The Exact Solution of Systems of Linear Equations with Polynomial Coefficients

MICHAEL T. MCCLELLAN

University of Maryland, College Park, Maryland

ABSTRACT. An algorithm is presented for computing exactly general solutions for systems of linear equations with integer or polynomial coefficients. The algorithm applies modular homomorphisms—reductions modulo a prime and evaluations—eventually applying a Gaussian elimination algorithm to systems with coefficients in $GF(p)$. Then, by applying interpolation and the Chinese Remainder Algorithm, a general solution is obtained if the system is found to be consistent. Also included is a modular algorithm for matrix multiplication, as required for substitution tests. The computing times of these modular algorithms are analyzed. The computing time bounds, which dominate these times, are obtained as functions of the size of the system and the numeric coefficient and degree sizes of the system coefficients. A comparison with similar bounds for the exact division algorithm reveals the clear superiority of the modular algorithm.

KEY WORDS AND PHRASES: polynomials, symbol manipulation, algebraic algorithms, linear algebra, linear equations, matrices, matrix inversion, matrix multiplication, null space, vector spaces, basis vectors, exact arithmetic, exact division elimination, modular arithmetic, Chinese Remainder Algorithm, computing time analysis, computational complexity.

CR CATEGORIES: 5.9

1. *Introduction*

1.1. SOME HISTORICAL COMMENTS ON THE PROBLEM. In recent years interest in the exact, or symbolic, computer solution of systems of linear equations has increased markedly and a corresponding effort has been directed to this problem. In this time remarkable progress has been made in the design and analysis of algorithms for solving systems of linear equations. Such systems are most often encountered with integer or rational number coefficients. Direct methods such as Gaussian or Gauss-Jordan elimination [26] employ rational operations and are consequently, in general, inefficient for exact computation. In particular, the computation of greatest common divisors (GCD's) and the phenomenon of integer coefficient growth restrict the applicability of these methods considerably. Moreover, the extension of these methods to the case of polynomial or rational function coefficients witnesses the related phenomenon of degree growth as well (see [6] for some experiences with this problem).

These phenomena—integer coefficient growth and degree growth—which have been alternately referred to by the phrase "intermediate expression swell," have been detailed and major advances in their control have been made for the very problem of polynomial GCD calculation (see [8] for a historical development). Thus GCD computations not

This paper is a revised version of one with the same title that appears in the Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, March 23–25, 1971, held by ACM SIGSAM, which was prepared before the symposium.

The research reported here was supported by the National Science Foundation (GJ-239) and by the National Aeronautics and Space Administration (NGL-21-002-008).

Author's address: Computer Science Center, University of Maryland, College Park, MD 20742.

Journal of the Association for Computing Machinery, Vol. 20, No. 4, October 1973, pp. 563–588.

previously tractable became so with the discovery of the subresultant polynomial remainder sequence (PRS) algorithm by Collins [9]. Then, after some initial work by Collins [10] and Brown [7], complete modular algorithms for polynomial GCD and resultant calculation were developed, which are far superior [8, 13].

A parallel development has taken place in the solution of systems of linear equations. Let $\mathfrak{I}$ be an arbitrary integral domain for which a division algorithm exists, and let A and B be matrices over $\mathfrak{I}$, where A is $m \times n$, nonzero, and of rank r, and B is $m \times q$. Then the matrix equation $AX = B$ represents q systems of linear equations, one for each column of B; but for simplicity we refer to this equation as a system of linear equations. We have mentioned above the inefficiency of exact elimination methods involving rational operations, i.e. operations in the quotient field $Q(\mathfrak{I})$, where $\mathfrak{I} = I$, the ring of integers, or $\mathfrak{I} = I[x_1, \cdots, x_s]$, the domain of integral polynomials in s variables. (Let $I[x_1, \cdots, x_s]$ denote I for $s = 0$.) The first step in reducing this inefficiency has been to consider only those methods which restrict $\mathfrak{I}$ to be $I[x_1, \cdots, x_s]$, $s \geq 0$, and perform the elimination using only arithmetic operations in $\mathfrak{I}$. That is, if $C = (A, B)$ is the augmented matrix of the linear system, then C is transformed only by elementary row operations in $\mathfrak{I}$. One method of this type was devised by Rosser [27] to control integer coefficient growth in inverting an integral matrix. In his method, the elimination involves only subtractions of multiples of rows from other rows, where multiplication by small integers is used almost exclusively; no divisions are performed.

Division may be employed in this class of methods if it is assured that the divisions are exact, producing quotients in $\mathfrak{I}$ (i.e. all elements of a row are multiples of the divisor). These methods, referred to as *exact division algorithms*, by applying predetermined exact divisions, restrict integer coefficient growth and, in the case $\mathfrak{I} = I[x_1, \cdots, x_s]$, $s \geq 1$, reduce degree growth as well. Brief descriptions of the basic computational method can be found in [3] and [16]; more mathematical treatments are given in [23], [1], and [21]. In [24] the author has reconsidered exact division algorithms and has found that, by performing pivot searches and row interchanges in a particular way, a canonical solution form is obtained whose elements are certain known subdeterminants of the augmented system matrix C. We will return to this algorithm in Section 1.2 to consider the structure of this canonical form.

Although no GCD's are computed during the elimination in exact division algorithms, integer coefficient growth and (in the polynomial case) degree growth are still considerable. In solving systems of linear equations by exact division, GCD calculations are necessary only in the final stage if the rational form of the solution components is desired. We note that, in general, the above discussion applies as well to the case of rational coefficients in $Q(\mathfrak{I})$ for $\mathfrak{I} = I[x_1, \cdots, x_s]$, $s \geq 0$, since the augmented matrix C can be transformed to a matrix over $\mathfrak{I}$ by premultiplying the rows of C by suitable elements of $\mathfrak{I}$.

We now give a definition of a general solution of a system of linear equations, $AX = B$, which reflects the approach of using only arithmetic in $\mathfrak{I}$. Let $d \in \mathfrak{I}$ and Y be an $n \times q$ matrix over $\mathfrak{I}$. The pair (d, Y) will be called a particular solution of the system if $A[(1/d)Y] = B$ or $AY = dB$. Moreover, if $r < n$, the null space of A is nontrivial. Hence if Z is any $n \times n-r$ matrix over $\mathfrak{I}$ which solves the homogeneous equation $AX = 0$ and whose columns are linearly independent, then the columns of Z form a basis for the null space of A. Such a matrix Z will be called a *null space basis* for A. Since every solution of the system $AX = B$ can be written as the sum of $(1/d)Y$ and some $n \times q$ matrix, each of whose columns is a linear combination of the columns of Z, any such triple (d, Y, Z) will be said to constitute a *general solution* of the system. When $r = n = q = m$ and B is the identity matrix, then (d, Y) can be considered to represent a matrix inverse for A. Such a representation of a matrix inverse is not unique, of course, as is true of a general solution (d, Y, Z) of $AX = B$. Note that in what follows, unless stated otherwise, all matrices, general solutions, etc., will be considered to be over an integral domain $\mathfrak{I}$ (which ordinarily is not a field).

The exact division algorithm is an analog of the reduced PRS GCD algorithm insofar as integer coefficient and degree growth are reduced by predetermined exact divisions. However, as in the case of that algorithm, the application of modular methods to the solution of linear equations and related problems has produced superior algorithms. One of the earliest reported applications is that of Takahasi and Ishibashi [28] for inverting integral matrices. Later applications to linear equations were done by Borosh and Fraenkel [4], Newman [25], and Howell and Gregory [18]. Only [4] treated the more general case of nonunique solutions (i.e. $1 \leq r \leq m \leq n$). All algorithms were restricted to systems with integer coefficients, employing reductions modulo a prime, the Chinese Remainder Theorem, and arithmetic in finite fields $GF(p)$ (see [20, Sec. 4.3.2]).

The extension of modular methods to multivariate polynomial calculations became possible with the introduction of evaluation and interpolation of polynomials over finite fields $GF(p)$, as sketched in [10] for polynomial GCD calculations. The impact of these general methods on polynomial GCD and resultant calculation, as developed by Brown and Collins, has already been cited. Starting with [10] and [4], the author was able to obtain a complete modular algorithm for solving systems of linear equations with multivariate polynomial coefficients. The general superiority of this algorithm over the exact division algorithm is no less dramatic than that for polynomial GCD and resultant calculation, as we shall observe in Section 5.

1.2. A Canonical Solution Form. The modular algorithm computes a general solution of the form defined in this section for integral domains. In fact, the modular algorithm most likely computes the same general solution as is computed by the particular variant of the exact division algorithm mentioned in Section 1.1. We now consider in more detail general solutions over an integral domain $\mathfrak{d}$ and their relation to exact division algorithms. Our intent is to completely specify a canonical form of a matrix and, as a consequence, a canonical general solution form of a system of linear equations. The discussion to follow is analogous to that for matrices over a field, as in [2]. A more detailed presentation, including proofs, may be found in [24].

An $m \times n$ nonzero matrix B is a *row echelon (RE) matrix* (is *in row echelon form*) if there is a sequence of integers $1 \leq k_1 < k_2 < \cdots < k_s \leq n$ for some $s : 1 \leq s \leq m$ such that $B(i, k_i) \neq 0$ and $B(i, j) = 0$, $1 \leq j < k_i$, and if $s < m$, rows $s + 1, \cdots, m$ are zero. If, in addition, the only nonzero element of column k_i is $B(i, k_i)$, then B is called a *reduced row echelon (RRE) matrix*. Thus an upper triangular matrix is an RE matrix and a diagonal matrix is an RRE matrix. The sequence $K = (k_1, k_2, \cdots, k_s)$ will be called the *row echelon (RE) sequence* of B. The elements $B(i, k_i)$, $1 \leq i \leq r$, will be said to constitute the *diagonal* of an RRE matrix B.

Suppose a matrix B is row equivalent to a matrix A. Then if B is an RE matrix, B is a *row echelon (RE) form* for A, and if B is an RRE matrix, B is a *reduced row echelon (RRE) form* for A. In this context, the RE sequence of B is considered an *RE sequence* of A. Trivially, a zero matrix is an RE and an RRE matrix with RE sequence $K = \varnothing$, the *null sequence* (i.e. the case $s = 0$).

Let $\mathcal{J}$ denote the set of all sequences $(j_1, \cdots, j_s)$ of positive integers in ascending order, including $\varnothing$. Clearly, in the context of this paper, the set $\mathcal{J}$ is simply the set of all RE sequences J. Next let $\mathcal{P}_m$ denote the set of all permutations on the first m positive integers and let $\mathcal{P} = \bigcup_{m=1}^{\infty} \mathcal{P}_m$. Note that each $\pi \in \mathcal{P}_m$ can be represented uniquely as a sequence $(\pi(1), \pi(2), \cdots, \pi(m))$. Both $\mathcal{J}$ and $\mathcal{P}$ are subsets of the set Ω of all finite sequences of positive integers, which includes $\varnothing$. Hence $\mathcal{J}$ and $\mathcal{P}$ are ordered by the *lexicographical ordering* $(>)$ of Ω, which is defined as follows. If $K \neq \varnothing$, then $K > \varnothing$. Otherwise, for two nonnull elements $K = (k_1, \cdots, k_s)$ and $H = (h_1, \cdots, h_t)$ of Ω, we have $K > H$ if and only if either (1) there exists a least integer $i : 1 \leq i \leq \min(s, t)$ such that $k_i \neq h_i$ and $k_i > h_i$; or (2) $s > t$ and $k_i = h_i$, $1 \leq i \leq t$. Ω is linearly ordered by $>$, and so every finite nonempty subset of Ω has a minimal element. This property is applied below to certain subsets of $\mathcal{P}$. The extension of $>$ to a linear ordering of $\mathcal{J} \times \mathcal{P}$ is done

in the natural way; i.e. for any two elements (K, H) and (J, I) of $\mathcal{J} \times \mathcal{P}$, we have $(K, H) > (J, I)$ whenever $K > J$ or $K = J$ and $H > I$.

Some helpful notation is introduced here. Let A be an $m \times n$ matrix and let $i_1, \cdots, i_s$ and $j_1, \cdots, j_t$ be sequences of integers such that $1 \leq i_k \leq m$ and $1 \leq j_h \leq n$ for $1 \leq k \leq s$ and $1 \leq h \leq t$. Then the matrix consisting of the elements of A common to rows $i_1, \cdots, i_s$ and columns $j_1, \cdots, j_t$ in that order is denoted by $A\begin{bmatrix} i_1, \cdots, i_s \\ j_1, \cdots, j_t \end{bmatrix}$. If $s = t$, its determinant is denoted by $A\begin{pmatrix} i_1, \cdots, i_s \\ j_1, \cdots, j_s \end{pmatrix}$. This notation is a generalization of that used in [17].

We now define, corresponding to any matrix A, unique elements $J_A \in \mathcal{J}$ and $I_A \in \mathcal{P}$. Suppose A is $m \times n$, nonzero, and of rank r, and let M_j be the matrix consisting of the first j columns of A, for $1 \leq j \leq n$. Then define $J_A = (j_1, \cdots, j_r)$, where j_h is the least integer j such that rank $(M_j) = h$. Clearly, $j_1 < j_2 < \cdots < j_r$, and so $J_A \in \mathcal{J}$. If A is a zero matrix, we take $J_A = \varnothing$. It is easily shown that for every matrix A, $K = J_A$ is the *unique* RE sequence for A.

Assuming A is nonzero, a sequence of distinct integers $h_1, \cdots, h_r$, where $1 \leq h_t \leq m$, $1 \leq t \leq r$, can be found such that

$$A\begin{pmatrix} h_1, \cdots, h_s \\ j_1, \cdots, j_s \end{pmatrix} \neq 0$$

for $1 \leq s \leq r$. If we let $h_{r+1}, \cdots, h_m$ be the remaining integers, in any order, such that $\{h_1, \cdots, h_m\} = \{1, \cdots, m\}$, then $H = (h_1, \cdots, h_m)$ is in $\mathcal{P}_m$. Define $\mathcal{P}_A$ as the set of all such sequences H:

$$\mathcal{P}_A = \left\{(h_1, \cdots, h_m) \in \mathcal{P}_m : A\begin{pmatrix} h_1, \cdots, h_s \\ j_1, \cdots, j_s \end{pmatrix} \neq 0, \quad 1 \leq s \leq r\right\}.$$

Then $\mathcal{P}_A$ has a minimal element $I_A = (i_1, \cdots, i_m)$. We note that I_A is minimal only if, for $s = 1, 2, \cdots, r$, i_s is the least integer $i : i \neq i_t$, $1 \leq t < s$, such that

$$A\begin{pmatrix} i_1, \cdots, i_{s-1}, i \\ j_1, \cdots\cdots, j_s \end{pmatrix} \neq 0.$$

Moreover, if $r < m$, I_A is minimal only if the remaining components satisfy $1 \leq i_{r+1} < i_{r+2} < \cdots < i_m \leq m$. If A is a zero matrix, we let $\mathcal{P}_A = \{(1, 2, \cdots, m)\}$, whereupon $I_A = (1, 2, \cdots, m)$.

For any $m \times n$ nonzero matrix C, consider the following two $m \times n$ matrices, $\hat{C}_H$ and $\bar{C}_H$, defined using C, $J_C = (j_1 \cdots, j_r)$, and any permutation $H = (h_1, \cdots, h_m) \in \mathcal{P}_C$:

$$\hat{C}_H(k, j) = \begin{cases} C\begin{pmatrix} h_1, \cdots\cdots, h_k \\ j_1, \cdots, j_{k-1}, j \end{pmatrix} & \text{for } 1 \leq k \leq r, \\ 0 & \text{otherwise;} \end{cases} \tag{1}$$

$$\bar{C}_H(k, j) = \begin{cases} C\begin{pmatrix} h_1, \cdots\cdots\cdots\cdots, h_r \\ j_1, \cdots, j_{k-1}, j, j_{k+1}, \cdots, j_r \end{pmatrix} & \text{for } 1 \leq k \leq r, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

If C is a zero matrix, we take $\hat{C}_H = \bar{C}_H = C$. It is not difficult to show, using the definition of J_C, that $\hat{C}_H$ is an RE matrix and $\bar{C}_H$ is an RRE matrix, each having RE sequence J_C. We shall denote $\hat{C}_H$ and $\bar{C}_H$ simply by $\hat{C}$ and $\bar{C}$ for $H = I_C$. It can further be shown that a particular version of the exact division algorithm, which we denote by Γ, when applied to a matrix C, computes $\hat{C}$ during the forward elimination and computes $\bar{C}$ as a result of the complete diagonalization. The mapping Γ requires that the pivot searches and row interchanges be performed in a specific way (see the discussion of Algorithm CRRE in Section 3.2). The proof of the above assertions is straightforward but tedious and the reader is referred to [24] for the details.

Thus we can write $\bar{C} = \Gamma(C)$. Analysis of the mapping Γ shows that $\bar{C}$ is row equivalent (in $\mathcal{G}$) to C and so $\bar{C}$ is an RRE form for C. It is in fact true that, for each $H \in \mathcal{P}_C$, $\bar{C}_H$ is an RRE form for C because, by applying row interchanges to C, the matrix C' can be obtained, where $C'(k, j) = C(h_k, j)$. Then the exact division algorithm applied to C' computes $\bar{C}' = \Gamma(C')$, where $J_{C'} = J_C$ and $I_{C'} = (1, 2, \cdots, m)$. Thus for each $H \in \mathcal{P}_{C'}$, $\bar{C}_H$ will be referred to as a *determinantal RRE (DRRE) form* for C. The matrix $\bar{C}$ will be called the *canonical DRRE form* for C in the sense that it is uniquely defined and effectively computable.

For $H = (h_1, \cdots, h_m) \in \mathcal{P}_C$, define

$$\delta_H(C) = C\begin{pmatrix} h_1, & \cdots, & h_r \\ j_1, & \cdots, & j_r \end{pmatrix} \tag{3}$$

and for C zero, define $\delta_H(C) = 0$. Then the diagonal elements of $\bar{C}_H$ all equal $\delta_H(C)$ as their common value. For $H = I_C$, we shall denote $\delta_H(C)$ simply by $\delta(C)$.

We turn now to the construction of general solutions for systems of linear equations, the basic construction being that for obtaining a null space basis for a matrix from an RRE form. Suppose E is an $m \times n$ nonzero RRE matrix with RE sequence $J_E = (j_1, \cdots, j_r)$, $r < n$, and common diagonal value d. Let $1 \leq k_1 < k_2 < \cdots < k_{n-r} \leq n$ be the sequence of integers which complements $j_1, j_2, \cdots, j_r$ with respect to $1, 2, \cdots, n$. Let Z be the $n \times n - r$ matrix defined by

$$Z(j_i, j) = E(i, k_j), \qquad Z(k_j, u) = \begin{cases} -d, & u = j \\ 0, & u \neq j. \end{cases} \tag{4}$$

It is not hard to show that $EZ = 0$ and that the columns of Z are linearly independent. Moreover, if E is an RRE form for a matrix C, then $C = UE$ for some $m \times m$ matrix U since C and E are row equivalent, and so $CZ = (UE)Z = U(EZ) = 0$. This gives the following theorem.

THEOREM 1. *Let C be an $m \times n$ nonzero matrix with $J_C = (j_1, \cdots, j_r)$, $r < n$, and let E be an RRE form for C with common diagonal value d. If Z is the $n \times n-r$ matrix defined by* (4), *then Z is a null space basis for C.*

If C is the augmented matrix of a consistent system of linear equations, then this theorem provides a type of general solution for the system.

THEOREM 2. *Let $C = (A, B)$ be the augmented matrix of a consistent linear system $AX = B$, where A is $m \times n$ and nonzero and B is $m \times q$. Let E be an $m \times n'$ RRE form for C, $n' = n + q$, with common diagonal value d and rank r. Let Z' be the $n' \times n'-r$ matrix defined from E and J_C according to* (4). *Then if Z consists of the first n rows and first $n - r$ columns of Z' and if Y consists of the first n rows and last q columns of Z', then (d, Y, Z) is a general solution of $AX = B$.*

PROOF. Partition Z' as $\begin{bmatrix} Z & Y \\ U & V \end{bmatrix}$, where Z and Y are defined in the theorem. Then $CZ' = (AZ + BU, AY + BV) = (AZ, AY - dB) = 0$, applying Theorem 1. The columns of Z are, of course, linearly independent. □

If in Theorem 2 the matrix E is a DRRE form $\bar{C}_H$ for C, then a special class of general linear equations solutions is obtained. Each member of this class $(\delta_H(C), Y, Z)$ will be referred to as a *determinantal general linear equations solution* (DLES) or simply *determinantal general solution*. The particular DLES corresponding to $H = I_C$ (i.e. that constructed from the canonical DRRE form $\bar{C}$) will be called the *canonical DLES*: $(\delta(C), Y, Z)$.

1.3. AN OUTLINE OF THE REMAINING SECTIONS. In this Introduction the problem of exactly computing a solution to a system of linear equations has been described, existing methods surveyed, and a basic theoretical framework established. It is against this background that the modular homomorphism algorithm, which is presented and analyzed in the following sections, has been developed. In Section 2, the basic results are obtained which concern the construction of general solutions by employing homomorphisms. Some

results deal with homomorphisms in general and others with mod-p and evaluation homomorphisms in particular. In Section 3, the modular homomorphism algorithm for computing a general linear equations solution is presented. The computing time analysis of this algorithm is done in Section 4. In Section 5, the new algorithm is compared with the exact division algorithm and, also, the time for reduction of a solution to rational form is taken into account.

2. *Constructing General Solutions by Using Homomorphisms*

In this section we present the basic concepts and methods employed in modular homomorphism algorithms in general and in computing DRRE forms of matrices (and hence DLES's) in particular. These methods are described in iterative form, applying at each step either reduction modulo a prime and the Chinese Remainder Algorithm for integers and integral polynomials, or evaluation and interpolation for polynomials with numeric coefficients in a finite field with a prime number of elements. Not all homomorphisms can be used in computing DRRE forms; so criteria are developed for deciding when a given homomorphism can be employed. Related to this, upper bounds on the number of unusable homomorphisms are obtained. Finally, with regard to computing both DRRE forms and general solutions, substitution tests and termination criteria for use in the modular homomorphism algorithm are described.

Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be integral domains and let θ be a homomorphism of $\mathcal{I}_1$ to $\mathcal{I}_2$. Let $\bar{\theta}$ be the mapping of matrices over $\mathcal{I}_1$ to matrices over $\mathcal{I}_2$ induced by θ; that is, $A^* = \bar{\theta}(A)$, for A a matrix over $\mathcal{I}_1$, if each element of A^* satisfies $A^*(i, j) = \theta(A(i, j))$. If for matrices R, S, T, U over $\mathcal{I}_1$, $R + S$ and $T \cdot U$ are defined, then $\bar{\theta}$ behaves like a homomorphism: $\bar{\theta}(R + S) = \bar{\theta}(R) + \bar{\theta}(S)$ and $\bar{\theta}(TU) = \bar{\theta}(T)\bar{\theta}(U)$. The convention we use is as follows. We denote $\bar{\theta}$ simply by θ and refer to each such mapping θ as a homomorphism, with the proviso that θ is applied only to matrix sums and products which are defined.

2.1. Mod-p Homomorphisms and the Chinese Remainder Algorithm. For any prime $p \in I$, let I_p be the finite field of integers modulo p and let φ_p be the unique homomorphism of I onto I_p such that $\varphi_p(a) = a$ for $0 \leq a < p$. This homomorphism induces a homomorphism of $I[x_1, \cdots, x_s]$ onto $I_p[x_1, \cdots, x_s]$, $s \geq 1$, in a natural way (i.e. such that the numeric coefficients of an integral polynomial are mapped by φ_p into I_p). We denote each such induced homomorphism also by φ_p and refer to each φ_p as a mod-p homomorphism. Similarly, the mappings induced by φ_p on matrices over $I[x_1, \cdots, x_s]$, $s \geq 0$, will be denoted by φ_p.

Suppose $p_1, \cdots, p_t$ are distinct odd primes and let $a_i \in I_{p_i}$, $1 \leq i \leq t$. By the Chinese Remainder Theorem, there is a unique integer a such that $a_i = \varphi_{p_i}(a)$ for $1 \leq i \leq t$, and $-p_1 \cdots p_t/2 < a < p_1 \cdots p_t/2$. One method for computing a from $a_1, \cdots, a_t$ and $p_1, \cdots, p_t$ is an iterative version of the Chinese Remainder Algorithm due to Garner (see [20, pp. 253–254], where the p_i are assumed only to be relatively prime). The steps are as follows:

Step 1. Set $b_1 = a_1$; if $a_1 \geq p_1/2$, replace b_1 by $b_1 - p_1$.

Step 2. For $i = 2, 3, \cdots, t$, perform the following computations:
(a) compute $c_i = [\varphi_{p_i}(p_1 \cdots p_{i-1})]^{-1} \cdot (a_i - \varphi_p(b_{i-1}))$ in I_{p_i};
(b) if $c_i \geq p_i/2$, replace c_i by $c_i - p_i$;
(c) compute $b_i = b_{i-1} + c_i p_1 \cdots p_{i-1}$.

Step 3. Finally, set $a = b_t$.

This algorithm can be incorporated directly into what will be called a Chinese Remainder Algorithm for polynomials. That is, if $A_i \in I_{p_i}[x_1, \cdots, x_s]$, $1 \leq i \leq t$, $s \geq 1$, then the unique polynomial $A \in I[x_1, \cdots, x_s]$, such that $\varphi_{p_i}(A) = A_i$ and each integer coefficient of A is less than $p_1 \cdots p_t/2$ in magnitude, can be constructed by applying the above algorithm to corresponding integer coefficients of the A_i. Hence suppose, for each

$1 \leq i \leq t$, that $M^{(i)}$ is an $m \times n$ matrix over $I_{p_i}[x_1, \cdots, x_s]$, $s \geq 0$. Let M be the unique matrix over $I[x_1, \cdots, x_s]$ such that $\varphi_{p_i}(M) = M^{(i)}$ and each element of M has integer ceofficients less than $p_1 \cdots p_t/2$ in magnitude. Then M can be computed by applying the Chinese Remainder Algorithm for polynomials to corresponding elements of the $M^{(i)}$.

Suppose c is a bound on the magnitudes of the integer coefficients of the elements of a matrix M over $I[x_1, \cdots, x_s]$, $s \geq 0$. Then if $p_1, \cdots, p_t$ are odd primes such that $c < p_1 \cdots p_t/2$, M can be constructed from the images $\varphi_{p_1}(M), \cdots, \varphi_{p_t}(M)$ by the Chinese Remainder Algorithm. Such a bound can be computed by using the mapping $|\ |_1$, defined on multivariate integral polynomials as follows. For $a \in I$, let $|a|_1 = |a|$, and for $A = \sum_{i=0}^{n} a_i x^i \in I[x]$, let $|A|_1 = \sum_{i=0}^{n} |a_i|_1$. By induction, for $A = \sum_{i=0}^{n} A_i x_s^{\,i} \in I[x_1, \cdots, x_s]$, $s \geq 2$, we define $|A|_1 = \sum_{i=0}^{n} |A_i|_1$. We let $|0|_1 = 0$. It can be easily shown that $|A + B|_1 \leq |A|_1 + |B|_1$ and that $|A \cdot B|_1 \leq |A|_1 \cdot |B|_1$. Hence, we may refer to $|\ |_1$ as a *norm*. Moreover, each integer coefficient e of $A \in I[x_1, \cdots, x_s]$ satisfies $e \leq |A|_1$. So the bound c above might be computed as the maximum of the $|M(i,j)|_1$.

Suppose $M = \bar{C}_H$, a DRRE form for an $m \times n$ nonzero matrix C over $I[x_1, \cdots, x_s]$ with $J_C = (j_1, \cdots, j_r)$ and $H = (h_1, \cdots, h_m) \in \mathcal{O}_C$. The only possibly nonzero elements of $\bar{C}_H$ are

$$\bar{C}_H(i,j) = C\begin{pmatrix} h_1, & \cdots\cdots\cdots\cdots\cdots, & h_r \\ j_1, \cdots, & j_{i-1}, & j, & j_{i+1}, \cdots, & j_r \end{pmatrix}$$

for $1 \leq i \leq r$ and $j_i \leq j \leq n$ and $j \neq j_u$, $i < u \leq r$. Hence if c bounds the norms of these elements, then we will know that $\bar{C}_H$ has been computed by the Chinese Remainder Algorithm whenever the primes used satisfy $p_1 \cdots p_t/2 > c$. One possible way to compute c is to use the following bound on the norm of a determinant.

THEOREM 3. *For every $m \times m$ matrix B over $I[x_1, \cdots, x_s]$, $s \geq 0$,*

$$|\det(B)|_1 \leq m! \cdot \pi_{j=1}^{m} \max_{1 \leq i \leq m} |B(i,j)|_1 .$$

PROOF. Write $\det(B)$ formally as the sum of $m!$ terms and apply the properties of norm $|\ |_1$ given above. □

Such a bound c is not required in the modular algorithm of Section 3. However, this theorem is needed in the computing time analysis of Section 4.

2.2. EVALUATION HOMOMORPHISMS AND INTERPOLATION. By applying mod-p homomorphisms above, we obtained matrices over $I_p[x_1, \cdots, x_s]$, $s \geq 0$. In Section 3 we will have to compute the canonical DRRE forms of such matrices and to do so we will require evaluations and interpolations of polynomials in $I_p[x_1, \cdots, x_s]$, $s \geq 1$.

For each $a \in I_p[x_1, \cdots, x_{s-1}]$, let Ψ_a denote the *evaluation homomorphism* of $I_p[x_1, \cdots, x_s]$ onto $I_p[x_1, \cdots, x_{s-1}]$ such that $\Psi_a(A) = A(x_1, \cdots, x_{s-1}, a)$ for all $A \in I[x_1, \cdots, x_s]$. Let a_i, b_i be elements of $I_p[x_1, \cdots, x_{s-1}]$ for $0 \leq i \leq q$, where the a_i are distinct. Suppose A is a polynomial in $I_p[x_1, \cdots, x_s]$ of degree at most q in x_s such that $\Psi_{a_i}(A) = b_i$, $0 \leq i \leq q$. Then A can be constructed by the following procedure, which is called the *Incremental Interpolation Algorithm*:

Step 1. Set $A_{-1}(x_1, \cdots, x_s) = 0$ and $E_{-1}(x_s) = 1$.

Step 2. For $i = 0, 1, \cdots, q$, perform the following computations:
(a) compute $E_i(x_s) = (x_s - a_i) \cdot E_{i-1}(x_s)$;
(b) compute $C_i(x_1, \cdots, x_{s-1}) = (b_i - A_{i-1}(x_1, \cdots, x_{s-1}, a_i))/E_{i-1}(a_i)$;
(c) compute $A_i(x_1, \cdots, x_s) = C_i(x_1, \cdots, x_{s-1}) \cdot E_{i-1}(x_s) + A_{i-1}(x_1, \cdots, x_s)$.

Step 3. Finally, set $A = A_q(x_1, \cdots, x_s)$.

We note that if for each i, a_i is chosen from I_p, then C_i is in $I_p[x_1, \cdots, x_{s-1}]$ and hence A_i is in $I[x_1, \cdots, x_s]$. This will be the case in the modular algorithm. We note that all Ψ_a and all φ_p are, in fact, classed together as modular homomorphisms and that the Incremental Interpolation Algorithm is, in fact, an example of a Chinese Remainder Algo-

rithm (see [20, pp. 394–395], [8], and especially [22] for an in-depth study of these concepts).

For each $a \in I_p[x_1, \cdots, x_{s-1}]$, an evaluation mapping of the matrices over $I_p[x_1, \cdots, x_s]$ to the matrices over $I_p[x_1, \cdots, x_{s-1}]$ is induced by Ψ_a, which will also be denoted Ψ_a. For $0 \leq i \leq q$, let $M^{(i)}$ be an $m \times n$ matrix over $I_p[x_1, \cdots, x_{s-1}]$ and let $a_i \in I_p[x_1, \cdots, x_{s-1}]$. Then the unique matrix M over $I_p[x_1, \cdots, x_s]$, each of whose elements has degree at most q and such that $\Psi_{a_i}(M) = M^{(i)}$, $0 \leq i \leq q$, can be constructed by applying the Incremental Interpolation Algorithm to corresponding elements of the $M^{(i)}$. Similar to the situation for mod-p homomorphisms, let $M = \bar{C}_H$, a DRRE form for a nonzero $m \times n$ matrix C over $I_p[x_1, \cdots, x_s]$, and let g be a bound on the degrees in x_s of the possibly nonzero elements of $\bar{C}_H$:

$$\bar{C}_H(i, j) = C\begin{pmatrix} h_1, \cdots\cdots\cdots\cdots\cdots, h_r \\ j_1, \cdots, j_{i-1}, j, j_{i+1}, \cdots, j_r \end{pmatrix}$$

for $1 \leq i \leq r$ and $j_i \leq j \leq n$ and $j \neq j_u$, $i < u \leq r$. Then if $a_0, a_1, \cdots, a_g$ are any distinct elements of $I[x_1, \cdots, x_{s-1}]$, M can be constructed from the matrices $\Psi_{a_i}(\bar{C}_H)$, $0 \leq i \leq g$, by incremental interpolation.

The following results give such a bound. The first, giving a bound on the degree of a determinant, is similar to Theorem 3 and is proven in a similar manner. Note that $\mathcal{I}[x_s]$ is isomorphic to $I_p[x_1, \cdots, x_s]$ for $\mathcal{I} = I_p[x_1, \cdots, x_{s-1}]$.

THEOREM 4. *For every $m \times m$ matrix B over $\mathcal{I}[x]$,*

$$\deg(\det(B)) \leq \sum_{j=1}^{m} \max_{1 \leq i \leq m} \deg(B(i, j)).$$

We next apply this theorem in obtaining the following bound on the degree of a determinant, which will give a better bound on the degrees of the elements of $\bar{C}_H$ than can be obtained using Theorem 4 directly.

THEOREM 5. *Let B be an $m \times m$ matrix over $\mathcal{I}[x]$. Then for any integer i: $1 \leq i \leq m$,*

$$\deg(\det(B)) \leq \max_{1 \leq k \leq m} \deg(B(i, k)) + \sum_{u=1}^{m} e_u - \min_{1 \leq u \leq m} e_u$$

where $e_u = \max_{1 \leq t \leq m, t \neq i} \deg(B(t, u))$, $1 \leq u \leq m$.

PROOF. By the Laplace expansion theorem, $\det(B) = \sum_{k=1}^{m} (-1)^{i+k} B(i, k) \cdot C_{i,k}$, where $C_{i,k}$ is the cofactor of $B(i,k)$, and so $\deg(\det(B)) \leq \max_{1 \leq k \leq m} \deg(B(i,k)) + \max_{1 \leq k \leq m} \deg(C_{i,k})$. Upon applying Theorem 4 to each $C_{i,k}$, the result follows easily. □

A bound b on the degrees of the elements of any DRRE form $\bar{C}_H$ is now obtained (refer to b as a DRRE degree bound).

THEOREM 6. *Let C be an $m \times n$ nonzero matrix with $J_C = (j_1, \cdots, j_r)$. Define $f = \max_{j \neq j_t} \max_{1 \leq i \leq m} \deg(C(i, j))$, $e_u = \max_{1 \leq i \leq m} \deg(C(i, j_u))$, for $1 \leq u \leq r$, and $e = \sum_{u=1}^{r} e_u$ and $e_0 = \min e_u$. Let $g = f - e_0$. Define $b = e$ if $g \leq 0$, and $b = e + g$ if $g > 0$. Then for any $H \in \mathcal{P}_C$, b bounds the degrees of the elements of $\bar{C}_H$.*

PROOF. Let $H = (h_1, \cdots, h_m)$ and let $f' = \max_{j \neq j_i} \max_{1 \leq t \leq r} \deg(C(h_t, j))$, $e_u' = \max_{1 \leq t \leq r} \deg(C(h_t, j_u))$ for $1 \leq u \leq r$, and $e' = \sum_{u=1}^{r} e_u'$ and $e_0' = \min e_u'$. By Theorem 4, each diagonal element of $\bar{C}_H$ has degree $\deg(\delta_H(C)) \leq e'$. For any nondiagonal element $\bar{C}_H(i, j)$, $j \neq j_i$, we have $\deg(\bar{C}_H(i, j)) \leq e' - e_0' + f'$ by applying Theorem 5. Finally, $e' \leq e \leq b$ and also $e' - e_0' + f' \leq e - e_0 + f \leq b$, since $e' - e_0' \leq e - e_0$ and $f' \leq f$. □

2.3. ACCEPTING AND REJECTING HOMOMORPHISMS. Returning to the general framework with which we began this section, let θ be a homomorphism of an integral domain $\mathcal{I}$ to an integral domain $\mathcal{I}^*$ and let $\bar{\theta}$ be the mapping induced by θ, which we also denote by θ. In what follows, θ will always be so defined. Recall from Section 1.2 the exact division mapping Γ. If $\theta\Gamma(A) = \Gamma\theta(A)$ for A a matrix over $\mathcal{I}$, we say that θ is a *commuting* homomorphism for A; otherwise, θ is called a *noncommuting* homomorphism for A. Suppose θ

is commuting for A (i.e., $\theta(\bar{A}) = \overline{\theta(A)}$). If θ is a mod-p homomorphism φ_p and A is over $I[x_1, \cdots, x_s]$, $s \geq 0$, then $\overline{\varphi_p(A)}$ can be used to construct $\bar{A}$ by the Chinese Remainder Algorithm. Also, if θ is an evaluation homomorphism Ψ_a and if A is over $I_p[x_1, \cdots, x_s]$, $s \geq 1$, then $\overline{\Psi_a(A)}$ can be used to construct $\bar{A}$ by the Incremental Interpolation Algorithm. Thus we need a sufficient condition for θ to be a commuting homomorphism for a matrix A. Such a criterion is given by the following theorem.

THEOREM 7. *Let A be an $m \times n$ matrix over $\mathcal{I}$ and let $A^* = \theta(A)$. Then $\overline{(A^*)} = (\bar{A})^*$ whenever $(J_A, I_A) = (J_{A^*}, I_{A^*})$.*

PROOF. Let $J_A = (j_1, \cdots, j_r)$ and $I_A = (i_1, \cdots, i_m)$ for A nonzero. The proof follows directly if we observe that, for $1 \leq i \leq r$ and $1 \leq j \leq n$,

$$\theta\left(A\begin{pmatrix} i_1, & \cdots & , i_r \\ j_1, & \cdots, j_{i-1}, j, j_{i+1}, & \cdots, j_r \end{pmatrix}\right) = A^*\begin{pmatrix} i_1, & \cdots & , i_r \\ j_1, & \cdots, j_{i-1}, j, j_{i+1}, & \cdots, j_r \end{pmatrix}. \quad \square$$

θ will be called an *accepted* homomorphism for A if $(J_A, I_A) = (J_{A^*}, I_{A^*})$ and a *rejected* homomorphism for A if $(J_A, I_A) \neq (J_{A^*}, I_{A^*})$. There are commuting homomorphisms θ which are also rejected homomorphisms. For example, let A be the matrix $\begin{bmatrix} t & 1 & 0 \\ 0 & t & 1 \\ 1 & 0 & t \end{bmatrix}$, where t is a nonzero element such that $\theta(t) = 0$ and $t^3 + 1 \neq 0$ (e.g. $\theta = \varphi_p$ and $t = p$). However, every noncommuting homomorphism is a rejected homomorphism. The application of Theorem 7 thus amounts to retaining only those commuting homomorphisms which are also accepted homomorphisms. This will be the criterion usually in effect in the algorithm presented in Section 3 for the cases $\mathcal{I} = I_p[x_1, \cdots, x_s]$, $s \geq 1$, and $\mathcal{I} = I[x_1, \cdots, x_s]$, $s \geq 0$, in attempting to construct $\bar{A}$ either by incremental interpolation or by the Chinese Remainder Algorithm.

The next result relates the ranks of A and A^* and follows easily if we note that a minor of A is nonzero whenever the corresponding minor of A^* is nonzero.

THEOREM 8. *Let $A^* = \theta(A)$ for A a matrix over $\mathcal{I}$. Then* $\operatorname{rank}(A) \geq \operatorname{rank}(A^*)$.

Theorem 8 is used to prove the following theorem, which provides acceptance and rejection tests for homomorphisms and implies the actual criterion applied in the new algorithm.

THEOREM 9. *Given a matrix A over $\mathcal{I}$, let $A^* = \theta(A)$, $r = \operatorname{rank}(A)$, and $r^* = \operatorname{rank}(A^*)$. Then either $r > r^*$ or $(J_A, I_A) \leq (J_{A^*}, I_{A^*})$.*

PROOF. Suppose A is $m \times n$ and nonzero, with $J_A = (j_1, \cdots, j_r)$ and $I_A = (i_1, \cdots, i_m)$. By Theorem 8, $r \geq r^*$. So assume $r = r^*$ and let $J_{A^*} = (j_1^*, \cdots, j_r^*)$ and $I_{A^*} = (i_1^*, \cdots, i_m^*)$. Noting from Theorem 8 that

$$\operatorname{rank}\left(A\begin{bmatrix} 1, & \cdots & , m \\ 1, & \cdots, j_i^* & \end{bmatrix}\right) \geq \operatorname{rank}\left(A^*\begin{pmatrix} 1, & \cdots & , m \\ 1, & \cdots, j_i^* & \end{pmatrix}\right],$$

it follows that $j_i \leq j_i^*$ for $1 \leq i \leq r$. Hence if $j_i < j_i^*$ for some i, then $J_A < J_{A^*}$. Otherwise, $J_A = J_{A^*}$ and then only the case $I_A \neq I_{A^*}$ need be considered. Let u be the least integer, $1 \leq u \leq m$, such that $i_u \neq i_u^*$. Then from the definition of I_A and I_{A^*} it follows that $i_u < i_u^*$, which implies $I_A < I_{A^*}$. Thus in all cases $(J_A, I_A) \leq (J_{A^*}, I_{A^*})$. $\square$

2.4. BOUNDING THE NUMBER OF NONCOMMUTING HOMOMORPHISMS. The modular algorithm for solving linear equations applies the acceptance-rejection tests just derived to both mod-p and evaluation homomorphisms, according as the matrix A is over $I[x_1, \cdots, x_s]$ or $I_p[x_1, \cdots, x_s]$, respectively, in that part of the algorithm. Hence showing that the number of noncommuting (mod-p or evaluation) homomorphisms is bounded, for a given matrix, will provide a guarantee (under minimal assumptions) that the algorithm terminates. Hence, for A a nonzero matrix over $\mathcal{I}$ with $J_A = (j_1, \cdots, j_r)$ and $I_A = (i_1, \cdots, i_m)$, define

$$\delta_k(A) = A\begin{pmatrix} i_1, & \cdots & , i_k \\ j_1, & \cdots & , j_k \end{pmatrix}, \qquad 1 \leq k \leq r.$$

Let $A^* = \theta(A)$ and assume $(J_A, I_A) = (J_{A^*}, I_{A^*})$. Then clearly,

$$A^*\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_k \end{pmatrix} = \delta_k(A^*) \neq 0, \qquad \text{for } 1 \leq k \leq r.$$

On the other hand, suppose $(J_A, I_A) \neq (J_{A^*}, I_{A^*})$ and let $J_{A^*} = (j_1^*, \cdots, j_{r^*}^*)$ and $I_{A^*} = (i_1^*, \cdots, i_m^*)$. Then applying Theorem 9, either (1) $r > r^*$; or (2) $r = r^*$ and $J_A < J_{A^*}$; or (3) $r = r^*$, $J_A = J_{A^*}$, and $I_A < I_{A^*}$. In each of these cases, it can be shown that there is a $k : 1 \leq k \leq r$ such that

$$A^*\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_k \end{pmatrix} = 0.$$

We summarize in the following theorem.

THEOREM 10. *Let A be a matrix over $\mathcal{I}$ and let $A^* = \theta(A)$. Then $(J_A, I_A) = (J_{A^*}, I_{A^*})$ if and only if*

$$A^*\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_k \end{pmatrix} \neq 0 \qquad \text{for } 1 \leq k \leq r.$$

We now apply this theorem to obtain a bound on the number of rejected homomorphisms for a nonzero matrix A. Such a bound will, of course, bound the number of noncommuting homomorphisms for A. By Theorem 10, a homomorphism θ is rejected if and only if

$$A^*\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_k \end{pmatrix} = 0, \qquad \text{for some } k.$$

For an evaluation homomorphism Ψ_a on $\mathcal{I}[x]$, $A\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_k \end{pmatrix}$ has at most $\deg(\delta_k(A))$ distinct roots in $\mathcal{I}$ and so a bound on the number of rejected evaluation mappings for A is given by $\gamma = \sum_{k=1}^{r} \deg(\delta_k(A))$. We note that if $\mathcal{I}[x] = I_p[x_1, \cdots, x_s]$, then there are p distinct evaluation homomorphisms Ψ_a for $a \in I_p$. Hence if a degree bound b, such as that obtained from Theorem 6, satisfies $b \leq p - \gamma$, then there are a sufficient number of the Ψ_a such that $\bar{A}$ can be constructed from the $\Psi_a(\bar{A}^*)$ by incremental interpolation.

Similarly, for mod-p homomorphisms φ_p on $I[x_1, \cdots, x_s]$, let $\rho_k = \lceil \log_2 |\, \delta_k(A)\,|_1 \rceil$, $1 \leq k \leq r$. Then ρ_k bounds the number of distinct prime divisors of the integer coefficients of $\delta_k(A)$, and so a bound ρ on the number of distinct primes p such that

$$\varphi_p\left(A\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_k \end{pmatrix}\right) = 0 \qquad \text{for some } k$$

is given by $\rho = \sum_{k=1}^{r} \rho_k$.

2.5. SUBSTITUTION TESTS AND TERMINATION CRITERIA. Consider the substitution test $CZ = 0$ employed in the proofs of Theorems 1 and 2, where Z is constructed from some RRE matrix E according to formula (4) in Section 1.2. In the modular homomorphism algorithm for solving linear equations, E is most likely the most recent iterate in the construction of a DRRE form $\bar{C}_H$, $H \in \mathcal{P}_C$, by the Chinese Remainder Algorithm or by the Incremental Interpolation Algorithm. The success of this test provides additional information, i.e. it determines the RE sequence of C.

THEOREM 11. *Let C be an $m \times n$ nonzero matrix of rank $r < n$ and E an $m \times n$ nonzero RRE matrix of rank $r' \leq r$, where $J_C \leq J_E$ if $r' = r$. Suppose Z is the matrix constructed from E according to formula* (4). *Then if $CZ = 0$, it follows that $J_C = J_E$.*

PROOF. If either $r' < r$ or $r' = r$ and $J_C < J_E$, it can be shown by using standard arguments that $CZ = 0$ is impossible, contrary to hypothesis. □

If $C = (A, B)$ is the augmented matrix of a linear system, then J_C can be used to test whether or not the system is consistent. If it is, then a general solution can then be obtained from the matrix Z, as described in Theorem 2. This is how Theorem 11 will be

used in the modular algorithm for C over $I[x_1, \cdots, x_s]$, $s \geq 1$. However, its use is different for C over $I_p[x_1, \cdots, x_s]$, $s \geq 1$, in which case the canonical DRRE form $\bar{C}$ for C is computed. This requires the computation of a bound b on the number of evaluation homomorphisms sufficient to construct $\bar{C}$ by incremental interpolation. Such a bound b, constructed using C and J_C, has already been given in Theorem 6, as was pointed out to the author by G. E. Collins.

THEOREM 12. *Let C be a matrix over $I_p[x_1, \cdots, x_s]$, $s \geq 1$, and let b be defined as in Theorem 6. Suppose Ψ_{a_i}, $0 \leq i \leq b$, are distinct evaluation homomorphisms such that each $C^{(i)} = \Psi_{a_i}(C)$ has RE sequence $J_{C^{(i)}} = J_C$ and permutation $I_{C^{(i)}} = H$. Then $H = I_C$. Moreover, if E is the $(b + 1)$-th iterate obtained by applying incremental interpolation to $\overline{C^{(0)}}, \cdots, \overline{C^{(b)}}$, then $E = \bar{C}$.*

PROOF. Suppose $H \neq I_C$ and let k be the least integer such that $h_k \neq i_k$ (i.e. $h_k > i_k$), where $H = (h_1, \cdots, h_m)$ and $I_C = (i_1, \cdots, i_m)$. Of course, $k \leq r$. Since all the $C^{(i)}$ have $I_{C^{(i)}} = H$, it follows by Theorem 10 that $\Psi_{a_i}(\delta_k(C)) = 0$ for $b + 1$ distinct evaluation homomorphisms. This is impossible, since there are at most $\deg(\delta_k(C))$ such homomorphisms (Section 2.4) and since $\deg(\delta_k(C)) \leq b$. Hence $H = I_C$, and by applying Theorem 6 we must have $E = \bar{C}$. □

Returning to the substitution test, we note that the bulk of the computation consists in a matrix multiplication. This test can be made more efficient and elegant in the following way. For E an $m \times n$ RRE matrix with $J_E = (j_1, \cdots, j_r)$ and common diagonal value d, define the *nondiagonal part* $\tilde{E}$ of E as follows. If $r = n$, then $\tilde{E}$ has the value 0, and if $r < n$, then $\tilde{E}$ is the $r \times n-r$ submatrix of E:

$$\tilde{E} = E\begin{bmatrix} 1, \cdots, r \\ k_1, \cdots, k_{n-r} \end{bmatrix}, \tag{5}$$

where $k_1, \cdots, k_{n-r}$ is the complement of $j_1, \cdots, j_r$. All elements omitted from $\tilde{E}$ are known to be zero except the diagonal elements, whose common value d can be saved. Let C' and C'' be the submatrices of C defined by

$$C' = C\begin{bmatrix} 1, \cdots, m \\ j_1, \cdots, j_r \end{bmatrix}, \qquad C'' = C\begin{bmatrix} 1, \cdots, m \\ k_1, \cdots, k_{n-r} \end{bmatrix}. \tag{6}$$

If Z is the $n \times n-r$ matrix defined from E by formula (4), then $CZ = C'\tilde{E} - d \cdot C''$, and so an equivalent form of the *substitution test* is

$$C'\tilde{E} = d \cdot C''. \tag{7}$$

This is the form to be employed in the modular algorithm and it will yield computing time bounds more representative of the substitution tests.

Suppose $C = (A, B)$ is the $m \times n+q$ augmented matrix of a consistent linear system $AX = B$, where A is $m \times n$, and suppose eq. (7) is satisfied by $\tilde{E}$ and d. Then a general solution (d, Y, Z) is obtained if Z is the $n \times n-r$ matrix and Y is the $n \times q$ matrix defined as follows:

$$Z(j_i, j) = \tilde{E}(i, j), \qquad Z(k_j, u) = \begin{Bmatrix} -d, u = j \\ 0, u \neq j \end{Bmatrix}; \tag{8}$$

$$Y(j_i, j) = \tilde{E}(i, n - r + j), \qquad Y(k_j, u) = 0. \tag{9}$$

3. *A Modular Homomorphism Algorithm for Solving Linear Equations*

In this section we present an algorithm for computing a general solution (d, Y, Z) for a system of linear equations $AX = B$ with coefficients in $I[x_1, \cdots, x_s]$, $s \geq 0$. The algorithm is best presented as three separate algorithms—an outer, a middle, and an inner algorithm—corresponding roughly to three distinct levels in computing the solution. The outer or main algorithm (PLES) applies mod-p homomorphisms and the Chinese Remainder Algorithm to obtain a general solution of the system $C = (A, B)$ or reports that it is inconsistent. This algorithm employs the middle algorithm (CPRRE) to com-

pute the canonical DRRE forms for the mod-p homomorphic images C^* of C. Evaluation homomorphisms and the Incremental Interpolation Algorithm for $I_p[x_1, \cdots, x_s]$ are applied in the middle algorithm, which is recursive in the number of variables. At the lowest level of recursion, the inner algorithm (CRRE) is applied to compute the canonical DRRE forms of matrices over the finite fields I_p by employing a Gaussian elimination algorithm. These algorithms are presented in the order: PLES, CPRRE, CRRE. Finally, to apply the substitution test (7), modular algorithms for matrix multiplication and for multiplying a matrix by a single element are given.

3.1. THE OUTER AND MIDDLE ALGORITHMS. In the outer and middle algorithms an iterative process takes place, as described in Section 2, and at each step a new RRE matrix E is constructed as the next iterate by the Chinese Remainder Algorithm or by the Incremental Interpolation Algorithm. What has not been discussed is how Theorem 9 is applied to pairs $(J, I) \in \mathcal{J} \times \mathcal{P}$. The typical situation is as follows for either mod-p or evaluation homomorphisms θ.

A sequence of homomorphisms θ_i, which will be referred to as being *retained*, has been applied and the DRRE forms $\overline{C^{(i)}}$ for the matrices $C^{(i)} = \theta_i(C)$ have been computed. Each $C^{(i)}$ has the same rank r, RE sequence $J_{C^{(i)}} = J$, and permutation $I_{C^{(i)}} = I$. E is the most recent iterate in the construction of some matrix F such that $\theta_i(F) = \overline{C^{(i)}}$, defined as follows, using $J = (j_1, \cdots, j_r)$ and $I = (i_1, \cdots, i_m)$:

$$\overline{C^{(i)}}(k, j) = \begin{cases} C^*\begin{pmatrix} i_1, & \cdots & , i_r \\ j_1, & \cdots, j_{k-1}, j, j_{k+1}, & \cdots, j_r \end{pmatrix} & \text{for } 1 \leq k \leq r, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

It is not yet known whether $J = J_C$ or $I \in \mathcal{P}_C$. Now another homomorphism θ is applied and $J^* = J_{C^*}$, $I^* = I_{C^*}$, and $\overline{C^*}$ are computed, where $C^* = \theta(C)$ and $r^* = \operatorname{rank}(C^*)$. Then one of the following three cases occurs:

$$\begin{aligned} &\text{(a)} \quad r^* < r, \text{ or } r^* = r \text{ and } (J^*, I^*) > (J, I); \\ &\text{(b)} \quad r^* > r, \text{ or } r^* = r \text{ and } (J^*, I^*) < (J, I); \\ &\text{(c)} \quad r^* = r \text{ and } (J^*, I^*) = (J, I). \end{aligned} \tag{11}$$

If (a) occurs, then the new homomorphism θ is *discarded* and is not used to compute a new iterate E; the θ_i are still retained. If (b) occurs, then the previously retained homomorphisms θ_i are discarded along with the previous iterate E, θ becomes the lone retained homomorphism, and E becomes the new first iterate, constructed using only $\overline{C^*}$. If (c) occurs, then θ is retained along with the θ_i, and the next iterate E is constructed using $\overline{C^*}$ and the preceding iterate.

Thus, if a homomorphism is discarded, it is known to be a rejected homomorphism by Theorem 9. It is not yet known if the currently retained homomorphisms are, in fact, accepted homomorphisms. They are, in any case, those for which $\theta_i(C)$ has maximal rank and minimal pair $(J, I) \in \mathcal{J} \times \mathcal{P}$ among all homomorphisms considered thus far. A similar technique for retaining and discarding homomorphisms has been used by Borosh and Fraenkel [4]; a variant on their method is described in [5].

We will continue the discussion of this process after the algorithms have been presented. Recall that in place of the RRE matrices E and $\overline{C^*}$, we compute their common diagonal values d and $d^* = \delta(C^*)$ and their nondiagonal parts $W = \tilde{E}$ and $W^* = \widetilde{C^*}$, respectively.

We first present the main algorithm, which computes a general solution to a system of linear equations with coefficients that are integers or multivariate polynomials over the integers. Note that this algorithm requires that a list $\mathcal{L}$ of distinct odd primes is available, from which primes may be selected in order.

ALGORITHM PLES

Input: An $m \times n'$ matrix C over $I[x_1, \cdots, x_s]$, $s \geq 0$, and an integer $n : 1 \leq n < n'$. $C = (A, B)$ is the augmented matrix of the linear system $AX = B$, where A is $m \times n$ and nonzero and B is $m \times q$, $q = n' - n$.

Output: The triple (d, Y, Z). If the system C is consistent, then the pair (d, Y) is a particular solution for $AX = B$, and Z is a null space basis for A. If the system C is inconsistent, then d, Y, and Z all have 0 as their common value.

Step 1. [Initialize] Set $r = 0$.

Step 2. [Apply a mod-p homomorphism] If the list $\mathfrak{L}$ is exhausted, the algorithm terminates in failure. Otherwise, obtain the next prime p and compute $C^* = \varphi_p(C)$. If C^* is zero, go to step 2.

Step 3. [Apply evaluation homomorphism algorithm] Compute $d^* = \delta(C^*)$, $J^* = J_{C^*}$, $I^* = I_{C^*}$, and nondiagonal part W^* for $\overline{C^*}$ by Algorithm CPRRE (given below).

Step 4. [Apply rejection tests] Set $r^* = \text{rank}(C^*)$. If $J^*[r^*] > n$, set $d = 0$, $Y = 0$, and $Z = 0$, and return (d, Y, Z). If $r^* > r$, go to step 5. If $r^* < r$, to to step 2. If $(J^*, I^*) < (J, I)$, go to step 5. If $(J^*, I^*) > (J, I)$, go to step 2. Otherwise, go to step 6.

Step 5. [Initialize Chinese Remainder Algorithm] Set $r = r^*$, $d = 0$, $J = J^*$, $I = I^*$, $h = 1$. Construct W as the $r \times n'-r$ zero matrix.

Step 6. [Compute next iterate] By the next iteration of the Chinese Remainder Algorithm, compute d' from d, d^*, h, and p, and similarly compute the $r \times n'-r$ matrix W' from W, W^*, h, and p. Replace h by $p \cdot h$.

Step 7. [Equality test] If $d' = d$ and $W' = W$, go to step 8. Otherwise, set $d = d'$, $W = W'$, and go to step 2.

Step 8. [Substitution test] Construct matrices C' and C'' from C and J according to eq. (6). Compute $C'W$ and $d \cdot C''$ by the modular homomorphism algorithms (Section 3.3). If $C'W \neq d \cdot C''$, go to step 2.

Step 9. [Construct general solution] If $r < n$, construct the $n \times n-r$ matrix Z from J, d, and W according to eq. (8). If $r = n$, set $Z = 0$. Construct the $n \times q$ matrix Y according to eq. (9). Return (d, Y, Z).

Most of the individual steps of Algorithm PLES have been discussed, and so its general execution should be fairly clear. The following remarks serve to complement that discussion, focusing on some interesting features of the algorithm. Upon applying the first homomorphism φ_p for which d^*, J^*, I^*, and W^* are obtained, we will have $r^* > r = 0$ in step 4 and so r, d, J, I, W, and h will be initially defined in step 5. Note that r^* is simply the length of sequence J^*. The test $J^*[r^*] > n$ in step 4 will detect an inconsistent system. The remaining tests in step 4 detect the cases (11a), (11b), and (11c) above, with the appropriate choice of the next step. Note that in step 6, $h = p_1 \cdots p_{t-1}$, the product of the primes currently retained prior to this iteration.

The equality test applied to successive iterates, d and d' and W and W', in step 7 is intended to delay the substitution test in step 8 until the earliest point at which it most likely will succeed. One very good reason for this is, of course, the large amount of computation required for matrix multiplication as compared with testing matrix equality (see Section 4.4). Another reason concerns the most likely behavior of the algorithm, which is considered in Section 4.4.

We know that the equality test will eventually succeed, because the successive iterates E being effectively constructed in step 6 will eventually produce $E = F$, an RRE matrix defined using eq. (10), if sufficiently many φ_p are retained. If this happens, the next iterate E' will also equal F, giving a successful equality test. If, however, case (11b) occurs for some φ_p before $E = E'$ occurs, the process begins again with a larger rank r and a new pair (J, I) or with the same rank r but a smaller pair (J, I). By Theorem 9, the number of times this discarding process can occur is clearly bounded for a given matrix C. Hence the equality test eventually succeeds and the substitution test is then applied in step 8. Note that this argument also assures that the inconsistent system test $J^*[r^*] > n$ in step 4 eventually succeeds for C inconsistent.

Continuing the above argument, the substitution test will eventually succeed. If equality occurs when $J \neq J_C$, the substitution test will fail, by Theorem 11. Eventually the retained φ_p will produce a common RE sequence $J = J_C$, and when this happens, it will also be true that $I \in \mathcal{P}_C$. In this case $F = \bar{C}_I$ is being constructed, and if sufficiently many φ_p are retained, the equality test and then the substitution test will succeed, the latter assertion following from Theorem 1. If $I \neq I_C$, then possibly the retained φ_p will

be discarded and the process begun again with a smaller permutation $I \in \mathcal{P}_C$. Of course, this can occur only a finite number of times before the equality and substitution tests succeed.

If a "worst" case did occur for a matrix C, it would be one in which every rejected homomorphism was employed—and later discarded—in such an order that as many as possible of the DRRE forms $\bar{C}_H$ as well as other RRE matrices F were constructed or nearly constructed. The occurrence of such a worst case and even the occurrence of a rejected homomorphism are very unlikely events. An argument to this effect is given in Section 4.4.

Thus once the substitution test is passed, a general solution (d, Y, Z) is constructed in step 9, as specified at the end of Section 2.5. As observed above, the general solution computed is a DLES.

We now give the middle algorithm, an evaluation homomorphism algorithm for computing the canonical DRRE form of a matrix of polynomials over I_p or simply a matrix over I_p.

ALGORITHM CPRRE

Input: An $m \times n$ nonzero matrix C over $I_p[x_1, \cdots, x_s]$, $s \geq 0$.

Output: $d = \delta(C)$, $J = J_C$, $I = I_C$, and the nondiagonal part W for $\bar{C}$.

Step 1. [Apply Gaussian elimination algorithm] If C is a matrix over I_p, compute $d = \delta(C)$, $J = J_C$, $I = I_C$, and W by Algorithm CRRE and return.

Step 2. [Initialize] Set $r = 0$, $a = p$, $k = 0$.

Step 3. [Apply an evaluation homomorphism] Set $a = a - 1$. If $a < 0$, the algorithm terminates in failure. Otherwise, set $C^* = \Psi_a(C)$. If C^* is zero, go to step 3.

Step 4. Apply Algorithm CPRRE recursively to C^*, obtaining $d^* = \delta(C^*)$, $J^* = J_{C^*}$, $I^* = I_{C^*}$, and nondiagonal part W^* for $\overline{C^*}$.

Step 5. [Apply rejection tests] Set $r^* = \text{rank}(C^*)$. If $r^* > r$, go to step 6. If $r^* < r$, go to step 3. If $(J^*, I^*) < (J, I)$, go to step 6. If $(J^*, I^*) > (J, I)$, go to step 3. Otherwise, go to step 7.

Step 6. [Initialize Incremental Interpolation Algorithm] Set $r = r^*$, $d = 0$, $J = J^*$, $I = I^*$. If $r < n$, construct W as the $r \times n-r$ zero matrix. If $r = n$, set $W = 0$. Set $E(x) = 1$.

Step 7. [Compute next iterates] By the next iteration of the Incremental Interpolation Algorithm, compute d' from d, d^*, $E(x)$, and a, and if $r < n$, similarly compute the $r \times n-r$ matrix W' from W, W^*, $E(x)$, and a. If $r = n$, set $W' = 0$. Recompute $E(x) = (x - a) \cdot E(x)$. If $k = 1$, go to step 11.

Step 8. [Equality test] If $d' = d$ and $W' = W$, go to step 9. Otherwise, set $d = d'$, $W = W'$, and go to step 3.

Step 9. [Substitution test] If $r = n$, go to step 10. Construct matrices C' and C'' from C and J according to eq. (6). Compute $C'W$ and $d \cdot C''$ by the evaluation homomorphism algorithms (Section 3.3). If $C'W \neq d \cdot C''$, go to step 3. Otherwise, set $k = 1$.

Step 10. [Degree bound] Compute the DRRE degree bound b as in Theorem 6.

Step 11. [Degree test] If $\deg(E(x)) \leq b$, go to step 3. Otherwise, set $d = d'$, $W = W'$, and return.

Note that steps 2–9 of this algorithm are a direct parallel with steps 1–8 of Algorithm PLES and that the remarks concerning initialization and rejection tests apply similarly to CPRRE (steps 2 and 5). There are basic differences, of course. In step 1 the inner algorithm, CRRE, is evoked when C is over I_p. The elements $a = p - 1, p - 2, \cdots$ of I_p are used for the evaluation homomorphisms Ψ_a. For large primes p, this should be ample. There are obvious differences from PLES in steps 3, 6, and 7 due to replacing reduction modulo a prime and the Chinese Remainder Algorithm by evaluation and interpolation in $I_p[x_1, \cdots, x_s]$ (Section 2.2). The same form for the equality and substitution tests in steps 8 and 9 are used as for PLES.

Following the successful substitution test, the flag k is set to 1 and the DRRE degree bound b is computed. Thereafter, following each interpolation step, the bound b is compared with $\deg(E(x))$, which equals the number of currently retained homomorphisms Ψ_a. We know from Section 2.4 that b is an upper bound on the number of currently re-

tained rejected evaluation homomorphisms, and from Theorem 12 that if $b + 1$ of the Ψ_a have been retained, then they are accepted evaluation homomorphisms and $\bar{C}$ has been constructed. Thus when $\deg(E(x)) = b + 1$ in step 11, we know, in addition to $J = J_C$, that $I = I_C$, $d = \delta(C)$, and W is the nondiagonal part of $\bar{C}$, upon termination in step 11.

Similar observations made concerning the eventual success of the equality and substitution tests for Algorithm PLES can be made for CPRRE. Moreover, as for PLES, the occurrence of even one rejected evaluation homomorphism in CPRRE is unlikely. The reasons for this are outlined in Section 4.4.

3.2. The Inner Algorithm. At the innermost level of the modular homomorphism algorithm, a nonzero matrix C over a finite field I_p is effectively transformed to its canonical DRRE form $\bar{C}$ (i.e. J_C, I_C, the common diagonal value d, and the nondiagonal part W are computed). The algorithm, called CRRE, which accomplishes this employs a Gaussian elimination algorithm which computes an RRE form D over I_p for C. Suppose C is $m \times n$ and of rank r. The Gaussian elimination algorithm performs a forward elimination ("triangularization") during which $\delta(C)$, J_C, and I_C are computed, and a back substitution, completing the "diagonalization." The matrix D is then transformed to $\bar{C}$ by multiplying the first r rows of D by $\delta(C)$. Finally, the nondiagonal part W for $\bar{C}$ is constructed. The Gaussian elimination algorithm over I_p is a classical algorithm and is not included in this paper. Such algorithms are common in the literature (see e.g. [4, 18, 24, 25]). Algorithm CRRE most closely resembles the algorithm sketched in [4].

We describe below the essential computations of the forward elimination which characterize this variant of Gaussian elimination. Initially, $J = \varnothing$, $I = (1, 2, \cdots, m)$, $d = 1$, and $C^{(0)} = C$. For the kth pivot operation, where $1 \leq k \leq r$, J, I, and d are recomputed, and matrix $C^{(k-1)}$ is transformed to $C^{(k)}$ as follows. The pivot search scans columns left to right and within each column the rows $k, k + 1, \cdots, m$ in order, eventually locating a pivot element $C^{(k-1)}(t, s) \neq 0$. Next s is suffixed to the sequence J (we note that $s = j_k = J_C[k]$) and d is recomputed as $d = d \cdot C^{(k-1)}(t, s)$. The change in the row order of $C^{(k-1)}$ is performed next to initially define $C^{(k)}$, using the *row permutation* $\tau_k \in \mathcal{P}_m$:

$$\tau_k(h) = \begin{cases} h & \text{if } 1 \leq h < k \text{ or } t < h \leq m, \\ k & \text{if } h = t, \\ h + 1 & \text{if } k \leq h < t. \end{cases} \tag{12}$$

That is, row $\tau_k(h)$ of $C^{(k)}$ equals row h of $C^{(k-1)}$ for $1 \leq h \leq m$. Thus $C^{(k)}(k, s)$ is the pivot element. The permutation I is similarly defined by changing the order of the elements: replace $I[\tau_k(h)]$ by $I[h]$, $1 \leq h \leq m$. Supposing inductively that $I[i] = I_C[i]$, $1 \leq i < k$, and that $I[k] < I[k + 1] < \cdots < I[m]$ after the $(k - 1)$th pivot operation, it can be shown that after the kth pivot operation, we have that $I[k + 1] < \cdots < I[m]$, that $I[1], \cdots, I[k - 1]$ have not been altered, and that $I[k] = I_C[k]$. This latter result requires that the minors of C defining the elements of $C^{(k-1)}$ be known explicitly.

The kth pivot operation is completed by the usual transformation of rows, $k, k + 1, \cdots, m$. These computations are listed here for reference during the computing time analysis in Section 4.3:

(a) Letting $e = [C^{(k)}(k, s)]^{-1}$, replace $C^{(k)}(k, j)$ by $e \cdot C^{(k)}(k, j)$, $s \leq j \leq n$.
(b) For each row $h > k$ whose pivot column element $C^{(k)}(h, s)$ is nonzero, replace $C^{(k)}(h, j)$ by $C^{(k)}(h, j) - C^{(k)}(h, s) \cdot C^{(k)}(k, j)$, $s \leq j \leq n$. (13)

Thus at the conclusion of the forward elimination, we have $J = J_C = (j_1, \cdots, j_r)$ and $I = I_C = (i_1, \cdots, i_m)$. Moreover, it can also be shown that $d = \delta(C)$ and that $C^{(r)} = M$, which is defined by

$$M(k, j) = \begin{cases} C\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_{k-1}, j \end{pmatrix} \Big/ C\begin{pmatrix} i_1, \cdots, i_k \\ j_1, \cdots, j_k \end{pmatrix} \\ \qquad \text{for } 1 \leq k \leq r \text{ and } j_k \leq j \leq n, \\ 0 \quad \text{elsewhere.} \end{cases} \tag{14}$$

We note that the numerators of the elements of M are the corresponding entries in $\hat{C}$ as defined by formula (1) for $H = I_C$.

The completion of the diagonalization by back substitution produces the matrix D:

$$D(k,j) = \begin{cases} C\begin{pmatrix} i_1, \cdots, i_r \\ j_1, \cdots, j_{k-1}, j, j_{k+1}, \cdots, j_r \end{pmatrix} \Big/ C\begin{pmatrix} i_1, \cdots, i_r \\ j_1, \cdots, j_r \end{pmatrix} \\ \qquad \text{for} \quad 1 \leq k \leq r \quad \text{and} \quad j_k \leq j \leq n, \\ 0 \qquad \text{elsewhere.} \end{cases} \tag{15}$$

This is a general form of Cramer's Rule, and in fact can be proved using that result. Detailed proofs of these assertions are straightforward but tedious, as mentioned concerning the exact division algorithm in Section 1.2, and may be found in [24]. We note that the numerators of the elements of D are the corresponding elements of $\bar{C}$. Hence multiplying rows $1, \cdots, r$ by d produces $\bar{C}$, and the nondiagonal part W is easily constructed.

3.3. MATRIX MULTIPLICATION. As discussed above, in the substitution test (7), both the outer and the middle algorithms will require the computation of a matrix product and the multiplication of a matrix by an element of the integral domain (i.e. $I[x_1, \cdots, x_s]$, $s \geq 0$, or $I_p[x_1, \cdots, x_s]$, $s \geq 1$). As is made clear in Section 4, these computations can best be done by modular (and evaluation) homomorphism algorithms.

The algorithm for matrix multiplication is organized like the linear equations algorithm as outer, middle, and inner algorithms. We now outline these, beginning with the outer algorithm (MMPY), a *modular homomorphism algorithm for matrix multiplication*. Given matrices A and B over $I[x_1, \cdots, x_s]$, $s \geq 0$, for which the product $C = AB$ is defined (i.e. A $m \times n$ and B $n \times q$), compute C as follows.

Step 1. Compute a bound K on the magnitudes of the integer coefficients of the elements of C by

$$K = 2 \cdot \sum_{h=1}^{n} K_h \cdot K_h', \qquad K_h = \max_i | A(i,h) |_1, \quad K_h' = \max_j | B(h,j) |_1 .$$

Step 2. For t odd primes p_i such that $p_1 \cdots p_t > K$, compute $A^* = \varphi_{p_i}(A)$ and $B^* = \varphi_{p_i}(B)$.

Step 3. Compute the t products A^*B^* by applying the evaluation homomorphism algorithm below, obtaining images $C^* = A^*B^*$.

Step 4. By t iterations of the Chinese Remainder Algorithm applied to the C^*, compute C as the tth iterate.

The middle algorithm (MCPMPY), an *evaluation homomorphism algorithm for matrix multiplication*, for A and B matrices over $I_p[x_1, \cdots, x_s]$, $s \geq 0$, has a similar format. The case $s = 0$ is handled separately in Step 1, which outlines the inner algorithm, a classical matrix multiplication algorithm.

Step 1. If A and B are over I_p, compute $C = AB$ by taking inner products of the rows of A and the columns of B.

Step 2. Otherwise, compute a bound η on the degrees in the main variable of the elements of the product AB:

$$\eta = \max_{i,j,k}\{\deg(A(i,k)) + \deg(B(k,j))\}.$$

Step 3. Apply $\eta + 1$ evaluation homomorphisms Ψ_a, obtaining $A^* = \Psi_a(A)$ and $B^* = \Psi_a(B)$ for each.

Step 4. Compute $\eta + 1$ products $C^* = A^*B^*$ by applying the algorithm recursively.

Step 5. By $\eta + 1$ iterations of the Incremental Interpolation Algorithm applied to the C^*, compute $C = AB$ as the $(\eta + 1)$th iterate.

Consider briefly the multiplication of a matrix A over $\mathcal{I} = I[x_1, \cdots, x_s]$, $s \geq 0$, by an element b in $\mathcal{I}$. This requires simply computing the products $b \cdot A(i,j)$ by a modular homomorphism algorithm (PMPY) for multiplication in $I[x_1, \cdots, x_s]$. For the case $\mathcal{I} = I_p[x_1, \cdots, x_s]$, an evaluation homomorphism algorithm (CPMPY) is used. These algorithms are described in [14]. We note that multiplication in I and $I_p[x]$ is just as well

accomplished by the classical algorithms. For matrix multiplication, the classical algorithm would just as well be applied for I.

4. *Computing Time Analysis*

4.1. Basic Definitions and Notation. Let f and g be real valued functions on a set S. If there is a positive real number c such that $|f(x)| \leq c \cdot |g(x)|$ for all $x \in S$, then we say that f is *dominated* by g and we write $f \lesssim g$. If $f \lesssim g$ and $g \lesssim f$, we say that f and g are *codominant* and we write $f \sim g$. Of course, $f \gtrsim g$ means $g \lesssim f$. This notation, which will be used to analyze the computing times of the algorithms, is due to Collins [13] (see also [8]) and was introduced as an improvement to the big-O notation of Knuth [19].

The general framework for the analysis is as follows. We are given an algorithm $\mathfrak{A}$ and a set S of "inputs" to $\mathfrak{A}$. We also have a set R whose elements "describe" the inputs; i.e. corresponding to each $y \in R$ there is a unique subset S_y of S, which will be assumed to be finite and nonempty. Note that each $x \in S$ can be considered an n-tuple and each $y \in R$ an m-tuple. Let $t_{\mathfrak{A}}(x)$ denote the time (in some convenient unit) to apply algorithm $\mathfrak{A}$ to the input x. Then we can define the function $T_{\mathfrak{A}}$ on R by $T_{\mathfrak{A}}(y) = \max\{t_{\mathfrak{A}}(x) : x \in S_y\}$. We call $T_{\mathfrak{A}}$ the *maximum computing time function* for the algorithm $\mathfrak{A}$ with respect to the set R. Analogously, we can define the *minimum computing time function* $U_{\mathfrak{A}}$ for algorithm $\mathfrak{A}$ by $U_{\mathfrak{A}}(y) = \min\{t_{\mathfrak{A}}(x) : x \in S_y\}$. For some algorithms there is a significant difference between the maximum computing time (i.e. the worst case) and the time for a typical computation. A computing time function $V_{\mathfrak{A}}$ reflecting the expected behavior of an algorithm $\mathfrak{A}$ is thus defined by $V_{\mathfrak{A}}(y) = [\sum_{x \in S_y} t_{\mathfrak{A}}(x)]/\| S_y \|$, where $\| Q \|$ denotes the number of elements in a finite set Q. The function $V_{\mathfrak{A}}$ is called the *average computing time function* for algorithm $\mathfrak{A}$ with respect to set R. (These definitions of computing time functions are slightly more formal versions of definitions due to Collins.)

For each algorithm $\mathfrak{A}$, dominance relations for the maximum computing time functions $T_{\mathfrak{A}}$ will be found; i.e. an explicitly given function f defined on R will be found such that $T_{\mathfrak{A}} \lesssim f$. Since clearly $U_{\mathfrak{A}} \lesssim V_{\mathfrak{A}} \lesssim T_{\mathfrak{A}}$, it will also be true that $U_{\mathfrak{A}} \lesssim f$ and $V_{\mathfrak{A}} \lesssim f$. For some algorithms it is possible to show that $U_{\mathfrak{A}} \sim T_{\mathfrak{A}} \sim f$, in which case we also have $V_{\mathfrak{A}} \sim f$. Such functions f will be referred to appropriately as *dominating* or *codominating functions* for the maximum computing time functions of algorithm $\mathfrak{A}$ and will be denoted by $T_{\mathfrak{A}}'$.

The analyses of the algorithms will require specifying the sets R and the mappings of R to subsets S_y of the input sets S. The elements of S usually involve matrices over $I[x_1, \cdots, x_s]$ or $I_p[x_1, \cdots, x_s]$, $s \geq 0$, and so we begin by specifying certain subsets of these sets.

Let $P(c)$ denote the set of all integers b such that $|b| < c$. Let $P(c, m_1, \cdots, m_s)$, $s \geq 1$, denote the set of all polynomials A in $I[x_1, \cdots, x_s]$ such that $|A|_1 < c$ and the degree of A in x_i is at most m_i. Also, for $s \geq 1$, let $P^*(m_1, \cdots, m_s)$ denote the set of all polynomials A in $I_p[x_1, \cdots, x_s]$ such that the degree of A in x_i is at most m_i. For $s = 0$, we let $P(c, m_1, \cdots, m_s)$ denote $P(c)$ and $P^*(m_1, \cdots, m_s)$ denote I_p. Let $M(m, n, U)$ denote the set of all $m \times n$ matrices with elements in the set U. The sets S_y will then be nearly or completely specified by requiring the matrices input to be in either $M(m, n, P(c, m_1, \cdots, m_s))$ or in $M(m, n, P^*(m_1, \cdots, m_s))$ for some s.

Since sequences of degree bounds $m_1, \cdots, m_s$ occur frequently, we let $\mathbf{m}_s$ denote the vector $(m_1, \cdots, m_s)$ while letting $\mathbf{m}_0$ denote the null vector $\varnothing$. The sequences $(m_1, \cdots, m_s)$ will be used interchangeably with $\mathbf{m}_s$. Hence $P(c, m_1, \cdots, m_s)$ is the same as $P(c, \mathbf{m}_s)$, and so on. Similarly for a second sequence of degree bounds $n_1, \cdots, n_s$ and the corresponding vector $\mathbf{n}_s$. By defining $\mu_i = m_i + 1$ we obtain the frequently recurring quantity $\prod_{i=1}^{s} \mu_i$, which is, for example, the maximum number of terms of an element in $P(c, \mathbf{m}_s)$. For $s = 0$, the value of $\prod_{i=1}^{s} \mu_i$ is taken to be 1, and in a similar fashion the value of $\sum_{i=1}^{s} \mu_i$ is taken to be 0.

The analysis requires that we know the computing times for operations of infinite-precision integers. Such times are best given in terms of the lengths of integers and the log function using an arbitrary and unspecified base β suffices for this. For example, the times to add and multiply positive integers c and d are $\sim (\log c) + (\log d)$ and $\sim (\log c)(\log d)$, respectively, and the times for these operations between integer coefficients of $A \in P(c, \mathbf{m}_s)$ and $B \in P(d, \mathbf{n}_s)$ are $\lesssim \log c + \log d$ and $\lesssim (\log c)(\log d)$, respectively.

The algorithms are designed so that they can operate on mathematical objects (polynomials, matrices, vectors) which are represented as list structures. This has, in fact, been done [24, 15] using Collins' SAC-1 System for Symbolic and Algebraic Calculation (see [12] for an overview of this system). The general analysis which follows is valid for this particular kind of implementation. This takes into account those additional operations for space management, including the time for the dynamic erasure of list structures, which is required by reference count systems.

As mentioned earlier, in implementations of the algorithms, the primes p employed for mod-p homomorphisms are large—nearly as large as can fit in a machine word. For example, on the Univac 1108, with a 36-bit word, the primes are ordinarily greater than 10^9.

4.2. AUXILIARY ALGORITHMS. The computing times for the basic arithmetic operations on elements in $I[x_1, \cdots, x_s]$ and in $I_p[x_1, \cdots, x_s]$, $s \geq 0$, are as given in [10] and [11], respectively (see also [8]). The times for those basic operations which are omitted can be easily derived. For $A \in P(c, m_1, \cdots, m_s)$, $s \geq 0$, the time to apply a mod-p homomorphism φ_p to A is $\lesssim (\log c) \prod_{i=1}^{s} (m_i + 1)$. In constructing A from images $A^* = \varphi_p(A)$ in $P^*(m_1, \cdots, m_s)$ by the Chinese Remainder Algorithm, the time to compute the hth iterate is $\lesssim h \cdot \prod_{i=1}^{s}(m_i + 1)$. For $A \in P^*(m_1, \cdots, m_s)$, $s \geq 1$, the time to apply an evaluation homomorphism Ψ_a to A is $\lesssim \prod_{i=1}^{s}(m_i + 1)$. In constructing A from images $A^* = \Psi_a(A)$ in $P^*(m_1, \cdots, m_{s-1})$ by the Incremental Interpolation Algorithm, the time to compute the hth iterate is $\lesssim (h + 1) \cdot \prod_{i=1}^{s-1} (m_i + 1)$.

Now let us consider matrix multiplication and, in particular, the evaluation homomorphism algorithm MCPMPY. Let $T_{\text{MCPMPY}}(m, n, q, \mathbf{m}_s, \mathbf{n}_s)$ denote the maximum computing time function of MCPMPY for $A \in M(m, n, P^*(\mathbf{m}_s))$ and $B \in M(n, q, P^*(\mathbf{n}_s))$, $s \geq 0$. Let $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $l_i = m_i + n_i$, $\lambda_i = l_i + 1$ and $\lambda = \max \lambda_i$ (or $\lambda = 1$ if $s = 0$).

THEOREM 13. *The maximum computing time functions of MCPMPY satisfy: for* $s = 0$,

$$T_{\text{MCPMPY}} \sim mnq$$

and for $s \geq 1$,

$$T_{\text{MCPMPY}} \lesssim mnq \left\{\prod_{i=1}^{s} \lambda_i\right\} + mn \left\{\sum_{i=1}^{s} \left(\prod_{j=1}^{i} \mu_j\right)\left(\prod_{j=i}^{s} \lambda_j\right)\right\} + nq \left\{\sum_{i=1}^{s} \left(\prod_{j=1}^{i} \nu_j\right)\left(\prod_{j=i}^{s} \lambda_j\right)\right\} + mq \left\{\left(\prod_{i=1}^{s} \lambda_i\right)\left(\sum_{i=1}^{s} \lambda_i\right)\right\}.$$

PROOF. For $s = 0$, mq inner products of vectors of length n over I_p are computed in step 1, and so $T_{\text{MCPMPY}} \sim mnq$. If $s \geq 1$, then the time to compute the degree bound η in step 2 is $\lesssim mnq = T_2'$, where $\eta \lesssim \mu_s + \nu_s \sim \lambda_s$. Hence steps 3, 4, and 5 are executed at most λ_s times, and the times T_i for these steps satisfy

$$T_3 \lesssim \lambda_s \cdot mn \left(\prod_{i=1}^{s} \mu_i\right) + \lambda_s \cdot nq \left(\prod_{i=1}^{s} \nu_i\right) = T_3',$$

$$T_4 \lesssim \lambda_s \cdot T'_{\text{MCPMPY}}(m, n, q, \mathbf{m}_{s-1}, \mathbf{n}_{s-1}) = T_4',$$

$$T_5 \lesssim \lambda_s \cdot mq\lambda_s \left(\prod_{i=1}^{s-1} \lambda_i\right) = T_5'.$$

Thus $T_{\text{MCPMPY}} \lesssim T_2' + T_3' + T_4' + T_5'$, and it is easily shown, for $s = 1$, that $T_{\text{MCPMPY}} \lesssim mnq\lambda_1 + mn\mu_1\lambda_1 + nq\nu_1\lambda_1 + mq\lambda_1^2$. For $s \geq 2$, assume the theorem holds for

$s-1$. Then since $T_2' \lesssim T_4'$, $T_{\text{MCPMPY}} \lesssim T_3' + T_4' + T_5'$, and by obtaining T_4' explicitly from the inductive hypothesis and simply rearranging terms, the inductive step is completed. □

The following are more compact but cruder bounds.

COROLLARY 1. $T_{\text{MCPMPY}} \lesssim (\prod_{i=1}^{s} \lambda_i)[mnq + (mn + nq + mq)(\sum_{i=1}^{s} \lambda_i)]$.

COROLLARY 2. *If* $m \sim n \sim q$ *and* $\lambda = \max_{1 \le i \le s} \lambda_i$, *then* $T_{\text{MCPMPY}} \lesssim m^2 \lambda^s (m + \lambda)$.

We next consider the modular homomorphism algorithm for matrix multiplication, MMPY. Let $T_{\text{MMPY}}(m, n, q, c, \mathbf{m}_s, d, \mathbf{n}_s)$ be the maximum computing time function of MMPY for $A \in M(m, n, P(c, \mathbf{m}_s))$ and $B \in M(n, q, P(d, \mathbf{n}_s))$, $s \ge 0$.

THEOREM 14. *The maximum computing time functions of MMPY satisfy,* for $s \ge 0$,

$$T_{\text{MMPY}} \lesssim (\log ncd) \cdot \Bigg[mnq \left(\prod_{i=1}^{s} \lambda_i \right) + mn \left\{ (\log c) \left(\prod_{i=1}^{s} \mu_i \right) + \sum_{i=1}^{s} \left(\prod_{j=1}^{i} \mu_j \right) \left(\prod_{j=i}^{s} \lambda_j \right) \right\} + nq \left\{ (\log d) \left(\prod_{i=1}^{s} \nu_i \right) + \sum_{i=1}^{s} \left(\prod_{j=1}^{i} \nu_j \right) \left(\prod_{j=i}^{s} \lambda_j \right) \right\} + mq \left(\prod_{i=1}^{s} \lambda_i \right) \left(\log cd + \sum_{i=1}^{s} \lambda_i \right) \Bigg].$$

PROOF. The time to compute the bound K in step 1 can be shown to be $\lesssim mn(\log c)(\prod_{i=1}^{s} \mu_i) + nq(\log d)(\prod_{i=1}^{s} \nu_i) + n(\log c)(\log d) = T_1'$. Moreover, if t is the number of primes required, then $t \sim \log K \lesssim \log ncd$. So the number of times steps 2, 3, and 4 are executed is $\lesssim \log ncd$ and we have

$$T_2 \lesssim (\log ncd)[mn(\log c) \left(\prod_{i=1}^{s} \mu_i \right) + nq(\log d) \left(\prod_{i=1}^{s} \nu_i \right)] = T_2',$$

$$T_3 \lesssim (\log ncd) \cdot T_{\text{MCPMPY}}(m, n, q, \mathbf{m}_s, \mathbf{n}_s) = T_3',$$

$$T_4 \lesssim (\log ncd) \cdot mq(\log ncd) \left(\prod_{i=1}^{s} \lambda_i \right) = T_4'.$$

By substituting for $T_{\text{MCPMPY}}(m, n, q, \mathbf{m}_s, \mathbf{n}_s)$ in T_3' and observing that $T_1' \lesssim T_2'$, we find $T_{\text{MMPY}}(m, n, q, c, \mathbf{m}_s, d, \mathbf{n}_s) \lesssim T_2' + T_3' + T_4'$. Then by codominance and rearrangement of terms, the theorem follows. □

The following more compact but cruder bounds are obtained for MMPY.

COROLLARY 3. *If* $c \sim d$, *then* $T_{\text{MMPY}} \lesssim (\log nc)(\prod_{i=1}^{s} \lambda_i)[mnq + (mn + nq + mq)(\log c + \sum_{i=1}^{s} \lambda_i)]$.

COROLLARY 4. *If* $c \sim d$ *and* $m \sim n \sim q$, *then* $T_{\text{MMPY}} \lesssim m^2 \lambda^s (\log mc)(m + \lambda + \log c)$.

Now consider multiplying a matrix A by a "scalar," i.e. by an element b of the integral domain $\mathcal{I}$ containing the matrix elements. Let $T_{\text{MSCPMPY}}(m, n, \mathbf{m}_s, \mathbf{n}_s)$ denote the maximum computing time function of the evaluation homomorphism algorithm for matrix-scalar multiplication, MSCPMPY, for $A \in M(m, n, P^*(\mathbf{m}_s))$ and $b \in P^*(\mathbf{n}_s)$. The time to compute each product $b \cdot A(i, j)$ by CPMPY is $\lesssim (\prod_{i=1}^{s} \lambda_i)(\sum_{i=1}^{s} \lambda_i)$, as given in [14].

THEOREM 15. *The maximum computing time functions of MSCPMPY satisfy: for* $s = 0$, $T_{\text{MSCPMPY}} \sim mn$, *and for* $s \ge 1$,

$$T_{\text{MSCPMPY}} \lesssim mn \left(\prod_{i=1}^{s} \lambda_i \right) \left(\sum_{i=1}^{s} \lambda_i \right).$$

Let $T_{\text{MSMPY}}(m, n, c, \mathbf{m}_s, d, \mathbf{n}_s)$ denote the maximum computing time function of the modular homomorphism algorithm for matrix-scalar multiplication, MSMPY, for $A \in M(m, n, P(c, \mathbf{m}_s))$ and $b \in P(d, \mathbf{n}_s)$. The time to compute each product $b \cdot A(i, j)$ by PMPY is $\lesssim (\log cd)(\prod_{i=1}^{s} \lambda_i)(\log cd + \sum_{i=1}^{s} \lambda_i)$, as given in [14].

THEOREM 16. *The maximum computing time functions of MSMPY satisfy, for* $s \ge 0$,

$$T_{\text{MSMPY}} \lesssim mn(\log cd) \left(\prod_{i=1}^{s} \lambda_i \right) \left(\log cd + \sum_{i=1}^{s} \lambda_i \right).$$

4.3. THE INNER ALGORITHM. We now begin an analysis of the modular homomorphism algorithm for solving linear equations. We begin with algorithm CRRE, for which we obtain a codominance relation. To obtain codominance, the following elementary result is needed.

LEMMA 1. *If r, m, and n are integers such that $1 \leq r \leq \min(m, n)$, then*

$$\sum_{i=1}^{r} (m - i + 1)(n - i + 1) \geq rmn/8.$$

Let $T_{\text{CRRE}}(m, n, r)$ be the maximum computing time function of CRRE for $C \in M(m, n, I_p)$ and C of rank r.

THEOREM 17. $T_{\text{CRRE}} \sim mnr$.

PROOF. Let $J_C = (j_1, \cdots, j_r)$ and $j_0 = 0$. Consider the forward elimination and in particular the kth pivot operation. There are at most $(j_k - j_{k-1})(m - k + 1)$ elements tested for zero and at most $n(m - k + 1)$ elements moved. Referring to Eq. (13) in Section 3.2, there are $(n - j_k + 1)$ multiplications of pivot row elements and at most $2(m - k)(n - j_k + 1)$ arithmetic operations to perform the elimination on rows $k + 1, \cdots, m$. Thus the number of operations (arithmetic and moving) for the kth pivot operation is at most $5n(m - k + 1)$. The time for any arithmetic operation in I_p is bounded, and it may be safely assumed that the time for each move is likewise bounded. Since $\sum_{k=1}^{r} n(m - k + 1) \leq mnr$, the time for the forward elimination is $\lesssim mnr$. It can be easily verified that the time for the back substitution is $\lesssim nr^2$, that the time to compute $\tilde{C}$ from D is $\lesssim nr$, and that the time to construct the nondiagonal part W for $\tilde{C}$ is $\lesssim nr$. Since all other computation is $\lesssim mnr$, clearly $T_{\text{CRRE}} \lesssim mnr$.

To establish the theorem, we now show that $T_{\text{CRRE}} \gtrsim mnr$. This will be accomplished if an $m \times n$ matrix C of rank r can be found such that during the forward elimination, the kth pivot is found in row k and column k (i.e. $J_C = (1, 2, \cdots, r)$ and $I_C = (1, 2, \cdots, m)$) and each element below the pivot element in rows $k + 1, \cdots, m$ in nonzero. Then at least $(m - k + 1)(n - k + 1)$ operations are required by the kth pivot operation and at least $N = \sum_{k=1}^{r} (m - k + 1)(n - k + 1)$ arithmetic operations are required by the forward elimination. From Lemma 1 we have that $8N \geq mnr$, and so $N \gtrsim mnr$. A matrix C satisfying these conditions is defined by

$$C(i, j) = \begin{cases} 1 & j \leq i \leq r \text{ or } i > r \text{ and } j \leq r, \\ 0 & \text{elsewhere.} \end{cases}$$

(More interesting matrices C can be found; for example, a Vandermonde matrix is used in [24].) Thus $T_{\text{CRRE}} \gtrsim N \gtrsim mnr$. □

4.4. THE MIDDLE AND OUTER ALGORITHMS. The analysis of the middle and outer algorithms for solving linear equations requires some prior consideration of the amount of "wasted" computation that might be performed. There are two main categories of wasted computation: (1) the computations of $C^* = \theta(C)$ and $\overline{C^*}$, and the iterations of the Chinese Remainder Algorithm or the Incremental Interpolation Algorithm, for retained homomorphisms θ which are later discarded; and (2) unsuccessful substitution tests. The occurrences of computations (1) and (2) depend on the probabilities of occurrence of the following events (1′) and (2′), respectively:

(1′) the use of a rejected homomorphism θ;

(2′) a premature equality test success (i.e. one followed by an unsuccessful substitution test).

We now sketch arguments which infer that the probabilities of the occurrence of (1′) or (2′) are very small.

Consider first the possible use of a rejected homomorphism θ for a matrix C. From Theorem 10 we know that θ is rejected if and only if $\theta(\delta_k(C)) = 0$ for some $k : 1 \leq k \leq r$. In case θ is an evaluation homomorphism Ψ_a, $a \in I_p$, on $I_p[x_1, \cdots, x_s]$, then Ψ_a is rejected if and only if a is a root in x_s of some $\delta_k(C)$. If θ is a mod-p homomorphism φ_p on $I[x_1, \cdots, x_s]$, $s \geq 0$, then φ_p is rejected if and only if p divides all the integer coefficients of some $\delta_k(C)$ for $s \geq 1$, or simply divides the integer $\delta_k(C)$ for $s = 0$. Assuming that only

large primes p are used, as stated in Section 4.1, convincing arguments can be given which show that the probabilities of $\Psi_a(\delta_k(C)) = 0$ or $\varphi_p(\delta_k(C)) = 0$ occurring are very small. Summing over k yields still very small probabilities [24]. We note that similar arguments have been given by Brown [8] for a modular algorithm for multivariate polynomial GCD calculation.

The treatment of premature equality test successes is similar. A sequence of matrix iterates $C^{(1)}, C^{(2)}, \cdots$ is computed, and the equality test succeeds when $C^{(k)} = C^{(k-1)}$ for some k. Consider corresponding entries of the $C^{(i)}$, $A_1, A_2, \cdots$, and let t be the least integer such that $A_i = A_{i-1}$ for all $i > t$. We consider the probability that $A_k = A_{k-1}$, $k < t$. Suppose an evaluation homomorphism Ψ_a is applied to compute $A_k \in I_p[x_1, \cdots, x_s]$ by the Incremental Interpolation Algorithm. Letting

$$f_k(x_1, \cdots, x_s) = b_k(x_1, \cdots, x_{s-1}) - A_{k-1}(x_1, \cdots, x_s)$$

in step 2b of this algorithm, it follows that $A_k = A_{k-1}$ if and only if $f_k(x_1, \cdots, x_{s-1}, a) = 0$, i.e. if and only if a is a root in x_s of f_k. On the other hand, suppose a mod-p mapping φ_p is applied to compute $A_k \in I[x_1, \cdots, x_s]$, $s \geq 0$, by the Chinese Remainder Algorithm. Letting $f_k(x_1, \cdots, x_s) = a_k(x_1, \cdots, x_s) - b_{k-1}(x_1, \cdots, x_s)$ (in $I[x_1, \cdots, x_s]$) from step 2a of this algorithm, where $a_k \in I_p[x_1, \cdots, x_s]$, we see that $A_k = A_{k-1}$ if and only if $a_k - \varphi_p(b_k) = 0$, i.e. if and only if p divides all the integer coefficients of f_k for $s \geq 1$, or simply divides the integer f_k for $s = 0$. As for rejected homomorphisms, if large primes are used, convincing arguments can be given to support the assertion that the probabilities of occurrence of $\Psi_a(f_k) = 0$ or $\varphi_p(f_k) = 0$ are very small. Summing over k yields still small probabilities. Moreover, the probability that *all* corresponding entries of $C^{(k)}$ and $C^{(k-1)}$ are simultaneously prematurely equal is indeed much smaller (again see [24]).

Thus it will be a rare occurrence when a homomorphism (mod-p or evaluation) is applied that is not an accepted homomorphism. So, with high probability, the canonical DRRE form $\bar{C}$ for C is being computed. Should a rejected homomorphism occur, it would rarely be followed by another, and so a minimal amount of computation would be expended on it. In addition, since it is unlikely that a premature equality success will occur, it is with high probability that equality first occurs for $C^{(k)} = C^{(k-1)} = \bar{C}$. If we were to account for all the possible "bad" cases (involving rejected homomorphisms) that might occur, along with the additional computation involved, the task of obtaining maximum or average computing times would be quite difficult. Moreover, in view of the fact that the chances are extremely small that a significant number of rejected homomorphisms occurs, it is reasonable to perform the computing time analysis under the assumption that no rejected homomorphisms occur and no premature equality test successes occur. Thus it can be expected that all homomorphisms are employed to construct the canonical DRRE form $\bar{C}$. The computing time dominance relations so obtained should consequently be accurate and meaningful.

The approach is thus to obtain maximum computing times under these assumptions for the middle and outer algorithms. We begin with the middle algorithm, CPRRE. Let $T_{\text{CPRRE}}(m, n, r, \mathbf{m}_s)$ denote the maximum computing time function of CPRRE for $C \in M(m, n, P^*(\mathbf{m}_s))$ and C of rank r. Define $\mu_i = m_i + 1$.

THEOREM 18. *Under the assumptions that no rejected evaluation homomorphisms are used and no premature equality test successes occur, the maximum computing time functions of CPRRE satisfy: for $s = 0$, $T_{\text{CPRRE}} \sim mnr$, and for $s \geq 1$,*

$$T_{\text{CPRRE}} \lesssim mr^{s+1}\left(\prod_{i=1}^{s} \mu_i\right)\left(n + (n - r + 1)\left(\sum_{i=1}^{s} \mu_i\right)\right).$$

PROOF. For $s = 0$, the algorithm simply applies CRRE in step 1, and so $T_{\text{CPRRE}} \sim T_{\text{CRRE}} \sim mnr$. Suppose $s \geq 1$. Steps 1, 2, and 10 are each executed once. Since no evaluation homomorphisms are discarded, step 6 is executed once, and since no premature equality test successes occur, step 9 is also executed just once. Since the nonzero elements of $\bar{C}$ are determinants of order r, by applying Theorem 4 we see that each element

of $\tilde{C}$ is in $P^*(rm_1, \cdots, rm_s)$, i.e. the maximum degree in x_i attainable by a solution component is rm_i. Hence the number t of iterations of steps 3, 4, 5, 7, and 8 satisfies $t \lesssim r\mu_s$. Step 11 is executed at most $t \lesssim r\mu_s$ times also.

Let $M_k = \prod_{i=1}^{k} \mu_i$ and $M_k' = \sum_{i=1}^{k} \mu_i$. The times for single executions of the individual steps are now obtained. The time for each of steps 1 and 2 is ~ 1. The time for step 3 is $\lesssim mnM_s$ and the time for step 4 is $\lesssim T_{\text{CPRRE}}(m, n, r, \mathbf{m}_{s-1})$, the latter holding since C^* has the same rank r as C under the hypothesis that no rejected evaluation homomorphisms occur. The time for step 5 is $\lesssim m$ and the time for step 6 is $\lesssim rn$. Consider the interpolations of step 7 and note that each polynomial interpolant has degree less than $r\mu_i$ in x_i. The time for the jth polynomial interpolation is thus

$$\lesssim j \cdot \prod_{i=1}^{s-1} r\mu_i \lesssim r\mu_s \cdot r^{s-1} M_{s-1} = r^s M_s .$$

The number of such interpolations is $\lesssim r(n - r) + 1 \leq r(n - r + 1)$, and so the time for each execution of step 7 is $\lesssim r^{s+1}(n - r + 1)M_s$. Each execution of step 8 involves at most $r(n - r + 1)$ polynomial subtractions, where the time for each is $\lesssim j \cdot \prod_{i=1}^{s-1} r\mu_i$ at the jth iteration. So the time for an execution of step 8 is $\lesssim r^{s+1}(n - r + 1)M_s$. Assuming a fixed time to obtain the degree of a polynomial, it is easily shown that the time for step 10 is $\sim mn$. The time for step 11 is ~ 1.

Consider now step 9, the substitution test. The time to construct matrices C' and C'' is $\lesssim mn$. Define $n_i = rm_i$, $\nu_i = n_i + 1$, $l_i = m_i + n_i$, and $\lambda_i = l_i + 1$. Then these variables are defined as for Theorem 13, from which we see that the time to compute $C'W$ is $\lesssim T_{\text{MCPMPY}}(m, r, n - r, \mathbf{m}_s, \mathbf{n}_s)$, since $C' \in M(m, r, \mathbf{m}_s)$ and $W \in M(r, n - r, \mathbf{n}_s)$. Note that $\lambda_i \sim \mu_i + \nu_i \sim \mu_i + r\mu_i \sim r\mu_i \sim \nu_i$. By employing the above definitions, the following dominance or codominance relations can be found for the four terms of the dominating function in Theorem 13: (a) $\sim mr^{s+1}(n - r + 1)M_s$, (b) $\lesssim mr^{s+1}M_sM_s'$, (c) $\sim r^{s+2}(n - r + 1)M_sM_s'$, (d) $\sim mr^{s+1}(n - r + 1)M_sM_s'$. Clearly, (d) dominates the other terms, and so the time to compute $C'W$ is $\lesssim mr^{s+1}(n - r + 1)M_sM_s'$. It can be similarly shown, using Theorem 15, that this function also dominates the time to compute $d \cdot C''$. Since this function also dominates the times to construct C' and C'' and to compare $C'W$ and dC'', the time for step 9 is $\lesssim mr^{s+1}(n - r + 1)M_sM_s'$.

Dominance relations for the total times T_i for all executions of the individual steps can now be easily derived. These are summarized as follows (the dominating functions T_i' explicitly given):

$$\begin{aligned}
&T_1 \lesssim 1, && T_7 \lesssim r^{s+2}(n - r + 1)\mu_s M_s ,\\
&T_2 \lesssim 1, && T_8 \lesssim r^{s+2}(n - r + 1)\mu_s M_s ,\\
&T_3 \lesssim mnr\mu_s M_s , && T_9 \lesssim mr^{s+1}(n - r + 1)M_sM_s'.\\
&T_4 \lesssim r\mu_s \cdot T'_{\text{CPRRE}}(m, n, r, \mathbf{m}_{s-1}), && T_{10} \lesssim mn,\\
&T_5 \lesssim mr\mu_s , && T_{11} \lesssim r\mu_s .\\
&T_6 \lesssim nr, &&
\end{aligned}$$

It is easy to see from these relations that $T_1 + T_2 + T_5 + T_6 + T_{10} + T_{11} \lesssim T_3'$. Moreover, we have that $T_3' \lesssim T_9'$ since $n \lesssim r(n - r + 1)$ and $\mu_s \leq M_s'$. Also, $T_7 + T_8 \lesssim T_9'$. Thus $T_{\text{CPPRE}} \lesssim T_4' + T_9'$ for $s \geq 1$. If $s = 1$, then clearly

$$\begin{aligned}
T_{\text{CPRRE}} &\lesssim r\mu_1 \cdot mnr + mr^2(n - r + 1)\mu_1^2\\
&= mr^2\mu_1(n + (n - r + 1)\mu_1) = mr^2 M_1(n + (n - r + 1)M_1').
\end{aligned}$$

Assume inductively that the theorem holds for $s - 1$, $s \geq 2$. Then

$$\begin{aligned}
T_{\text{CPRRE}}(m, n, r, \mathbf{m}_s) &\lesssim r\mu_s \cdot T'_{\text{CPRRE}}(m, n, r, \mathbf{m}_{s-1}) + mr^{s+1}(n - r + 1)M_sM_s'\\
&\sim r\mu_s \cdot mr^s M_{s-1}(n + (n - r + 1)M'_{s-1}) + mr^{s+1}(n - r + 1)M_sM_s'\\
&= mr^{s+1}M_s(n + (n - r + 1)M'_{s-1} + (n - r + 1)M_s')\\
&\sim mr^{s+1}M_s(n + (n - r + 1)M_s'). \quad \square
\end{aligned}$$

Thus the time for all executions of the inner algorithm CRRE and the time for the substitution test dominate the times for all other operations.

Define $\mu = \max \mu_i$ (take $\mu = 1$ if $s = 0$).

COROLLARY 5. *For* $s \geq 0$, $T_{\text{CPRRE}} \lesssim mr^{s+1}\mu^s(n + (n - r + 1)\mu)$.

We now obtain maximum computing times for the outer algorithm PLES, making assumptions similar to those made in Theorem 18 for CPRRE. Let $T_{\text{PLES}}(m, n, q, r, c, \mathbf{m}_s)$ denote the maximum computing time of PLES, for C the $m \times n'$ augmented matrix (A, B) of rank r of a linear system $AX = B$, where $A \in M(m, n, P(c, \mathbf{m}_s))$ and $B \in M(m, q, P(c, \mathbf{m}_s))$.

THEOREM 19. *Under the assumptions that no rejected mod-p or evaluation homomorphisms are used and no premature equality test successes occur, the maximum computing time functions of PLES satisfy, for* $s \geq 0$,

$$T_{\text{PLES}} \lesssim mr^{s+2}(\log rc)\left(\prod_{i=1}^{s}\mu_i\right) \cdot\left[n' + (n' - r + 1)\left(\log rc + \sum_{i=1}^{s}\mu_i\right)\right] + n(n' - r + 1).$$

PROOF. The analysis of PLES is quite similar to that of CPRRE. Steps 1 and 9 are executed at most once, and as implied by the assumptions, steps 5 and 8 are each executed once. The remaining steps 2, 3, 4, 6, and 7 form a loop which is iterated t times, where t is the number of primes p_i required for successive iterates $C^{(t-1)}$ and $C^{(t)}$ to be equal and to satisfy the substitution test. If $C^{(t)}$ is not a DRRE form $\bar{C}_H$ for C, then continuation of the iterations would eventually yield $C^{(t')} = \bar{C}_H$, which is by hypothesis $\bar{C}$. Hence by Theorems 3 and 4 we have $\bar{C}_H \in M(m, n', P(r!\, c^r, rm_1, \cdots, rm_s))$, and by the Chinese Remainder Theorem we have $C^{(t)}$ in this set also. If α is the maximum of the magnitudes of the integer coefficients of the elements of $\bar{C}_H$, then $p_1 \cdots p_t \lesssim \alpha \leq r!\, c^r \leq (rc)^r$, and so $t \sim \log p_1 \cdots p_t \lesssim \log (rc)^r = r(\log rc)$. Thus the number of executions of steps 2, 3, 4, 6, and 7 is $\lesssim r(\log rc)$.

The following is a list of the dominating functions for the times for single executions of the steps: step 1, 1; step 2, $mn' \cdot (\log c)M_s$; step 3, $T_{\text{CPRRE}}(m, n', r, \mathbf{m}_s)$; step 4, m; step 5, $r(n' - r + 1)$; step 6, $r(n' - r + 1) \cdot r^{s+1}(\log rc)M_s$; step 7, $r(n' - r + 1) \cdot r^{s+1}(\log rc)M_s$; step 8, $mr^{s+2}(n' - r + 1)(\log rc)M_s(\log rc + M_s')$; step 9, $n(n' - r + 1)$. The verification of these computing times is very similar to that in Theorem 18 for CPRRE. We note in particular regarding step 6 that time to apply the Chinese Remainder Algorithm to a polynomial (or integer) is $\lesssim r(\log rc) \cdot \prod_{i=1}^{s} r\mu_s$, since each is in $P(r!\, c^r, rm_1, \cdots, rm_s)$, as discussed above. The times for the remaining steps can be easily verified, with the exception of the substitution test in step 8 which we now consider.

The time for step 8 is dominated by the time to compute $C'W$ and $d \cdot C''$. Consider first the computation of $C'W$. Defining $n_i = rm_i$, $\nu_i = n_i + 1$, $l_i = m_i + n_i$, and $\lambda_i = l_i + 1$, we have that the time to compute $C'W$ is $\lesssim T_{\text{MMPY}}(m, r, n' - r, c, \mathbf{m}_s, (rc)^r, \mathbf{n}_s)$. Referring to the dominating function of Theorem 14, the following dominating (or co-dominating) functions are obtained which correspond to the four terms within the brackets: (a) $\sim mr^{s+1}(n' - r + 1)M_s$, (b) $\lesssim mrM_s(\log c + r^s M_s')$, (c) $\sim r^{s+2}(n' - r + 1)M_s(\log rc + M_s')$, (d) $\sim mr^{s+1}(n' - r + 1)M_s(\log rc + M_s')$. Clearly, (d) dominates the other functions, and so by substituting into the bound of Theorem 14 and applying codominance, we find that the time to compute $C'W$ is $\lesssim mr^{s+2}(n' - r + 1) \cdot (\log rc)M_s(\log rc + M_s')$. Similarly, applying Theorem 16, we find that the time to compute $d \cdot C''$ is dominated by this same function.

A single dominance relation for PLES is now obtained from the total times for all the steps. It is easy to show that the times for steps 1, 4, and 5 are dominated by the time

for step 2. The times T_i for the remaining steps have the following dominating functions T_i':

$$
\begin{aligned}
T_2 &\lesssim mn'r(\log c)(\log rc)M_s = T_2', \\
T_3 &\lesssim mr^{s+2}(\log rc)M_s(n' + (n' - r + 1)M_s') = T_3', \\
T_6 &\lesssim r^{s+3}(n' - r + 1)(\log rc)^2 M_s = T_6', \\
T_7 &\lesssim r^{s+3}(n' - r + 1)(\log rc)^2 M_s = T_7', \\
T_8 &\lesssim mr^{s+2}(n' - r + 1)(\log rc)M_s(\log rc + M_s') = T_8', \\
T_9 &\lesssim n(n' - r + 1) = T_9'.
\end{aligned}
$$

Clearly, $T_6 + T_7 \lesssim T_8'$, and since $n' \lesssim r(n' - r + 1)$, also $T_2' \lesssim T_8'$. Thus $T_{\text{PLES}} \lesssim T_3' + T_8' + T_9' = mr^{s+2}(\log rc)M_s[n' + (n' - r + 1)M_s' + (n' - r + 1)(\log rc + M_s')] + n(n' - r + 1) \sim mr^{s+2}(\log rc)M_s[n' + (n' - r + 1)(\log rc + M_s')] + n(n' - r + 1)$. □

Thus, just as for CPRRE, the time for all applications of the inner algorithm CRRE and the time for the substitution test dominate the times for all other operations, except for the final construction of the general solution in step 9, a relatively simple operation. The following more concise but crude dominance relations are easily obtained for PLES.

COROLLARY 6. *The maximum computing time functions of PLES satisfy, for* $s \geq 0$,

$$T_{\text{PLES}} \lesssim mr^{s+2}\mu^s(\log rc)[n' + (n' - r + 1)(\log rc + \mu)] + n(n' - r + 1).$$

It is quite common to have m, n, and n' on the same order of magnitude and also to have rank r on the order of m and n. For example, in solving a single system of linear equations, we ordinarily have $n = m$ and $n' = n + q = m + 1$, and in matrix inversion $n = m$ and $n' = n + q = 2m$. Computing time bounds for these applications are contained in the following corollary, in which the above conditions are stated more formally.

COROLLARY 7. *If* $n - r \lesssim 1$ *and* $n' \sim n \sim m$, *then*

$$T_{\text{PLES}} \lesssim m^{s+3}\mu^s(\log mc)(m + q(\mu + \log mc)) \lesssim m^{s+4}\mu^s(\log mc)(\mu + \log mc).$$

Better bounds can, of course, be obtained for special cases; e.g. for single systems of linear equations, $T_{\text{PLES}} \lesssim m^{s+3}\mu^s(\log mc)(m + \mu + \log c)$. Moreover, only one prime need normally be applied in PLES to detect an inconsistent system, and in this case only steps 1, 2, 3, and 4 are executed, each once. Hence the dominating functions for this special case can be much smaller than those given in Theorem 19.

5. *Conclusion*

It has been asserted that the modular homomorphism algorithm is superior to the exact division algorithm for the solution of linear equations with integer or polynomial coefficients. A comparison of the computing time dominating functions will reveal this. Suppose first that the classical algorithms for polynomial multiplication and division are used. Then, for the exact division algorithm (ED), the maximum computing time functions can be shown to satisfy $T_{\text{ED}} \lesssim m^{2s+4}(m + q)\mu^{2s}(\log mc)^2 = T_{\text{ED}}'$ under the same assumptions as in Corollary 7 (giving dominating function T_{PLES}'). In fact, it is quite plausible that $T_{\text{ED}} \sim T_{\text{ED}}'$ and also that T_{ED}' is codominant with the average computing time of ED (assuming each $\mu_i = \mu$). Granting these conjectures, the ratio

$$T_{\text{ED}}'/T_{\text{PLES}}' \tilde{=} m^{s+1}\mu^{s-1} \cdot [(m + q)\mu(\log mc)/(m + q(\mu + \log mc))] \qquad (16)$$

is a good measure of the comparative efficiency of these algorithms. Thus for reasonably large integers (i.e. $\geq \beta$, the radix base of the number representation), this ratio will be at least m for $s = 0$, and at least $m^{s+1}\mu^{s-1}$ for $s \geq 1$. In fact, it may be quite a bit larger, since for large problems the quantity in brackets in (16) is much larger than 1.

Now suppose that the modular homomorphism algorithms are used for polynomial multiplication and division, as given in [14], in the exact division algorithm (ED'). Considering only the multiplications used in the forward elimination, the term con-

tributed to the dominating function for exact division by the multiplications is $m^{s+4}(m + q)\mu^s(\log mc)(\log mc + \mu)$. Since the method of analysis giving this function is analogous to that for obtaining T'_{PLES}, the dominating functions obtained for ED′ satisfy

$$T'_{\text{ED}'}/T'_{\text{PLES}} \gtrsim m\cdot[(m + q)(\mu + \log mc)/(m + q(\mu + \log mc))]. \qquad (17)$$

Thus, as a conservative measure of relative efficiency, we find that the ratio in (17) is likely greater than m and, in fact, much greater than m for m large relative to q. Thus the superiority of the modular algorithm over the exact division algorithm is clear.

Suppose that we desire to reduce the particular solution (d, Y) computed by PLES to rational form. Then mq GCD's of pairs of polynomials (or integers) in $P((mc)^m, mm_1, \cdots, mm_s)$ must be computed. Using a modular algorithm, the maximum computing time to compute each GCD is $\lesssim [m(\log mc)]^2$ if $s = 0$, and $\lesssim m^{s+2}\mu^s(\log mc)(\mu + \log mc)$ if $s \geq 1$ (see [8] or [14]). The same assumptions concerning rejected homomorphisms and the equality test are made. So the time for the reduction to rational form is $\lesssim m^3q(\log mc)^2$ if $s = 0$, and $\lesssim m^{s+3}q\mu^s(\log mc)(\mu + \log mc)$ if $s \geq 1$. Clearly, these times are $\lesssim T'_{\text{PLES}}$ for $s \geq 0$. Thus the time for this reduction is dominated by the time to compute the solution by PLES.

We note that a modular homomorphism algorithm for computing determinants, which is analogous to PLES, has been developed [24] and is likewise an improvement on older methods such as the exact division method.

ACKNOWLEDGMENT. The author is indebted to Professor George E. Collins for having brought some of the earlier work to the author's attention and for his sustained interest and many valuable suggestions during the course of this research. The comments of the editor, Professor B. F. Caviness, are likewise appreciated.

REFERENCES

1. Bareiss, E. H. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comp. 22*, 103 (July 1968), 565–578.
2. Birkhoff, G., and MacLane, S. *A Survey of Modern Algebra*, 3rd Ed., Macmillan, New York, 1965.
3. Bodwig, E. *Matrix Calculus*. Interscience, New York, 1959.
4. Borosh, I., and Frankel, A. S. Exact solutions of linear equations with rational coefficients by congruence techniques. *Math. Comp. 20*, 93 (Jan. 1966), 107–112.
5. Fraenkel, A. S., and Loewenthal, D. Exact solutions of linear equations with rational coefficients. *J. Res. NBS 75B*, 1, 2 (Jan.–June, 1971), 67–75.
6. Brown, W. S., Hyde, J. P., and Tague, B. A. The ALPAK system for nonnumerical algebra on a digital computer—III: Systems of linear equations and a class of side relations. *Bell Syst. Tech. J. 43*, 2 (July 1964), 1547–1562.
7. Brown, W. S. The compleat Euclidean algorithm. Bell Telephone Labs. Rep., Murray Hill, N.J., June 1968.
8. Brown, W. S. On Euclid's algorithm and the computation of polynomial greatest common divisors. Proc. 2nd Symp. on Symbolic and Algebraic Manipulation, ACM, New York, 1971; also, *J. ACM 18*, 4 (Oct. 1971), 478–504.
9. Collins, G. E. Subresultants and reduced polynomial remainder sequences. *J. ACM 14*, 1 (Jan. 1967), 128–142.
10. Collins, G. E. Computing time analyses for some arithmetic and algebraic algorithms. Proc. 1968 Summer Inst. on Symbolic Mathematical Computation, R. Tobey, Ed., IBM Federal Systems Ctr., June 1969, pp. 195–232.
11. Collins, G. E., Heindel, L. E., Horowitz, E., McClellan, M. T., and Musser, D. R. The SAC-1 modular arithmetic system. Computing Ctr. Tech. Rep. 10, U. of Wisconsin, Madison, Wisc., June 1969.
12. Collins, G. E. The SAC-1 system: An introduction and survey. Proc. 2nd Symp. on Symbolic and Algebraic Manipulation, ACM, New York, 1971, pp. 144–152.
13. Collins, G. E. The calculation of multivariate polynomial resultants. Proc. 2nd Symp. on Symbolic and Algebraic Manipulation, ACM, New York, 1971; also, *J. ACM 18*, 4 (Oct. 1971), 515–532.

14. COLLINS, G. E. The SAC-1 polynomial greatest common divisor and resultant system. Computer Sci. Dep. Tech. Rep. 145, U. of Wisconsin, Madison, Wisc., Feb. 1972.
15. COLLINS, G. E., AND MCCLELLAN, M. T. The SAC-1 polynomial linear algebra system. Computer Sci. Dep. Tech. Rep. 154, U. of Wisconsin, Madison, Wisc., Apr. 1972.
16. FOX, L. *An Introduction to Numerical Linear Algebra*. Clarendon Press, Oxford, 1964.
17. GANTMACHER, F. R. *Matrix Theory, Vol. 1*. Chelsea, New York, 1959.
18. HOWELL, J. A., AND GREGORY, R. T. An algorithm for solving linear algebraic equations using residue arithmetic. *BIT 9* (1969), 200–234, 324–337.
19. KNUTH, D. E. *The Art of Computer Programming, Vol. 1 (Fundamental Algorithms)*. Addison-Wesley, Reading, Mass., 1968.
20. KNUTH, D. E. *The Art of Computer Programming, Vol. 2 (Seminumerical Algorithms)*. Addison-Wesley, Reading, Mass., 1969.
21. LIPSON, J. D. Symbolic methods for the computer solution of linear equations with applications to flow-graphs. Proc. 1968 Summer Inst. on Symbolic Mathematical Computation, Robert Tobey, Ed., IBM Federal Systems Ctr., June 1969, pp. 233–303.
22. LIPSON, J. D. Chinese Remainder and interpolation algorithms. Proc. 2nd Symposium on Symbolic and Algebraic Manipulation, *ACM*, New York, 1971, pp. 372–398.
23. LUTHER, H. A., AND GUSEMAN, L. F. JR. A finite sequentially compact process for the adjoints of matrices over arbitrary integral domains. *Comm. ACM 5*, 8 (Aug. 1962), 447–448.
24. MCCLELLAN, M. T. The exact solution of systems of linear equations with polynomial coefficients. Computer Sci. Dep. Tech. Rep. 136 (Ph.D. Th.), U. of Wisconsin, Madison, Wisc., Sept. 1971.
25. NEWMAN, M. Solving equations exactly. *J. Res. NBS 71B*, 4 (Oct.–Dec. 1967), 171–179.
26. RALSTON, A. *A First Course in Numerical Analysis*. McGraw-Hill, New York, 1965.
27. ROSSER, J. B. A method of computing exact inverses of matrices with integer coefficients. *J. Res. NBS 49*, 5 (Nov. 1952), 349–358.
28. TAKAHASI, H., AND ISHIBASHI, Y. A new method for "exact calculation" by digital computer. *Information Processing in Japan 1* (1961), 28–42.

RECEIVED DECEMBER 1970; REVISED DECEMBER 1972