EXACT SOLUTION OF LINEAR EQUATIONS

by

Stanley Cabay

DEPARTMENT OF COMPUTER SCIENCE
University of Toronto
Toronto 181, Ontario

Abstract

The congruential method of obtaining the exact solution of a system of linear equations with integral coefficients is critically reviewed. A new and efficient test for checking that a sequence of residue solutions determines the correct integer solution of the system of equations is presented. Also discussed is an improved method for finding the adjoint of a singular matrix.

1. Introduction

The numerous proposed methods for finding the approximate solution to a system of n linear equations, $Ax = b$, are inadequate when exact solutions are required, or when the coefficient matrix, A, is "ill-conditioned". Methods for obtaining the exact solution of a system of equations with integral coefficients (solving systems with rational or floating-point coefficients can be reduced to solving systems with integer coefficients by appropriate scaling) fill this gap. Since research on exact methods has been of a rather independent and scattered nature, a review of the literature is justified.

The components of the exact solution of a system of equations with integral coefficients will normally be rational numbers with large numerators and denominators. This suggests that multiple-precision operations are inherent in exact methods. Since multiple-precision operations are relatively slow, it should be the intent of exact methods to minimize the use of such operations.

Non-congruential methods fall far short of this goal. Elimination of variables by cross-multiplication only aggravates the problem since this causes a tremendous growth of the size of intermediate results (see Rosser[16]). Rosser develops an alternative method which controls the growth of intermediate results but which introduces many more computations on, now smaller, multiple-precision numbers. Briefly, his technique of eliminating a variable corresponds to finding the greatest common divisor of the n multiple-precision coefficients of that variable. The algorithm of Blankinship[3] which eliminates the rows, rather than the columns, of the augmented matrix suffers from the same defects as does Rosser's.

The fraction-free (integer-preserving) method described by Bodewig[4] and by Fox[8] is a significant attempt to minimize the size of intermediate results. The method is essentially that of Gaussian elimination, except that at each stage of the elimination a common factor is systematically removed from the transformed matrix. Luther and Guseman[14] basically rediscover the fraction-free technique for computing the adjoint of an integer matrix. More recently, Bareiss[1] has described a variant of the method which by eliminating two variables at-a-time (rather than one) improves the efficiency of the elimination considerably. The fraction-free method, however, still requires multiple-precision arithmetic for most of the computation.

The significant computational advantages accruing from the congruential approach of obtaining the exact solution of a system of linear equations were discussed by Takahasi and Ishibashi[18] in 1961. More recently, Borosh and Fraenkel[5], and Newman[15] have cultivated the approach and suggested improvements. The congruential method permits the effective removal of all multiple-precision computations except at the initial and final steps of the process.

The method consists of converting the given system of linear equations with integral coefficients to a system of congruences modulo a number of primes. Multiple-precision computations will be necessary to perform this step if the coefficients of the augmented matrix are larger than the capacity of the accumulator. A solution is then obtained for each system of congruences, using a modified Gaussian elimination process. Only single-precision computations are required by the elimination process if the primes are selected to be within the capacity of the accumulator. Having computed $x_1, x_2, \ldots, x_{k-1}$ the solution of the system of congruences (mod p_k) with the aid of the Chinese Remainder theorem (see Knuth[12]), determines at the kth stage another term, x_k, of the mixed-radix representation (see Knuth[12]) of x:

$$x = x_1 + p_1 x_2 + \ldots + p_1 p_2 \ldots p_{m-1} x_m. \quad (1.1)$$

Evaluation of the expression (1.1), the final step, may again require multiple-precision computation.

2. Description of the Method

Let A be an $n \times n$ nonsingular integral matrix, and b an integral $n \times 1$ vector. Denote the determinant of A by d and the adjoint of A by A^{adj}, so that

$$A^{adj}A = AA^{adj} = dI. \tag{2.1}$$

Then A^{adj} is also an integral matrix, and

$$y = A^{adj}b \tag{2.2}$$

is an integral vector. The solution of the system of linear equations

$$Ax = b \tag{2.3}$$

is then

$$x = y/d. \tag{2.4}$$

Before continuing, let us introduce some notation. Given an integer z, the barred variable, $\bar{z}_k$, refers to the residue of z (mod p_k); that is,

$$\bar{z}_k \equiv z \pmod{p_k}. \tag{2.5}$$

The unbarred variable, z_k, refers to the kth coefficient of the mixed-radix representation of z:

$$z = z_1 + p_1z_2 + \ldots + p_1p_2\cdots p_{m-1}z_m. \tag{2.6}$$

Given a sequence of distinct prime numbers $\{p_k: k = 1,2,\ldots,m\}$, a corresponding sequence of solutions $\{(\bar{y}_k,\bar{d}_k): k = 1,2,\ldots,m\}$ is computed such that

$$A\bar{y}_k \equiv \bar{d}_k b \pmod{p_k}, \tag{2.7}$$

where

$$\bar{d}_k \equiv d \pmod{p_k},$$

$$\bar{y}_k \equiv y \pmod{p_k}. \tag{2.8}$$

In section 4, we shall see that only single-precision computations are required to find the solutions $(\bar{y}_k,\bar{d}_k)$ in the Galois field $GF(p_k)$, except perhaps during the conversion of the system Ax = b to the system of congruences (mod p_k). Using the Chinese Remainder theorem, we can then construct the solution (y,d) of Ay = db, provided that the solutions $(\bar{y}_k,\bar{d}_k)$ of the system of congruences (mod p_k) have been found for a sufficient number of primes p_k. Since the solution (y,d) may have negative values, it is desirable to represent $GF(p_k)$ symmetrically about zero, namely by

$$GF(p_k) = \{(1-p_k)/2,\ldots,-1,0,1,\ldots,(p_k-1)/2\}.$$

Given the residues $\bar{z}_k$ $(|\bar{z}_k| \le (p_k-1)/2)$ for k = 1,2,...,m of an integer z, that is

$$\bar{z}_k \equiv z \pmod{p_k}, \tag{2.9}$$

z can be reconstructed using the Chinese Remainder theorem as follows. Suppose that

$$|z| \le (p_1p_2\cdots p_m-1)/2. \tag{2.10}$$

A sequence $\{s_k:|s_k| \le (p_1p_2\cdots p_k-1)/2\}$ is obtained which satisfies

$$\begin{aligned} z &\equiv s_1 \pmod{p_1} \\ z &\equiv s_2 \pmod{p_1p_2} \\ &\vdots \\ z &\equiv s_m \pmod{p_1p_2\cdots p_m}. \end{aligned} \tag{2.11}$$

But since the Chinese Remainder theorem guarantees the uniqueness of residues, the bound (2.10) implies that $z = s_m$.

Starting with $s_1 = z_1 = \bar{z}_1$, the s_k's are generated recursively for k = 2,3,...,m by

$$z_k = (p_1p_2\cdots p_{k-1})^{-1}(\bar{z}_k-s_{k-1}) \pmod{p_k}, \tag{2.12}$$

$$\begin{aligned} s_k &= s_{k-1} + p_1p_2\cdots p_{k-1}z_k \\ &= z_1 + p_1z_2 + \ldots + p_1p_2\cdots p_{k-1}z_k. \end{aligned} \tag{2.13}$$

The proof follows by induction. The initial step, $z \equiv s_1 \pmod{p_1}$ is obvious. Assuming that $z \equiv s_{k-1} \pmod{p_1p_2\cdots p_{k-1}}$, then

$$z = s_{k-1} + cp_1p_2\cdots p_{k-1} \tag{2.14}$$

for some integer c; and $z \equiv \bar{z}_k \pmod{p_k}$ implies that

$$c \equiv (p_1p_2\cdots p_{k-1})^{-1}(\bar{z}_k-s_{k-1}) \pmod{p_k}. \tag{2.15}$$

Let z_k be the right-hand side of (2.15), and let $s_k = s_{k-1} + p_1p_2\cdots p_{k-1}z_k$. Then

$$\begin{aligned} s_k &\equiv s_{k-1} \pmod{p_1p_2\cdots p_{k-1}} \\ s_k &\equiv \bar{z}_k \pmod{p_k}. \end{aligned} \tag{2.16}$$

That $z \equiv s_k \bmod p_1p_2\cdots p_k$ follows by the Chinese Remainder theorem.

Various methods, one of which is the recursion (2.12) - (2.13), of reconstructing a number from its residues are available. Detailed descriptions of these methods and their relative efficiencies are discussed by Lipson[13]. It is shown that the recursion (2.12) - (2.13) is the best method with respect to storage requirements and the total number of operation needed.

Takahasi and Ishibashi[18] have implemented the recursion for solving systems of linear equations. In their paper, the recursion (2.13) for s_k is accumulated as a multiple-precision sum. Multiple-precision computations are then also required for the calculation of z_k in (2.12).

If instead the z_i, $i = 1,2,\ldots,k-1$, are made available, the computation of z_k is reduced to a sequence of single-precision multiplications and additions in $GF(p_k)$. None of the multiple-precision numbers, s_k, need be computed except for the last one, s_m, which is the desired multiple-precision number z. Although this approach does not significantly reduce the total computation time, it permits the effective removal of a large proportion of multiple-precision calculations. Furthermore, the storage required for the z_k's is not restrictive if the algorithm is written for a compiler with dynamic storage allocation capabilities. While expanding the mixed-radix representation (2.11) for $z = s_m$, the storage required by the z_k's may simply be freed as we evaluate successive terms of the representation.

3. Terminating the Recursion

A crucial problem in the described algorithm is that the correct number of primes $p_1,p_2,\ldots,p_m$ that should be used is not known. If a sufficiently large number of primes is used, so that

$$|d|,\ |y| \le (p_1p_2\cdots p_m-1)/2 \qquad (3.1)$$

($|y|$ refers to the absolute value of each element of the vector y), a unique solution (y,d) is guaranteed by the Chinese Remainder theorem; but needless computations will be performed if the bound is too large.

The approach taken by Borosh and Fraenkel[3] in overcoming the problem is to continue computing terms, y_k and d_k, for the mixed-radix representation of y and d:

$$y = y_1 + p_1y_2 + \ldots + p_1p_2\cdots p_{k-1}y_k + \ldots \qquad (3.2)$$

$$d = d_1 + p_1d_2 + \ldots + p_1p_2\cdots p_{k-1}d_k + \ldots ,$$

until, for some m, $d_{m+1} = 0$ and y_{m+1} is the zero vector. The probability that successive terms y_k and d_k for $k = m+2, m+3,\ldots$ are also zero is then extremely high since the primes, p_k, are large numbers. However, in order to guarantee that

$$y = y_1 + p_1y_2 + \ldots + p_1p_2\cdots p_{m-1}y_m \qquad (3.3)$$

$$d = d_1 + p_1d_2 + \ldots + p_1p_2\cdots p_{m-1}d_m$$

is the required solution, a substitution check is made and more primes are used if $Ay \neq db$.

Since the solution (y,d) will normally be composed of large multiple-precision numbers, the substitution check is a very undesirable feature. The main result of this paper is the following theorem which yields a method for completely avoiding this multiple-precision check.

Theorem: Assume that the primes are ordered such that $p_1 < p_2 < p_3 < \ldots$. Suppose that $A = (a_{ij})$, an $n \times n$ matrix, is such that

$$\sum_{j=1}^{n} |a_{ij}| \le p_1p_2\cdots p_t, \qquad (3.4)$$

and $b = (b_i)$ is such that

$$|b_i| \le p_1p_2\cdots p_t, \qquad i = 1,2,\ldots,n. \qquad (3.5)$$

If the mixed-radix representation of (y,d) is

$$y = y_I + p_1p_2\cdots p_m\cdot 0 + \ldots + p_1p_2\cdots p_{m+t-1}\cdot 0 + p_1p_2\cdots p_{m+t}\cdot y_R \qquad (3.6)$$

$$d = d_I + p_1p_2\cdots p_m\cdot 0 + \ldots + p_1p_2\cdots p_{m+t-1}\cdot 0 + p_1p_2\cdots p_{m+t}\cdot d_R,$$

where

$$y_I = y_1 + p_1y_2 + \ldots + p_1p_2\cdots p_{m-1}y_m \qquad (3.7)$$

$$d_I = d_1 + p_1d_2 + \ldots + p_1p_2\cdots p_{m-1}y_m,$$

then

$$Ay_I = d_Ib. \qquad (3.8)$$

Proof: Suppose the contrary; that is $Ay_I - d_Ib \neq 0$. Then for some element of $Ay_I - db$ (say, $(Ay - db)_i$), there exists an integer $c \neq 0$ ($(Ay_R-d_Rb)_i = c$) such that

$$0 = (Ay-db)_i = (Ay_I-d_Ib)_i + c\,p_1p_2\cdots p_{m+t}. \qquad (3.9)$$

Thus,

$$|(Ay_I-d_Ib)_i| \ge p_1p_2\cdots p_{m+t}. \qquad (3.10)$$

But, using the bounds (3.4) and (3.5), and the fact that $|y_i| \le (p_i-1)/2$, $|d_i| \le (p_i-1)/2$, $i = 1,2,\ldots,m$,

$$(Ay_I-d_Ib)_i = (Ay_1-d_1b)_i + p_1(Ay_2-d_2b)_i + \ldots + p_1p_2\cdots p_{m-1}(Ay_m-d_mb)_i$$
$$< p_1p_2\cdots p_t(p_1 + p_1p_2 + \ldots + p_1p_2\cdots p_m). \qquad (3.11)$$

Whence, since $p_i < p_j$ for $i < j$,

$$(Ay_I - d_I b)_i < p_1 p_2 \cdots p_t p_{t+1}(p_2 + p_2 p_3 + \cdots + p_2 p_3 \cdots p_m)$$

$$< p_1 p_2 \cdots p_{t+2}(p_3 + p_3 p_4 + \cdots + p_3 p_4 \cdots p_m)$$

$$\vdots$$

$$< p_1 p_2 \cdots p_{m+t}. \qquad (3.12)$$

Similarly,

$$(Ay_I - d_I b)_i > - p_1 p_2 \cdots p_{m+t}, \qquad (3.13)$$

so that

$$|(Ay_I - d_I b)_i| < p_1 p_2 \cdots p_{m+t}. \qquad (3.14)$$

This contradiction completes the proof.

Corollary 1: If $d_I \neq 0$, then

$$x = y_I / d_I \qquad (3.15)$$

is the solution of $Ax = b$.

Corollary 2: If $d_I = 0$ and $y_I \neq 0$, then A is singular.

Note that the theorem does not insure that A is singular (although we are almost certain that it is) if both $d_I = 0$ and y_I is the zero vector. Singularity can be guaranteed only if we know that

$$|d| \leq (p_1 p_2 \cdots p_t - 1)/2. \qquad (3.16)$$

Note also that d_I may be such that $d \neq d_I \neq 0$ and yet (3.8) will be satisfied; so that, for certain systems of equations, we will have obtained the solution, x, without having computed the full determinant, d.

The theorem is especially interesting for the case t = 1, that is when A and b are composed of single-precision elements satisfying

$$\sum_{j=1}^{n} |a_{ij}| \leq p_1, \qquad |b_i| \leq p_1. \qquad (3.17)$$

In this case, only one occurrence of zero coefficients, $y_{m+1} = 0$ and $d_{m+1} = 0$, is sufficient to guarantee that (y_I, d_I) is the required solution. A substitution check is, therefore, certainly unnecessary.

For t > 1, the question remains as to whether, having encountered one occurrence of zero coefficients, it is more economical to make a substitution check, or to continue iterating the process until t successive occurrences of zero coefficients have been encountered. An indication is provided by comparing the number of single-precision multiplications (or the equivalent for multiple-precision computations) required by both methods. If the primes, p_k, are chosen so that they just fit into the accumulator, multiplication of two multiple-precision numbers, bounded by $p_1 p_2 \cdots p_{t_1}$ and $p_1 p_2 \cdots p_{t_2}$, respectively, will correspond approximately to $t_1 t_2$ single-precision multiplications. Furthermore, since the magnitude of d will normally be of the order

$$(p_1 p_2 \cdots p_t)^n \qquad (3.18)$$

an estimate for m is given by

$$m = nt. \qquad (3.19)$$

Using these assumptions, a very rough approximation to the number of single-precision multiplications required for the substitution check is

$$n^3 t^2. \qquad (3.20)$$

If we make the further assumption that one multiplication is GF(p) corresponds to three single-precision multiplications (one division is required for mod p conversion of an integer, and on most computers multiplications are about twice as fast as divisions), the total number of single-precision multiplications required to produce t-1 additional terms in the mixed-radix representation (3.3) of y and d will be of the order

$$(t-1)\, n^2 \quad (n+3t). \qquad (3.21)$$

The estimates (3.21) and (3.20) indicate that the check described in the theorem, rather the substitution check, will be superior except perhaps for very small n.

While this argument does show that the use of the theorem will save most of the time required for the substitution check, it should be noted that the overall reduction in computation time is not too spectacular. Extending the above arguments, we find that a rough estimate of the total number of single-precision multiplications required to compute the solution and to verify it by the substitution check is

$$n^3 t(n+6t),$$

as compared to

$$n^3 t(n+5t)$$

multiplications if we make use of the theorem. Thus, the expected reduction in the computation time required for the entire algorithm is approximately

$$t/(n+6t) \qquad (3.22)$$

of the total time. We must not, however, overlook the other advantages of the proposed check; namely, the check no longer requires multiple-precision computation, and no special code is necessary to implement it.

An obvious shortcoming of both the proposed method and the substitution check is that always more systems of congruences are solved than are necessary to determine the solution (y_I, d_I). The inequality (3.1)

suggests that we might instead seek bounds for d and y and then choose a sufficient number of primes $p_1, p_2, \ldots, p_m$ so that the inequality is satisfied. This approach has been taken by Takahasi and Ishibashi[18] and by Newman[15], who suggest using Hadamard's inequality to compute bounds for d and y.

The merits of this approach are apparent when A and b are such that Hadamard's inequality yields good bounds for y and d (for example, the bound for d will be attained for any A which can be reduced by row-scaling to an orthogonal matrix). Even for near-singular matrices, when d will be "small" relative to its bound computed by Hadamard's inequality, the number of primes selected by this approach will closely approximate the correct number (except for very special b). The calculation of these bounds, however, will require multiple-precision computations, including multiple-precision comparisons when searching for larger elements of a multiple-precision matrix A. Despite this fact, it may very well be that this approach of choosing the required number of primes will require less computation than the method proposed by the theorem.

In any case, from an argument similar to the one which led to (3.22), we see that the overall reduction in computation time using either method will be negligible (except for very special systems for which Hadamard's bounds for d and for y are bad). Thus, we suggest the use of the method proposed by the theorem because of its elegance and ease of implementation.

4. Adjoint Solution of Ax = b in GF(p)*

Given a system of linear equation Ax = b with integral coefficients, it is required to obtain a scalar, $\bar{d}$, and a vector, $\bar{y}$, in GF(p) such that

$$A\bar{y} \equiv \bar{d}b \pmod{p}, \tag{4.1}$$

where the determinant, d, of A satisfies

$$\bar{d} \equiv d \pmod{p}. \tag{4.2}$$

We begin by computing $\bar{A}$ and $\bar{b}$ with elements in GF(p) (using multiple-precision arithmetic if A and b contain multiple-precision elements), such that

$$\begin{aligned} \bar{A} &\equiv A \pmod{p} \\ \bar{b} &\equiv b \pmod{p}. \end{aligned} \tag{4.3}$$

The solution $(\bar{d}, \bar{y})$ of

$$\bar{A}\,\bar{y} \equiv \bar{d}\,\bar{b} \tag{4.4}$$

will then also be the solution of (4.1). Note that

$$\bar{y} \equiv \bar{A}^{adj}\bar{b} \pmod{p}. \tag{4.5}$$

To solve the system of congruences (4.4), one can use the usual Gaussian elimination method with partial pivoting, except that now all operations are performed in GF(p). Thus, the division, a/b, involves computing b^{-1} in GF(p) and then multiplying b^{-1} by a. Searching for a pivot row no longer involves searching for the largest pivot element; any non-zero element will suffice since round off errors do not enter into the computation. Elimination and back-substitution normally yield a vector $\bar{z}$ in GF(p) such that

$$\bar{A}\,\bar{z} \equiv \bar{b} \pmod{p}. \tag{4.6}$$

Then if $\bar{d} \neq 0$ ($\bar{d}$ is the product of the pivot elements),

$$\bar{y} = \bar{d}\,\bar{z} \pmod{p} \tag{4.7}$$

is the solution of (4.1).

If $\bar{d} = 0$, then d = 0 (that is, the coefficient matrix, A, is singular) or d is multiple of p, and $\bar{y}$ cannot be determined by (4.7). Newman[15], and Takahasi and Ishibashi[18] suggest terminating the search for a solution when $\bar{d} = 0$, inasmuch as A is then probably singular. Borosh and Fraenkel[5] suggest discarding this particular prime if it is known that $d \neq 0$ (e.g. $\bar{d} \neq 0$ in GF(p) for previous primes, p). It turns out, however, that by modifying the Gaussian elimination process, $\bar{y}$ can still be found.

The modification is the reduction of A to echelon form rather than upper-triangular form, during the forward Gauss process. If rank (A) = n (mod p), the modification does not alter the normal Gaussian elimination process; no zero pivot column (column currently being eliminated, with zero elements on and below the diagonal) will be encountered. If, however, a zero pivot column is encountered (the ith column, say), elimination of this column is omitted, and we now eliminate column j, j = i+1,i+2,... as if it were the j-1st column. In this case the nth row of the echelon form will be zero, so that rank $(\bar{A}) < n \pmod{p}$.

If no more zero columns are encountered, then rank $(\bar{A}) = n-1 \pmod{p}$. The echelon form of the system $\bar{A}\,\bar{z} \equiv \bar{b} \pmod{p}$ becomes

$$\bar{A}^{(n-2)}\bar{z} = \bar{b}^{(n-2)} \tag{4.8}$$

where

$$\bar{A}^{(n-2)} = \begin{bmatrix} a_{1,1}^{(0)} & \cdots & a_{1,i-1}^{(0)} & a_{1,i}^{(0)} & \cdots & a_{1,n}^{(0)} \\ & & \ddots & \ddots & & \\ & & & a_{i-1,i-1}^{(i-2)} & a_{i-1,i}^{(i-2)} & \\ & & & & \ddots & a_{n-1,n}^{(n-2)} \\ & & & & & 0 \end{bmatrix},$$

$$\bar{b}^{(n-2)} = [b_1^{(0)}, \ldots, b_{i-1}^{(i-2)}, \ldots, b_{n-1}^{(n-2)}, b_n^{(n-2)}]^T$$

* Although we focus our attention in this section on the finite field GF(p), the arguments which follow hold equally well in any field.

(see Forsythe and Moler[7] for notation). Applying (4.5) to the reduced system (4.8), we immediately obtain

$$\bar{y}_i = (-1)^{n-i} a_{1,1}^{(0)} \cdots a_{i-1,i-1}^{(i-2)} \cdot a_{i,i+1}^{(i-1)} \cdots a_{n-1,n}^{(n-2)} \cdot b_n^{(n-2)}; \qquad (4.9)$$

that is, $\bar{y}_i$ is the determinant of the matrix appearing in (4.8) with the ith column replaced by the right-hand side. Furthermore, $\bar{d} = 0$ implies that

$$A\bar{y} \equiv 0 \pmod{p}, \qquad (4.10)$$

so that the remaining elements of $\bar{y}$ can be obtained by back-substitution in (4.8) with zero right-hand side;

$$\bar{y}_j = \begin{cases} 0, & j = n, n-1, \ldots, i+1 \\ -[a_{j,j}^{(j-1)}]^{-1} \sum_{k=j+1}^{i} a_{j,k}^{(j-1)} \cdot \bar{y}_k \pmod{p}, & j = i-1, i-2, \ldots, 1. \end{cases} \qquad (4.11)$$

In the case that a second zero pivot is encountered, at least the last two rows of the echelon form of A will be zero. Then rank $(\bar{A}) < n-1$, rank $(\bar{A}|\bar{b}) < n \pmod{p}$; and by Cramer's rule, $\bar{y}$ is the zero vector.

Shapiro[17] describes a method of computing the adjoint of a singular matrix, A, which is comparable to the proposed method in that Cramer's rule is used to compute one row of the adjoint matrix. In his method, however, $\bar{A}$ is first reduced to upper-triangular form, which when rank $(\bar{A}) < n \pmod{p}$ will contain zero diagonal elements.

Since the occurrence of more than one zero diagonal element in an upper-triangular matrix does not necessarily indicate a matrix of rank less than n-1, Shapiro's method does not distinguish between the cases rank $(\bar{A}) = n-1$ and rank $(\bar{A}) < n-1$ during the forward elimination. The trivial case rank $(\bar{A}) < n-1$ is detected only during the backward elimination process when two zero rows of the transformed matrix $\bar{A}$ occur simultaneously.

The case rank $(\bar{A}) = n-1$ is more complex than the proposed method as well, inasmuch as further reduction on A will be necessary before Cramer's rule can be applied to determine a row of the adjoint. This, however, results in only a moderate increase in the total number of operations, since the inefficiency occurs during the backward elimination process.

5. Acknowledgement

The author is indebted to Professor J.D. Lipson of the University of Toronto for his direction during the numerous inspiring consultations.

6. Bibliography

1 E.H. Bareiss, "Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination", Math. Comp. 22 (1968), 565-578.

2 W.A. Blankinship, "A New Version of the Euclidean Algorithm", Amer. Math. Month. 70 (1963), 742-745.

3 W.A. Blankinship, "Algorithm 288: Solution of Simultaneous Linear Diophantine Equations", Comm. ACM 9 (July 1966), 514.

4 E. Bodewig, Matrix Calculus, North-Holland, 1959.

5 I. Borosh and A.S. Fraenkel, "Exact Solutions of Linear Equations with Rational Coefficients by Congruence Techniques", Math. Comp. 20 (1966), 107-112.

6 G.E. Collins, "Computing Multiplicative Inverses in GF(p)", Math. Comp. 23 (1969), 197-200.

7 G. Forsythe and C.B. Moler, Computer Solution of Linear Algebraic Equations, Prentice-Hall, 1967.

8 L. Fox, An Introduction to Numerical Linear Algebra, Clarendon Press, 1964.

9 J.A. Howell and R.T. Gregory, "An Algorithm for Solving Linear Algebraic Equations using Residue Arithmetic I", BIT 9 (1969), 200-224.

10 J.A. Howell and R.T. Gregory, "An Algorithm for Solving Linear Algebraic Equations using Residue Arithmetic II", BIT 9 (1969), 324-337.

11 J.A. Howell and R.T. Gregory, "Solving Linear Equations using Residue Arithmetic - Algorithm II", BIT 10 (1970), 23-37.

12 D.E. Knuth, The Art of Computer Programming Vol. 2: Seminumerical Algorithms, Addison-Wesley, 1969.

13 J.D. Lipson, "Interpolation and Chinese Remainder Algorithms", University of Toronto Computer Science Technical Report, 1970.

14 H.A. Luther and L.F. Guseman, Jr., "A Finite Sequentially Compact Process for the Adjoints of Matrices over Arbitrary Integral Domains", Comm. ACM 5 (1962), 447-448.

15 M. Newman, "Solving Equations Exactly", J. Res. Nat. Bureau Standards-B, 71B (1967), 171-179.

16 J.B. Rosser, "A Method of Computing Exact Inverses of Matrices with Integer Coefficients", J. Res. Nat. Bureau Standards, 49 (1952), 349-358.

17 G. Shapiro, "Gauss Elimination for Singular Matrices", Math. Comp. 17 (1963), 441-445.

18 H. Takahasi and Y. Ishibashi, "A New Method for 'Exact Calculation' by a Digital Computer", Information Processing in Japan, 1 (1961), 28-42.