element of* CK *is nonpositive and strictly less in absolute value than the diagonal element in its row.*

CK is called the Hermite normal form of C. Let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x and let $\lceil x \rceil$ denote the least integer greater than or equal to x. Let $\operatorname{sgn}[x]$ be -1, 0 or 1 according to whether x is negative, zero or positive.

The new algorithms differ from the previously published algorithms only in the way that elementary column operations are used to construct the diagonal elements of the matrix. It will be convenient to define subroutines to perform this operation and then to describe the common operations of the new and previously published algorithms in a single algorithm.

Subroutine-Row (C, i, n, m). Elementary column operations are performed on the ith, $(i+1)$th, $\cdots$, nth columns of C until c_{ii} divides c_{ij}, $j = i+1, \cdots, n$. Assume not all c_{ij}, $j = i, i+1, \cdots, n$, are equal to zero. The column operations involve rows i to m of C.

1. Determine k $(k = i, i+1, \cdots, n)$ such that $0 < |c_{ik}| \leqq |c_{ij}|$, $j = i, i+1, \cdots, n$.
2. For $j = i, i+1, \cdots, m$, do
   a. $c_{jk} = \operatorname{sgn}[c_{ik}]c_{jk}$,
   b. exchange c_{jk} and c_{ji}.
3. If $c_{ii} \mid c_{ij}$, $j = i+1, \cdots, n$, return.
4. If c_{ii} does not divide c_{ik}, let $d = \lfloor c_{ik}/c_{ii} \rfloor$.
5. For $j = i, i+1, \cdots, m$, do
   a. $c_{jk} = c_{jk} - dc_{ji}$,
   b. exchange c_{jk} and c_{ji}.
6. Go to 3.

Proof. We need to show that the loop consisting of steps 3-4-5-6 can be repeated only a finite number of times. At the completion of step 5, the new c_{ii} is positive and strictly less than the old c_{ii} (since the old c_{ii} did not divide the old c_{ik} and since $0 \leqq c_{ik} - \lfloor c_{ik}/c_{ii} \rfloor c_{ii} < c_{ii}$). Clearly, this can happen only a finite number of times. □

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

The previously published algorithms for constructing the Hermite normal form of an integer matrix can be described as follows.

Algorithm H [2].

I. *Triangular Form.* The matrix is transformed row by row. Begin with $i = 1$.

1. If $c_{ij} = 0$, for $j = i, i + 1, \cdots, n$, go to 4.
2. Call subroutine-row (C, i, n, n).
3. For $j = i + 1, i + 2, \cdots, n$, do
   a. $d = c_{ij}/c_{ii}$,
   b. $c_{kj} = c_{kj} - dc_{ki}$, for $k = i, i + 1, \cdots, n$.
4. If $i < n - 1$, replace i by $i + 1$ and go to 1.
5. $c_{nn} = \operatorname{sgn}(c_{nn})c_{nn}$.

If any diagonal element is zero, then the matrix C is singular and the Hermite normal form, as defined above, does not exist. If C is nonsingular, proceed to II.

II. *Reduction of Off-Diagonal Elements.* The elements are reduced row by row. Begin with $i = 2$.

6. For $j = 1, 2, \cdots, i - 1$, do
   a. $d = \lceil c_{ij}/c_{ii} \rceil$,
   b. $c_{kj} = c_{kj} - dc_{ki}$, for $k = i, i + 1, \cdots, n$.
7. If $i < n$, replace i by $i + 1$ and go to 6. Otherwise, stop.

The calculations of subroutine-row are equivalent to the positive remainder version of the Euclidean algorithm for the greatest common divisor of n integers. If subroutine-row involves k_i iterations of step 5, then algorithm H requires $n^3/2 + nk_1 + (n - 1)k_2 + \cdots + 2k_{n-1}$ multiplications. (For the algorithms, we will count the number of multiplications, in all cases there is an equal number of additions. Divisions, comparisons and exchanges of columns can be ignored.) Since the k_i may be arbitrarily large, it is impossible to develop a bound for the number of multiplications that does not involve the k_i.*

The new subroutine-row utilizes a new version of the Euclidean algorithm (EA) [7], [8] for computing the gcd of n integers and multipliers x_i such that gcd $= \sum_{i=1}^{n} x_i a_i$. If k is the number of iterations of the Euclidean algorithm, then EA requires at most $2k + 3(n - 1)$ multiplications, $2k + n - 1$ additions and $k + n - 1$ divisions.

New Subroutine-Row (C, i, n, m). Elementary column operations are performed on the ith, $(i + 1)$th, $\cdots$, nth columns of C until c_{ii} divides c_{ij}, $j = i + 1, \cdots, n$. Assume not all c_{ij}, $j = i, i + 1, \cdots, n$, are equal to zero. Column operations involve rows i to m of C.

1. Determine k $(k = i, i + 1, \cdots, n)$ such that $0 < |c_{ik}| \leqq |c_{ii}|$, $j = i, i + 1, \cdots, n$ for $c_{ij} \neq 0$.
2. For $j = i, i + 1, \cdots, m$, do
   a. $c_{jk} = \operatorname{sgn}[c_{ik}]c_{jk}$,
   b. exchange c_{jk} and c_{ji}.
3. Via EA calculate multipliers such that $\gcd(c_{ii}, c_{i,i+1}, \cdots, c_{in}) = \sum_{j=i}^{n} x_j c_{ij}$.
4. If for some k, $x_k = 1$ or $c_k = 0$, go to 12.
5. If for some k, $x_k = -1$, then let $c_{jk} = -c_{jk}$ for $j = i, i + 1, \cdots, m$. Go to 12.

---

* In [7], a bound for the number of iterations for computing the gcd of n integers is given in terms of the integer with the smallest nonzero absolute value, but this bound is useful only for k_1 since the magnitude of the numbers that result after row 1 has been cleared cannot, in general, be predicted.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

6. If for some k ($k = i + 1, \cdots, n$) c_{ii} divides c_{ik}, let $x_i = x_i - c_{ik}/c_{ii}$. Go to 12.

7. If for some k, $x_k = 0$, let $d = c_{ik}/\gcd(c_{ii}, c_{i,i+1}, \cdots, c_{in})$. Let $x_j = (1 - d)x_j$ for $j = i, i + 1, \cdots, n$. Go to 12.

8. Via EA calculate $\gcd(c_{ii}, c_{i,i+1})$ and multipliers y_1, y_2 such that $\gcd(c_{ii}, c_{i,i+1}) = y_1c_{ii} + y_2c_{i,i+1}$.

9. Let $z_1 = -c_{i,i+1}/\gcd(c_{ii}, c_{i,i+1})$. Let $z_2 = c_{ii}/\gcd(c_{ii}, c_{i,i+1})$.

10. For $j = i, i + 1, \cdots, m$, do
    a. $d = y_1c_{ji} + y_2c_{j,i+1}$,
    b. $c_{j,i+1} = z_1c_{ji} + z_2c_{j,i+1}$,
    c. $c_{ji} = d$.

11. Go to 3.

12. For $j = i, i + 1, \cdots, n$, $j \neq k$, do
    a. $c_{hk} = c_{hk} - x_jc_{hj}$ for $h = i, i + 1, \cdots, m$.

13. Exchange c_{ji} and c_{jk} for $j = i, i + 1, \cdots, m$.

14. Return.

Proof. The new algorithm is finite since step 10 places a zero in element $c_{i,i+1}$ and hence the condition of step 4 will hold with $k = i + 1$. The construction of step 10 is valid since

$$\begin{bmatrix} y_1 & z_1 \\ y_2 & z_2 \end{bmatrix}$$

is unimodular. In step 6 it is assumed that $x_k = 0$; since c_{ii} divides c_{ik}, EA (and every other gcd algorithm known to the author) will construct multipliers with $x_k = 0$. □

Note. After exit from step 11 to step 3, only the multipliers x_1 and x_2 need be recalculated. New $x_2 = 0$ and new x_1 can be immediately recovered from the working storage of EA ([7, p. 434] Section 3, step 2.2, new $x_1 = y_3'$).

New algorithm H is algorithm H with new subroutine-row replacing subroutine-row. For new subroutine-row, let p_i be the number of times an integer multiple of one column is added to another column in constructing row i. Then p_i is equal to the number of nonzero x_j (step 12) plus, if step 10 is executed, the number of nonzero integers among y_1, y_2, z_1, z_2. Hence $p_i \leqq n - i + 4$. Recall that k_i is the number of times an integer multiple of one column is added to another column in subroutine-row for row i. Except for an unlikely special case,** $p_i \leqq k_i$. Usually p_i is much less than k_i. For example, for the row (13, 26, 47, 50), $p_i = 2$ and $k_i = 5$.

It is possible to compare the new algorithm and the previously published algorithms by noting that k_i is the number of iterations for calculating the greatest common divisor of $c_{ii}, c_{i,i+1}, \cdots, c_{in}$ by means of the positive remainder version of the Euclidean algorithm. For a row with two nonzero integers c_{ii}, $c_{i,i+1}$ where $|c_{ii}| \leqq |c_{i,i+1}|$, the expected value of k_i is $1.9405 \log_{10} |c_{ii}|$ [20, pp. 320–333]; for the new algorithm, $p_i \leqq 2$ for $i < n - 1$ and $p_i \leqq 4$ for $i = n - 1$. For h nonzero integers, $h > 2$, where $|c_{ii}| \leqq |c_{ij}|$, $j = i, \cdots, i + h - 1$, a bound for k_i is $\log_{1.6} |c_{ii}|$ [7, p. 435] (the expected value is not known but is certainly greater than the expected value

** For a special case that may arise when step 10 is executed, $p_i = k_i + 1$. Although the special case can be detected and the new algorithm can be modified so that $p_i = k_i$, the extra effort does not seem to be justified because the case is unlikely.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

for $h = 2$). For the new algorithm, $p_i \leqq \min(h, \log_2 |c_{ii}|)$ for $i \leqq n - h$ and $p_i \leqq \min(h + 3, \log_2 |c_{ii}|)$ for $i = n - h + 1$.

Although the bound on p_i ($p_i \leqq n - i + 4$) is sharp, the expected value of p_i is much less than the bound. The bound assumes that the greatest common divisor of $c_{ii}, c_{i,i+1}, \cdots, c_{in}$ involves $n - i + 1$ multipliers, none of which is equal to -1, 0 or 1. It should be expected that the gcd of some small subset of the c_{ij}'s will equal the gcd of all the c_{ij}'s and hence there will be only a small number of nonzero multipliers. This claim is supported by the result of Cesaro, see [20, p. 301], that shows that the gcd of two randomly chosen integers is 1 with probability $6/\pi^2$ ($6/\pi^2 > .6$). If $\gcd(c_{ii}, c_{i,i+1}) = 1$, then $p_i = 2$. Thus, under the hypothesis that $c_{ii}, c_{i,i+1}$ may be viewed as random integers, the expected value of p_i is less than $(.6)2 + (.4)(n - i + 4) = 2.8 + (.4)(n - i)$.

In many practical applications, particularly integer linear programming it is very desirable to have bounds for the amount of computation in terms of n. The new algorithm is bounded by $n^3 + 2(k_1 + \cdots + k_{n-1})$. Ignoring the k_i, the number of operations is bounded by n^3 which is only twice the number of operations for Gaussian elimination over the field of real numbers.

3. Smith Normal Form.

THEOREM (SMITH [26]). *Given a nonsingular* $(n) \times (n)$ *integer matrix* C, *there exist* $(n) \times (n)$ *unimodular matrices* E, F *such that* $D = ECF$ *is a diagonal matrix with positive diagonal elements such that* $d_{11} |d_{22}| \cdots |d_{nn}$.

Define subroutine-column (C, i, n, m) and new subroutine-column (C, i, n, m) to be the column analogs subroutine-row (C, i, n, m) and new subroutine-row (C, i, n, m). That is, the column subroutines perform elementary row operations on the ith, $(i + 1)$th, $\cdots$, nth rows of C until c_{ii} divides c_{ji}, $j = i + 1, \cdots, n$, where row operations involve columns i to m.

The following algorithm due to D. A. Smith [25] and Hu [17, pp. 377–381] is an improvement of previous algorithms, see for example [12], [13, pp. 137–139], [18, pp. 79–84], [21, pp. 226–235], [28, pp. 106–109], [29, pp. 222–224].

Algorithm S [17], [25].

I. *Diagonal Form.* The diagonal form is constructed row by row. Begin with $i = 1$.

1. If $d_{ij} = 0$, for $j = i, i + 1, \cdots, n$, go to 9.
2. Call subroutine-row (D, i, n, n).
3. If d_{ii} divides d_{ji}, $j = i + 1, i + 2, \cdots, n$, go to 7.
4. Call subroutine-column (D, i, n, n).
5. If d_{ii} divides d_{ij}, $j = i + 1, i + 2, \cdots, n$, go to 7.
6. Go to 2.
7. For $j = i + 1, i + 2, \cdots, n$, do
   a. $c = d_{ij}/d_{ii}$,
   b. $d_{kj} = d_{kj} - cd_{ki}$, for $k = i, i + 1, \cdots, n$.
8. Set $d_{ji} = 0$, for $j = i + 1, i + 2, \cdots, n$.
9. If $i < n - 1$, replace i by $i + 1$ and go to 1.
10. $d_{nn} = \operatorname{sgn}(d_{nn})d_{nn}$. Stop.

If any diagonal element is zero, the matrix is singular and the Smith normal form, as defined above, does not exist. If D is nonsingular, proceed to part II.

II. $d_{11} |d_{22}| \cdots |d_{nn}$. The construction proceeds row by row. Begin with $i = 1$.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

11. If d_{ii} divides d_{ij}, for $j = i + 1, i + 2, \cdots, n$, go to 15.
12. If d_{ii} does not divide d_{kk}, calculate $\gcd(d_{ii}, d_{kk})$ via EA. Let $c = \gcd(d_{ii}, d_{kk})$.
13. Let $d_{kk} = d_{ii}d_{kk}/c$. Let $d_{ii} = c$.
14. Go to 11.
15. If $i < n - 1$, replace i by $i + 1$ and go to 11. Otherwise stop.

Proof. Part I is finite because each time subroutine-row (D, i, n, n) or subroutine-column (D, i, n, n) is called, c_{ii} is strictly decreased while remaining positive. Part II is finite because d_{ii} is strictly decreased by step 13 while remaining positive. Let x_1, x_2 be such that $c = x_1c_{ii} + x_2c_{kk}$. Step 13 is valid because it can be accomplished by the following elementary row and column operations: (1) add x_1 times row i to row k, (2) add x_2 times column k to column i, (3) exchange row i and row k, (4) add $-d_{kk}/c$ times column i to column k, (5) add $-d_{ii}/c$ times row i to row k and multiply column k by minus one. □

The new algorithm S is formed from algorithm S by replacing the subroutines with the new subroutines. Unfortunately, it is not possible to bound the number of multiplications in terms of n alone. It is not possible to bound the number of times the loop consisting of steps 2-3-4-5-6 is repeated without additional information. In order to bound the calculations, it is necessary to have information about the magnitude of the integers as the algorithm proceeds. When constructing the ith diagonal element, let q_i be equal to the magnitude of the element in row i with the smallest nonzero absolute value. Then the 2-3-4-5-6 loop will require at most $(n - i + 1) \log_2 q_i$ multiplications (each column multiplication in the subroutines reduces c_{ii} to less than one half its previous value).

It is possible to develop a bound in terms of n and the determinant of C. After new subroutine-row has been called for the first time for row i, c_{ii} is not greater than $|\det(C)|$ (if the matrix is triangulated by algorithm H, the product of c_{ii} and the other diagonal elements is $|\det(C)|$). Hence, $(n - i + 1)(\log_2 |\det(C)| + n - i + 4)$ is a bound for the number of multiplications in the 2-3-4-5-6 loop. A rough bound for the number of multiplications in part I of the new algorithm S is then

$$n^2(\log_2 |\det(C)| + 3)/2 + 2n^3/3.$$

Part II will, in general, require only a small number of multiplications. If the value of $\det(C)$ is not known, a theorem due to Hadamard gives the bound

$$|\det(C)| \leqq \prod_{j=1}^{n} \left(\sum_{i=1}^{n} c_{ij}^2 \right)^{1/2}.$$

In general, for row i $(i \ll n)$, c_{ii} will equal 1 after new subroutine-row (C, i, n, n) has been called for the first time; hence, a good approximate bound is $3n^2/2 + 2n^3/3$.

Some applications of the Smith construction to integer linear programming involve matrices with elements that are integers mod k. Thus, each number in the matrix may be reduced to a number strictly less than k and bounds for new algorithm S can be developed in terms of $\log_2 k$.

4. Linear Diophantine Equations. Given a system of linear diophantine equations

$$Ax = b; \qquad x \text{ integer}, \tag{1}$$

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

where A is an $(m) \times (n)$ integer matrix and b is an integer m-vector, it is possible to calculate the set of all solutions to (1) or to show that no solution exists, see for example [3], [15, pp. 67–76], [20, pp. 303–304]. The algorithm of [3], [15] involves the calculations of algorithm H. Form the matrix

$$C = \begin{bmatrix} A & -b \\ I & 0 \end{bmatrix},$$

C is $(m + n) \times (n + 1)$. In order to avoid some notational complications when rank $A < m$, the algorithm below includes an operation "delete row i" which means delete row i of the matrix and relabel the number of each row accordingly.

Algorithm DE [3]. Transform the matrix row by row. Begin with $i = 1$.

1. If $c_{ij} = 0$, for $j = i, i + 1, \cdots, n + 1$, go to 10.
2. If $c_{ij} = 0$, for $j = i, i + 1, \cdots, n$, and $c_{i,n+1} \neq 0$, then (1) has no solution. Stop.
3. Call subroutine-row $(C, i, n, n + m)$.
4. If c_{ii} does not divide $c_{i,n+1}$, then (1) has no solution. Stop.
5. For $j = i + 1, i + 2, \cdots, n + 1$, do
   a. $d = c_{ij}/c_{ii}$,
   b. $c_{kj} = c_{kj} - dc_{ki}$, for $k = i, i + 1, \cdots, m + n$.
6. If $i < m$, replace i by $i + 1$. Otherwise stop.
7. If $i \leqq n$, go to 1.
8. If $c_{k,n+1} = 0$ for $k = n + 1, n + 2, \cdots, m$, then delete rows $n + 1, n + 2, \cdots, m$ and stop.
9. If $c_{k,n+1} \neq 0$, for some $n + 1 \leqq k \leqq m$, then (1) has no solution. Stop.
10. Delete row i, replace m with $m - 1$. If $i \leqq m$, go to 1, otherwise stop.

If the algorithm does not indicate that (1) has no solution, then the algorithm constructs the matrix

$$\begin{bmatrix} T & \phi & 0 \\ K_1 & K_2 & f \end{bmatrix}$$

where T is an $(p) \times (p)$ lower triangular matrix with positive diagonal elements (p is rank of A), K_2 is $(n) \times (n - p)$ and f is an n-vector. The vector f is a solution to (1) and the set of all solutions to (1) is given by $\{x: x = f + K_2 z, z \text{ integer}\}$. K_2 has rank $n - p$, thus the set of solutions to (1) is either void or has dimension $n - p$. Note that if $p = n$, then f is the unique solution to (1).

To see why the algorithm constructs the set of solutions to (1), let

$$x = [K_1 \ K_2] \begin{bmatrix} y^1 \\ y^2 \end{bmatrix} + f. \tag{2}$$

Since only elementary column operations have been used in the algorithm, $[K_1 \ K_2]$ is unimodular and therefore (2) defines a 1-1 mapping of the integers in R^n onto themselves. Substitute (2) into (1) and delete the rows deleted by the algorithm, the result is $Ty^1 = 0$. Since T is triangular with positive diagonal elements, (2) defines a solution to (1) if and only if $y^1 = 0$. A similar argument shows that if the algorithm terminates at steps 2, 4 or 9, then (1) has no solution.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

New algorithm DE is algorithm DE with new subroutine-row (C, i, n, m). Ignoring the two multiplications per iteration of the Euclidean algorithm, the number of multiplications for the new algorithm DE is bounded by a cubic polynomial in m and n.

5. Integer-Preserving Gaussian Elimination. Given a system of linear equations with integer coefficients in real-valued variables,

$$Ax = b \tag{3}$$

where A is $(m) \times (n)$, it is sometimes necessary to calculate the solution (or set of solutions) error-free. For digital computers this means the solution technique cannot produce fractions. A modification of algorithm H gives an integer preserving Gaussian elimination method that is similar to the approach of [23]. The algorithm presented here seeks to minimize the number of operations and the amount of storage; the algorithm in [23], which was developed in 1952 for calculating machines, seeks to limit the size of the integers in order to minimize the amount of human time.

Form the matrix $C = [A\ b]$; the row analog of algorithm H together with the operation of interchanging any of the first n columns of C can reduce C to

$$\begin{bmatrix} T & S & d_1 \\ \phi & \phi & d_2 \end{bmatrix}$$

where T is an $(p) \times (p)$ upper triangular matrix with positive diagonal elements (p is the rank of A), S is an $(p) \times (n-p)$ matrix, d_1 is an p-vector and d_2 is an $(m-p)$-vector. Then (3) has a solution if and only if $d_2 = 0$.

Partition x in an obvious manner to get

$$Tx^1 = d_1 - Sx^2. \tag{4}$$

Let $\delta = \prod_{i=1}^{p} t_{ii}$. Multiply each element of d_1 and S by δ; this is equivalent to the substitution $x^1 = z^1/\delta$. It is then possible to do the back-substitution without producing fractions. The result is

$$z^1 = f + Rx^2 \quad \text{(or)} \quad x^1 = (1/\delta)f + (1/\delta)Rx^2. \tag{5}$$

The set of solutions to (3) is given by

$$\left\{ \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} : x^1 = (1/\delta)f + (1/\delta)Rx^2,\ x^2 \text{ arbitrary} \right\}$$

where f, R and δ are integer.

The number of operations for the integer-preserving algorithm is bounded roughly by a cubic polynomial in m and n. The cross multiply algorithm has the same bound. However, the algorithm presented here should, in general, take fewer iterations than the bound would indicate and so should be expected to be better than the cross multiply method. Recently, Bareiss [1] has proposed a multistep integer-preserving Gaussian elimination algorithm which is a major improvement over the cross multiply algorithm (which is a single step Gaussian elimination). The bounds for the multistep Gaussian elimination algorithms are better than the bounds for the algorithm presented here; computer testing would be necessary to compare the actual number of operations for the two algorithms.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

The algorithm of this section can solve (1) when rank $A = n$; in this case the algorithm of this section should be used since it involves fewer operations than algorithm DE. An obvious generalization of the algorithm presented here can be used to compute the inverse of an integer matrix error-free.

6. **Numerical Example.** The following matrix will be converted to Hermite normal form by means of new algorithm H. Steps of the algorithm will be noted with primes indicating the steps of new subroutine-row (step 7′ of new subroutine-row is deleted for reasons discussed below in Section 7).

$$C = \begin{bmatrix} 13 & 21 & 0 & -37 \\ 10 & 20 & -15 & 0 \\ 1 & 3 & 2 & 1 \\ 7 & -1 & -1 & 0 \end{bmatrix} \rightarrow (\text{Steps } 1, 2, 1', 2').$$

Step 3′ constructs multipliers $x_1 = -8$, $x_2 = 5$, $x_3 = x_4 = 0$. At step 4′, $k = 3$.

$$(3', 4', 12') \rightarrow \begin{bmatrix} 13 & 21 & 1 & -37 \\ 10 & 20 & 5 & 0 \\ 1 & 3 & 9 & 1 \\ 7 & -1 & -62 & 0 \end{bmatrix} \rightarrow (13', 3)$$

$$\rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & -85 & -55 & 185 \\ 9 & -186 & -116 & 334 \\ -62 & 1301 & 813 & -2294 \end{bmatrix}.$$

Step 3′ constructs multipliers $x_2 = -2$, $x_3 = 3$, $x_4 = 0$. The conditions of steps 4′, 5′, 6′ are not met; steps 8′ and 9′ calculate $\gcd(-85, -55) = 5$; $y_1 = -2$, $y_2 = 3$, $z_1 = 11$, $z_2 = -17$.

$$(10') \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & 5 & 0 & 185 \\ 9 & 24 & -74 & 334 \\ -62 & -163 & 490 & -2294 \end{bmatrix} \rightarrow (11', 3').$$

Step 3′ constructs multipliers $x_2 = 1$, $x_3 = x_4 = 0$. At step 4′, $k = 2$.

$$(12', 3) \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & 5 & 0 & 0 \\ 9 & 24 & -74 & -554 \\ -62 & -163 & 490 & 3737 \end{bmatrix} \rightarrow (4, 1, 2, 1', 2', 3').$$

Step 3′ constructs multipliers $x_3 = -15$, $x_4 = 2$. The conditions of steps 4′, 5′, 6′ are

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

not met; step 8′ calculates $\gcd(-74, -554) = 2$; $y_1 = -15$, $y_2 = 2$, $z_1 = 277$, $z_2 = -37$.

$$(10') \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & 5 & 0 & 0 \\ 9 & 24 & 2 & 0 \\ -62 & -163 & 124 & -2539 \end{bmatrix}.$$

The final Hermite normal form is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ -1 & 0 & 2 & 0 \\ -1570 & -1651 & -2415 & 2539 \end{bmatrix}.$$

7. Numerical Considerations. The possible increase in the magnitude of the elements of the matrix as the algorithm proceeds is a serious computational consideration (especially if the determinant of the matrix is large), see [2], [23] and [25]. The magnitudes of the gcd multipliers constructed by EA for new algorithms H, S and DE are, in general, large, relative to the magnitude of the integers (the same multipliers are implicitly constructed in the previously published algorithms). A minor modification of EA which is described in [7, p. 435] will construct "minimal multipliers." The minimal multipliers are smaller in magnitude and will include, in general, fewer nonzero integers than the multipliers constructed by the unmodified version of EA. The amount of additional computation needed to construct minimal multipliers is linear in n.

Step 7 of new subroutine-row (C, i, n, m) will increase the size of the numbers if d is large. The elimination of 7 does not change the bounds for the algorithms but may involve 3 or 4 extra column multiplications.

A FORTRAN implementation of new algorithm H is available from the author. The program was used for the integer linear programming test problems reported in [9].

8. Acknowledgment. The author would like to thank Professors John D. Murchland and Ellis Horowitz for valuable suggestions and references. Comments by a referee aided the development of the integer-preserving algorithm for linear equations. Suggestions by a referee greatly improved the exposition.

Administrative Sciences Department
Yale University
New Haven, Connecticut 06520

1. E. H. Bareiss, "Sylvester's identity and multistep integer-preserving Gaussian elimination," *Math. Comp.*, v. 22, 1968, pp. 565–578. MR **37** #2416.
2. W. A. Blankinship, "Algorithm 287, matrix triangulation with integer arithmetic [F1]," *Comm. ACM*, v. 9, 1966, p. 513.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use

3. W. A. BLANKINSHIP, "Algorithm 288, solution of simultaneous linear diophantine equations [F4]," *Comm. ACM*, v. 9, 1966, p. 514.

4. J. BOOTHROYD, "Algorithm 290, linear equations, exact solutions," *Comm. ACM*, v. 9, 1966, pp. 383–384.

5. I. BOROSH & A. S. FRAENKEL, "Exact solutions of linear equations with rational coefficients by congruence techniques," *Math. Comp.*, v. 20, 1966, pp. 107–112. MR **32** #4831.

6. G. H. BRADLEY, "Equivalent integer programs and canonical problems," *Management Sci.*, v. 16, 1970, pp. 354–366.

7. G. H. BRADLEY, "Algorithm and bound for the greastest common divisor of n integers," *Comm. ACM*, v. 13, 1970, pp. 433–436.

8. G. H. BRADLEY, "Algorithm 386, greatest common divisor of n integers and multipliers [A1]," *Comm. ACM*, v. 13, 1970, p. 447.

9. G. H. BRADLEY & P. N. WAHI, *An Algorithm for Integer Linear Programming: A Combined Algebraic and Enumeration Approach*, Report #29, Administrative Sciences Dept., Yale University, New Haven, Conn., 1969.

10. J. W. S. CASSELS, *An Introduction to the Geometry of Numbers*, Die Grundlehren der math. Wissenschaften, Band 99, Springer-Verlag, Berlin, 1959. MR **28** #1175.

11. L. FOX, *An Introduction to Numerical Linear Algebra*, Monographs on Numerical Anal., Clarendon Press, Oxford, 1964. MR **29** #1733.

12. C.-E. FRÖBERG & A. SUNDSTRÖM, "Contribution no. 20, Smith's normal form," *Nordisk Tidskr. Informations-Behandling (BIT)*, v. 7, 1967, pp. 163–169.

13. F. R. GANTMAHER, *The Theory of Matrices*, GITTL, Moscow, 1953; English transl., Chelsea, New York, 1959. MR **16**, 438.

14. R. E. GOMORY, "On the relation between integer and noninteger solutions to linear programs," *Proc. Nat. Acad. Sci. U.S.A.*, v. 53, 1965, pp. 260–265. MR **31** #6677.

15. R. H. GONZÁLEZ-ZUBIETA, *On Some Aspects of Integer Linear Programming*, Technical Report #16, Center for Operations Res., M.I.T., Cambridge, Mass., 1965.

16. C. HERMITE, "Sur l'introduction des variables continues dans la théorie des nombres," *J. Reine Angew. Math.*, v. 41, 1851, pp. 191–216.

17. T. C. HU, *Integer Programming and Network Flows*, Addison-Wesley, Reading, Mass., 1969.

18. N. JACOBSON, *Lectures in Abstract Algebra*. Vol. II. *Linear Algebra*, Van Nostrand, Princeton, N. J., 1953. MR **14**, 837.

19. R. E. KALLMAN, P. L. FALB & M. A. ARBIB, *Topics in Mathematical System Theory*, McGraw-Hill, New York, 1969.

20. D. E. KNUTH, *The Art of Computer Programming*: Vol. II. *Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969.

21. C. C. MACDUFFEE, *An Introduction to Abstract Algebra*, Wiley, New York, 1940. MR **2**, 241.

22. M. NEWMAN, "Solving equations exactly," *J. Res. Nat. Bur. Standards Sect. B*, v. 71, 1967, pp. 171–179. MR **37** #2421.

23. J. B. ROSSER, "A method of computing exact inverses of matrices with integer coefficients," *J. Res. Nat. Bur. Standards*, v. 49, 1952, pp. 349–358. MR **14**, 1128.

24. J. F. SHAPIRO, "Dynamic programming algorithms for the integer programming problem—I. The integer programming problem viewed as a knapsack problem," *Operations Res.*, v. 16, 1968, pp. 103–121. MR **38** #921.

25. D. A. SMITH, "A basis algorithm for finitely generated abelian groups," *Math. Algorithms*, v. 1, 1966, pp. 13–26. MR **40** #4367.

26. H. J. S. SMITH, "On systems of linear indeterminate equations and congruences," *Philos. Trans. Roy. Soc. London Ser. A*, v. 151, 1861, pp. 293–326.

27. H. W. TURNBULL & A. C. AITKEN, *An Introduction to the Theory of Canonical Matrices*, Blackie, London, 1932.

28. B. L. VAN DER WAERDEN, *Moderne Algebra*. Vol. 2, Springer, Berlin, 1931; English transl., Ungar, New York, 1950.

29. L. A. ZADEH & E. POLAK, *Systems Theory*, McGraw-Hill, New York, 1969.

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use