WIS-CS-154-72

Computer Sciences Department
The University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

Received May 23, 1973 - Rerun May 1973 with revisions.

THE SAC-1 POLYNOMIAL LINEAR ALGEBRA SYSTEM*

by

G. E. Collins and M. T. McClellan†

Technical Report No. 154 +

April 1972

*Research supported by NSF grants GJ-239 and GJ-30125X and by NASA grant NGL-21-002-008.

† Present affiliation: Computer Science Center, University of Maryland.

+This report also appears as University of Wisconsin Madison Academic Computing Center Technical Report No. 28 .

ERRATA

The SAC-1 Polynomial Linear Algebra System

Page 81, line -17.(second line of CDET). Change "V" to "U".

Page 23, line 8. Change to "(1)[Initialize.] If A=0, return; B←FIRST(A); t←-1; if B≠0, t←TYPE(B).".

Page 23, line 10. After "DECAP(B,A)" insert "; if t<0, go to (2)".

Page 90, 4th line of MCPERS. Replace "1 T=TYPE(FIRST(A))" with:

```
1  IF (A.EQ.0) RETURN
   B=FIRST(A)
   T=-1
   IF (B.NE.0) T=TYPE(B)
```

Page 90, line 10. After "21 CALL DECAP(B,A)" insert:

```
   IF (T.LT.0) GO TO 2
```

July 8, 1974

George E. Collins

TABLE OF CONTENTS

ABSTRACT

This system is the tenth in a series of subsystems comprising the SAC-1 System for Symbolic and Algebraic Calculation. The present subsystem consists of programs implementing modular algorithms for linear equations solution, matrix inversion, determinant calculation, null space basis generation, and matrix multiplication, all for matrices with integer or polynomial entries. For each program in the system is given a functional specification, an algorithm description, an analytical computing time, and a Fortran program listing. Empirically observed computing times for some of the key programs are presented. Also, a test program is supplied as an aid in implementing the system and to illustrate its use.

1. INTRODUCTION

The SAC-1 Polynomial Linear Algebra System is a subsystem of the SAC-1 (Symbolic and Algebraic Calculation - Version 1) System. This subsystem is designed to perform exactly a number of those calculations which are basic to linear algebra on matrices of multivariate polynomials with infinite-precision integer coefficients or, as a special case, matrices of infinite-precision integer coefficients. The main capability is the solution of systems of linear equations with either integer or polynomial coefficients, which conveniently includes matrix inversion. However, several other computing abilities are available such as matrix arithmetic, determinant calculation, and null space basis generation.

Let us consider the relation of this subsystem to the SAC-1 System as a whole. The SAC-1 System is a set of hierarchically organized modules, each consisting for the most part of Fortran Subprograms (with possibly a few machine-coded primitive routines), which conform to USASI specifications, [1]. Almost all SAC-1 capabilities required by the Polynomial Linear Algebra System may be found in the following SAC-1 Subsystems: the List Processing System, [2], the Infinite-Precision Integer Arithmetic System, [3], the Polynomial System, [4], and the Modular Arithmetic System, [6]. Only

three additional routines are required: one (PNORMF) from the Polynomial Real Zero System, [8], and two (CPNV and IFACT) from the Polynomial GCD and Resultant System, [9]. Other subsystems in existence are: the Rational Function System, [5], the Partial Fraction Decomposition and Rational Function Integration System [7], and the Polynomial Factorization System, [10].

The documentation provided in the remainder of this manual for the Polynomial Linear Algebra System is as follows. In the remainder of this section basic definitions and results are presented and the data structures and related conventions which are peculiar to this subsystem are described. In Section 2 each algorithm is explicitly specified and is accompanied by its analytically derived computing time. For some of the main algorithms empirically observed computing times are given in Section 3. Section 4 contains a test program together with the input and output for a sample run. This test program can be used to test a typical implementation of the SAC-1 Polynomial Linear Algebra System. The Fortran subprograms corresponding to the algorithms of Section 2 are listed in alphabetical order in Section 5. A thorough treatment of the theory underlying the algorithms may be found in [11] along with further explanation of the algorithms and derivations of the computing times.

We now describe the computational problems treated and the

new SAC-1 data structures involved. As stated above, the primary capability of the Polynomial Linear Algebra System is the exact solution of the matrix equation:

$$AX = B \qquad (1)$$

where A is m by n and nonzero, B is m by q, and both A and B are over $I[x_1,\ldots,x_s]$, $s \geq 0$, the ring of polynomials in s variables over the integers ($s = 0$ corresponding to the case of integer entries). Such a system may 1) have no solution, 2) have a unique solution, or 3) have infinitely many solutions. In cases 2) and 3) a <u>particular solution</u> is computed as a pair (d,Y), where d is in $I[x_1,\ldots,x_s]$ and Y is an n by q matrix over $I[x_1,\ldots,x_s]$ such that $AY = d \cdot B$. In case 3) the null space of A is nontrivial and an infinity of solutions can be obtained in the form (d,Y+W), where each W is an n by q matrix over $I[x_1,\ldots,x_s]$ such that $AW = 0$.

What is actually computed in case 3) is an n by n-r matrix Z over $I[x_1,\ldots,x_s]$, whose columns form a basis for the null space of A (i.e. the columns are linearly independent and $AZ = 0$). Such a matrix Z will be called a <u>null space basis matrix</u> for A and the triple (d,Y,Z) will be referred to as a <u>general solution</u> of the matrix equation (1). If the system has a unique solution, the null space is trivial and we take the value of Z to be 0.

The computation of a general solution to (1) is intimately connected to general elimination algorithms. Such an algorithm, when restricted to elementary row operations and applied to an m by n' nonzero matrix C of rank r, will compute a sequence of pivot columns $1 \leq j_1 < j_2 < \cdots < j_r \leq n'$ and a transformed matrix F satisfying: for $1 \leq i \leq r$, the i^{th} row of F has $F(i,j_i) \neq 0$ and $F(i,j) = 0$, for $1 \leq j < j_i$ and $j = j_u$, $i < u \leq r$, and for $r < i \leq m$, row i is zero. A matrix with this general structure is called a reduced row echelon (RRE) matrix and, if in addition it is row equivalent to some matrix C, it is called a reduced row echelon (RRE) form for C. The sequence $J = (j_1, j_2, \cdots, j_r)$ is unique for C and is called the row echelon (RE) sequence for C and denoted by J_C. The elements $F(i,j_i)$, $1 \leq i \leq r$, constitute the diagonal of F.

In the case that C is a matrix over an integral domain with a division algorithm, C can be transformed to an RRE form F using only arithmetic operations in the integral domain. The best known method of this kind is the exact division algorithm (see [11]). When in this algorithm the pivot searches and row interchanges are performed in a certain way, a special RRE form $F = \overline{\overline{C}}$, called the determinantal RRE form for C, is obtained. In particular, if $J_C = (j_1, \cdots, j_r)$, then, for $1 \leq i \leq r$, $\overline{\overline{C}}(i,j)$ is the minor of C defined from the elements common to rows $i_1, \cdots, i_r$ and columns $j_1, \cdots, j_{i-1}, j, j_{i+1}, \cdots, j_r$ in that order.

Here $I_C = (i_1,\ldots,i_m)$ is a special permutation of $1,2,\ldots,m$, which is uniquely defined for each matrix C and which may be computed using the row permutations employed in the elimination (see [11]). When C is over a field, $\overline{\overline{C}}$ can be computed by a Gaussian elimination algorithm, provided that the same method of pivot searches and row interchanges is used as in the exact division algorithm. This algorithm is used in subsection 2.5 for the finite field $GF(p)$ with a prime number p of elements.

Consider now the construction of a general solution from an RRE form of a matrix. Let E be any m by n nonzero RRE matrix with common diagonal value d and let $J_E = (j_1,\ldots,j_r)$, where rank $r < n$. Let $1 \le h_1 < h_2 < \cdots < h_{n-r}$ be the sequence of integers which complements $j_1,j_2,\ldots,j_r$ with respect to $1,2,\ldots,n$. Then an n by $n-r$ matrix Z can be defined from E by

$$Z(j_i,k) = \left\{ \begin{array}{ll} E(i,h_k), & \text{if } j_i < h_k \\ 0, & \text{otherwise} \end{array} \right\} ,$$

$$Z(h_k,j) = \left\{ \begin{array}{ll} -d, & \text{if } j = k \\ 0, & \text{otherwise} \end{array} \right\} . \qquad (2)$$

Several theorems concerned with computing general solutions to systems of linear equations will now be stated.

Theorem 1. Let E be an m by n nonzero RRE matrix with $J_E = (j_1,\ldots,j_r)$, $r < n$, and common diagonal value d. If Z is the n by $n-r$ matrix defined by (2), then Z is a null space basis for E.

This leads immediately to the following result:

Theorem 2. Let C be an m by n nonzero matrix with rank $r < n$. If $E = \overline{\overline{C}}$ and Z is defined by (2), then Z is a null space basis for C.

The next result provides a sufficient condition for deciding if the RE sequence of some RRE matrix is the RE sequence of another matrix.

Theorem 3. Let C be an m by n nonzero matrix of rank $r < n$ and let E be an m by 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 as in (1). Then, if $CZ = 0$, it follows that $J_C = J_E$.

This condition will be applied directly in the algorithms of Section 2.5. However, it is also required by the following theorem, which applies the previous results to the computation of general solutions.

Theorem 4. Let $C = (A,B)$ be the augmented matrix of a consistent linear system $AX = B$, where A is m by n, nonzero, and of rank $\hat{r}$ and B is m by q. Let E be an m by n' RRE matrix, $n' = n+q$, with common diagonal value d and rank r, and suppose $r \leq \hat{r}$ and $J_E \leq J_C$. Also, let Z' be the n' by $n'-r$ matrix constructed from E according to (2), let Z consist of the first n rows and first $n-r$ columns of Z', and let Y consist of the first n rows and last q columns of Z'. Then, if $CZ' = 0$, the triple (d,Y,Z) is a general solution of $AX = B$.

This theorem implies that, if $E = \bar{\bar{C}}$, then $CZ' = 0$ by Theorem 2 and, letting $\hat{d}$ be the common diagonal value of $\bar{\bar{C}}$, the triple $(\hat{d},Y,Z)$ constitutes a general solution, called the determinantal general solution of $AX = B$. It is in fact true that with high probability the modular algorithm given in subsection 2.5 computes the determinantal general solution, when a solution exists.

We now describe the additional SAC-1 data structures and conventions required for the Polynomial Linear Algebra System. Several mathematical objects are represented as single-precision integers (i.e. as atoms), that is, as integers less than γ in magnitude, the maximum Fortran integer plus one. The integer quantities in this class are: the radix β for the infinite-precision integer (L-integer) representation, prime numbers p when used to

specify a finite field GF(p), the elements of GF(p), the degrees and degree bounds for polynomials, the dimensions of matrices, and hence the elements of RE sequences J and of permutations I of initial segments of the positive integers.

The representations of those objects requiring list structures are now described. A non-null sequence $H = (h_1,\ldots,h_k)$ of Fortran integers is represented as the first-order list $(h_1,\ldots,h_k)$. The null sequence is represented by the null list ( ). Let M(m,n,S) denote the set of m by n matrices over a set S and assume the elements of S have a SAC-1 data representation. A matrix A in M(m,n,S) is represented <u>rowwise</u> by the list $(A_1,\ldots,A_m)$, where A_i is the list $(A_{i,1},\ldots,A_{i,n})$. Thus, if S = GF(p), then $A_{i,j}$ is the atom representing A(i,j) and if S is the ring of integers I, or $I[x_1,\ldots,x_s]$ or $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, or a subset of one of these sets, then $A_{i,j}$ is the list representing the integer or polynomial A(i,j). Here $GF(p)[x_1,\ldots,x_s]$ denotes the domain of polynomials in s variables over GF(p). The <u>columnwise</u> representation is defined similarly as the list $(A_1,\ldots,A_n)$, where A_j is the list $(A_{1,j},\ldots,A_{m,j})$ representing column j. The rowwise representation will ordinarily be used and exceptions will be properly noted.

The case of a nonzero RRE matrix F with common diagonal value d presents an exception to the above representation. In the

interests of efficiency, the known zero elements are discarded from the list representation, resulting in a special list representation. Let k be the least non-negative integer such that $j_{i+1} = j_i + 1$, for $k < i \leq r$, where $J_F = (j_1,\ldots,j_r)$ and we take $j_{r+1} = n+1$ and $j_0 = 0$. If $k = 0$, the only nonzero elements of F are the diagonal elements. Otherwise, $0 < k \leq r$ and each of rows $1,2,\ldots,k$ of F have non-diagonal elements which are possibly nonzero; for row i, these are $F(i,j)$, $j_i < j \leq n$ and $j \neq j_u$, $i < u \leq r$. We then define the list V as follows. If $k = 0$, V is the null list $(\,)$. If $k > 0$, V is the list $(V_1,\ldots,V_k)$, where V_i is the list of possibly nonzero non-diagonal elements of row i of F, as given above. Clearly, F can be constructed from J_F, d, and V; hence, the triple (J_F,d,V) is referred to as the <u>compact representation</u> of the RRE matrix F. The list V is called the <u>non-diagonal part</u> of the compact representation of F.

In order to more compactly represent the analytically derived computing times which will accompany the algorithms of Section 2, the following useful notation for certain subsets of $I[x_1,\ldots,x_s]$, $s \geq 0$, and $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, is employed. Let A be in $I[x_1,\ldots,x_s]$, $s \geq 0$, and define norm(A) to be $|A|$, if $s = 0$, or the sum of the absolute values of the integer coefficients of A, if $s > 0$. We note that norm satisfies the following important properties:

1) if $A \neq 0$, $\text{norm}(A) > 0$, 2) $\text{norm}(A+B) \leq \text{norm}(A) + \text{norm}(B)$, and 3) $\text{norm}(AB) \leq \text{norm}(A) \cdot \text{norm}(B)$. For A in $I[x_1,\ldots,x_s]$ or in $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, let $\deg_i(A)$ denote the degree of A in x_i. Then let $P(d)$ denote the set of all integers a such that $|a| \leq d$ and let $P(d,m_1,\ldots,m_s)$ denote the set of all polynomials A in $I[x_1,\ldots,x_s]$ such that $\text{norm}(A) \leq d$ and $\deg_i(A) \leq m_i$. By convention we let $P(d,m_1,\ldots,m_s)$ denote $P(d)$ for the case $s = 0$. Similarly, let $P^*(m_1,\ldots,m_s)$ denote the set of all polynomials A in $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, such that $\deg_i(A) \leq m_i$, where we let $P^*(m_1,\ldots,m_s)$ denote $GF(p)$ for $s = 0$. Thus, $M(m,n,P(d,m_1,\ldots,m_s))$ is the set of all m by n matrices whose polynomial or integer entries have norms of at most d and, if they are polynomials, have degrees in x_i of at most m_i. Similarly for $M(m,n,P^*(m_1,\ldots,m_s))$.

Another useful notation for expressing computing times concerns the length of an integer. For any integer a, define

$$L_\beta(a) = \begin{cases} 1, & \text{if } a = 0 \\ \lceil \log_\beta(|a|+1) \rceil, & \text{if } a \neq 0. \end{cases}$$

Then $L_\beta(a)$ is the number of digits in the β-radix representation of a, for $a \neq 0$. We note that $L_\beta(a) \sim L_\delta(a)$, for any two fixed bases, β and δ. It is then appropriate to omit the base, writing $L(a)$, which may be referred to as the <u>length</u> of a. This particular formulation of length is due to D. R. Musser, [12].

2. ALGORITHM DESCRIPTIONS

In this section the algorithms comprising the SAC-1 Linear Algebra System are presented. The format employed for each consists in 1) a functional specification, 2) the explicit steps of the algorithm, and 3) analytically derived computing times, stated in terms of dominance and codominance relations. The algorithms are grouped by subsection as follows. In 2.1 classical algorithms for matrix arithmetic (sum, difference, and product) are given. In 2.2 algorithms for applying mod-p and evaluation homomorphisms elementwise to matrices and for applying the Chinese Remainder Algorithm and incremental interpolation elementwise to matrices are presented. These auxiliary algorithms are employed in the modular homomorphism algorithms for matrix multiplication, given in 2.3, for the solution of systems of linear equations, given in 2.5, and for determinant calculation, given in 2.6. The latter two algorithms will also require additional auxiliary algorithms; these are found in 2.4. Finally, in 2.7 algorithms for matrix inversion and null space basis generation are given.

2.1 Classical Matrix Arithmetic Algorithms

C = MSUM(A,B)

A and B are m by n matrices over $I[x_1,\ldots,x_s]$, $s \geq 0$. C = A + B.

Algorithm

(1) [Initialize.] X ← A; Y ← B; C ← ( ).

(2) [Begin next row D.] ADV(E,X); ADV(F,Y); D ← ( ).

(3) [Compute and prefix next element of row D.] D ← PFL(PSUM(FIRST(E),FIRST(F)),D); E ← TAIL(E); F ← TAIL(F); if E ≠ ( ), go to (3).

(4) [Prefix next row of sum matrix.] C ← PFL(INV(D),C); if X ≠ ( ), go to (2).

(5) [Return.] C ← INV(C); return C.

Computing time: $\lesssim mn(L(d) + L(e)) \cdot (\Pi_{i=1}^{s}\mu_i + \Pi_{i=1}^{s}\nu_i)$, for $A \in M(m,n,P(d,m_1,\ldots,m_s))$, $B \in M(m,n,P(e,n_1,\ldots,n_s))$, $\mu_i = m_i + 1$, and $\nu_i = n_i + 1$.

C = MDIF(A,B)

A and B are m by n matrices over $I[x_1,\ldots,x_s]$, $s \geq 0$. C = A - B.

Algorithm

(1) [Initialize.] X ← A; Y ← B; C ← ( ).

(2) [Begin next row D.] ADV(E,X); ADV(F,Y); D ← ( ).

(3) [Compute and prefix next element of row D.] D ← PFL(PDIF(FIRST(E),FIRST(F)),D); E ← TAIL(E); F ← TAIL(F); if E ≠ ( ), go to (3).

(4) [Prefix next row of difference matrix.] C ← PFL(INV(D),C); if X ≠ ( ), go to (2).

(5) [Return.] C ← INV(C); return C.

Computing time: Same as MSUM.

B = MTRAN(A)

A is an m by n matrix over $I[x_1,\ldots,x_s]$ or $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$. B is the n by m matrix transpose of A: B(i,j) = A(j,i).

Algorithm

(1) [Initialize.] G ← CINV(A); F ← G; C ← FIRST(F); t ← TYPE(C); n ← LENGTH(C); construct the list B = (( ), ( ),...,( )) of length n.

(2) [Begin next column of transpose.] ADV(C,F); S ← B.

(3) [Prefix next element of current column.] ADV(D,C); if t = 0, ALTER(PFA(D,FIRST(S)),S); if $t \neq 0$, ALTER(PFL(BORROW(D), FIRST(S)),S); S ← TAIL(S); if $S \neq ( )$, go to (3); if $F \neq ( )$, go to (2).

(4) [Return.] Erase G; return B.

Computing time: $\sim mn$.

C = MPROD(A,B)

Matrices A and B over $I[x_1,\ldots,x_s]$, $s \geq 0$, where A is m by n and B is n by q. C = AB, the m by q matrix product of A and B.

Algorithm

(1) [Initialize.] X ← CINV(A); Y ← INV(MTRAN(B)); C ← ( ); V ← X.

(2) [Begin next row of product C.] ADV(R,V); W ← Y; D ← ( ).

(3) [Begin next element of current row.] S ← R; ADV(T,W); E ← ( ).

(4) [Add next term of inner product.] F ← PPROD(FIRST(S), FIRST(T)); G ← PSUM(E,F); erase E and F; E ← G; S ← TAIL(S); T ← TAIL(T); if S ≠ ( ), go to (4).

(5) [Prefix next element of current row.] D ← PFL(E,D); if W ≠ ( ), go to (3).

(6) [Prefix next row of product C.] C ← PFL(D,C); if V ≠ ( ), go to (2).

(7) [Return.] Erase X and Y; return C.

Computing time: $\lesssim mnq \cdot L(d)L(e) \cdot (\Pi_{i=1}^{s}\mu_i)(\Pi_{i=1}^{s}\nu_i) + mnq \cdot L(n) \cdot (\Pi_{i=1}^{s}\kappa_i)$, where $A \in M(m,n,P(d,m_1,\ldots,m_s))$, $B \in M(n,q,P(e,n_1,\ldots,n_s))$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, and $\kappa_i = \mu_i + \nu_i$.

2.2 Applying Mod-p and Evaluation Homomorphisms, the Chinese Remainder Algorithm and Interpolation Elementwise to Matrices

Nine relatively simple algorithms are given, all required by the modular homomorphism algorithms in subsections 2.3, 2.5, and 2.6. The first tests for a zero matrix.

k = MZERO(A)

An m by n matrix A over $I[x_1,\ldots,x_s]$ or $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$. k = 0, if A is nonzero, and k = 1, if A is zero.

Algorithm

(1) [Initialize.] $X \leftarrow A$; $k \leftarrow 0$.

(2) [Obtain next row.] ADV(Y,X).

(3) [Test next row element.] If FIRST(Y) $\neq 0$, return k;
$Y \leftarrow$ TAIL(Y); if $Y \neq ( )$, go to (3); if $X \neq ()$, go to (2);
$k \leftarrow 1$; return k.

Computing time: the maximum computing time is $\sim mn$.

The next four algorithms concern the application of mod-p homomorphisms and Garner's variant of the Chinese Remainder Algorithm elementwise to matrices.

L = MVLIST(A)

A is an m by n nonzero matrix over $I[x_1,\ldots,x_s]$, $s \geq 0$. If $s = 0$, L is the null list ( ) and, if $s \geq 1$, L is the list of variables $(x_1,\ldots,x_s)$.

Algorithm

(1) [Initialize.] $X \leftarrow A$.

(2) [Get next row.] ADV(B,X).

(3) [Get next row element.] ADV(C,B); if $C \neq ( )$, go to (4); if $B \neq ( )$, go to (3); go to (2).

(4) [Compute and return variable list.] $L \leftarrow$ PVLIST(C); return L.

Computing time: the maximum computing time is $\sim mn$.

MERASE(A)

A is an m by n matrix over $I[x_1,\ldots,x_s]$, $s \geq 0$. A is erased. The variable A is redefined as the location of the list representation of the first row which overlaps another list. The usual result, however, is that all rows are erased and A = 0 is returned.

Algorithm

(1) [Begin erasure of next row.] If A = ( ), return; k ← COUNT(A) - 1; if k > 0, (SCOUNT(k,A); return); DECAP(B,A).

(2) [Erase next row element.] k ← COUNT(B) - 1; if k > 0, (SCOUNT(k,B); go to (1)); DECAP(C,B); PERASE(C); if B ≠ ( ), go to (2); go to (1).

Computing time: $\lesssim mn \cdot L(d) \cdot (\Pi_{i=1}^{s} \mu_i)$, where $A \in M(m,n,P(d,m_1,\ldots,m_s))$ and $\mu_i = m_i + 1$.

B = MMOD(p,A,s)

A is an m by n matrix over $I[x_1,\ldots,x_s]$, $s \geq 0$, and p is a prime number. Both p and s are Fortran integers. B is the m by n matrix over $GF(p)[x_1,\ldots,x_s]$ such that corresponding elements of A and B satisfy: $B(i,j) = \varphi_p(A(i,j))$.

Algorithm

(1) [Initialize.] $X \leftarrow A$; $B \leftarrow ( )$.

(2) [Begin next row.] ADV(T,X); $C \leftarrow ( )$.

(3) [Apply φ_p to next row element.] ADV(R,T); $R \leftarrow$ CPMOD(p,R); if $s = 0$, $C \leftarrow$ PFA(R,C); if $s \neq 0$, $C \leftarrow$ PFL(R,C); if $T \neq ( )$, go to (3).

(4) [Prefix next row.] $B \leftarrow$ PFL(INV(C),B); if $X \neq ( )$, go to (2).

(5) [Return,] $B \leftarrow$ INV(B); return B.

Computing time: $\lesssim mn \cdot L(d) \cdot (\Pi_{i=1}^{s} \mu_i)$, where $A \in M(m,n,P(d,m_1,\ldots,m_s))$ and $\mu_i = m_i + 1$.

C = MGARN(Q,B,p,A,L)

p is an odd prime number and Q is an odd infinite-precision integer relatively prime to p. B is an m by n matrix over $I[x_1,\ldots,x_s]$ and A is an m by n matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$. L is the null list, if $s = 0$, or is the list $(x_1,\ldots,x_s)$ of variables, if $s \geq 1$. The integer coefficients of B are less than Q/2 in magnitude. C is the m by n matrix over $I[x_1,\ldots,x_s]$ such that each element is that unique polynomial or integer such that $C(i,j) \equiv B(i,j) \pmod{Q}$, $C(i,j) \equiv A(i,j) \pmod{p}$, and the integer coefficients of C(i,j) are each less than Qp/2 in magnitude.

Note that the case Q = 1 is allowed, in which case the elements of C have integer coefficients less than p/2 in magnitude.

Algorithm

(1) [Initialize.] X ← B; Y ← A; C ← ( ).

(2) [Begin next row of C.] ADV(S,X); ADV(T,Y); D ← ( ).

(3) [Compute next row element by Garner's Method.] ADV(U,S); ADV(V,T); E ← CPGARN(Q,Y,p,V,L); D ← PFL(E,D); if $S \neq ( )$, go to (3).

(4) [Prefix next row.] C ← PFL(INV(D),C); if $X \neq ( )$, go to (2).

(5) [Return.] C ← INV(C): return C.

Computing time: the maximum computing time is $\sim mn \cdot L(Q) \cdot (\Pi_{i=1}^{s} \mu_i)$, for $A \in M(m,n,P^*(m_1,\ldots,m_s))$, B an m by n matrix over $I[x_1,\ldots,x_s]$, each of whose elements have degree in x_i at most m_i and whose integer coefficients are less than $Q/2$ in magnitude, and $\mu_i = m_i + 1$.

The final four algorithms are concerned with the application of evaluation homomorphisms and interpolation elementwise to matrices of polynomials over GF(p).

s = MCPNV(A)

A is an m by n nonzero matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$.

s is the number of variables, a Fortran integer.

Algorithm

(1) [Initialize.] B ← A; if TYPE(FIRST(B)) = 0, (s ← 0; return s).

(2) [Begin next row search.] ADV(C,B).

(3) [Obtain and test next row element.] ADV(D,C); if $D \neq 0$, go to (4); if $C \neq (\,)$, go to (3); go to (2).

(4) [Compute and return variable count.] s ← CPNV(D); return s.

Computing time: the maximum computing time is $\sim mn$, if $s \geq 1$, and is ~ 1, if $s = 0$.

MCPERS(A)

A is an m by n matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$. A is erased. The variable A is redefined as the location of the list representation of the first row which overlaps another list. The usual result, however, is that all rows are erased and A = 0 is returned.

Algorithm

(1) [Initialize.] t ← TYPE(FIRST(A)).

(2) [Begin erasure of next row.] If A = ( ), return; k ← COUNT(A) - 1; if k > 0, (SCOUNT(k,A); return); DECAP(B,A).

(3) [Erase next row element.] k ← COUNT(B) - 1; if k > 0, (SCOUNT(k,B); go to (2)); DECAP(C,B); if t = 1, CPERAS(C); if B ≠ ( ), go to (3); go to (2).

Computing time: $\lesssim mn(\Pi_{i=1}^{s}\mu_i)$, for $A \in M(m,n,P^*(m_1,\ldots,m_s))$ and $\mu_i = m_i + 1$.

C = MCPEVL(p,A,b,s)

p is a prime number, b is an element of GF(p), and A is an m by n matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$. p,b, and s are Fortran integers. C is the m by n matrix over $GF(p)[x_1,\ldots,x_{s-1}]$ such that each element satisfies: $C(i,j) = A(i,j)(x_1,\ldots,x_{s-1},b) = \psi_b(A(i,j))$.

Algorithm

(1) [Initialize.] X←A; C←( ).

(2) [Begin next row.] ADV(W,X); D←( ).

(3) [Compute and prefix next row element.] ADV(U,W); if $U \neq ( )$, U←CPEVAL(p,U,b); if $s = 1$, D←PFA(U,D); if $s \neq 1$, D←PFL(U,D); if $W \neq ( )$, go to (3).

(4) [Prefix next row.] C←PFL(INV(D),C); if $X \neq ( )$, go to (2).

(5) [Return.] C←INV(C); return C.

Computing time: $\lesssim mn(\Pi_{i=1}^{s}\mu_i)$, for $A \in M(m,n,P^*(m_1,\ldots,m_s))$ and $\mu_i = m_i + 1$.

H = MCPINT(p,A,b,C,D,s)

p is a prime number, b is an element of GF(p), A is an m by n matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, C is an m by n matrix over $GF(p)[x_1,\ldots,x_{s-1}]$, and D is a univariate polynomial $D(x) = \Pi_{i=0}^{k}(x-b_i)$ over GF(p), where $b \neq b_t$, $0 \leq t \leq k$, and the b_t are distinct. The degree of each element of A in x_s is at most k. H is the m by n matrix over $GF(p)[x_1,\ldots,x_s]$ such that each element H(i,j) is the unique interpolating polynomial of degree k+1 or less in x_s such that $H(i,j)(x_1,\ldots,x_{s-1},b_t) = A(i,j)(x_1,\ldots,x_{s-1},b_t)$, $0 \leq t \leq k$, and also $H(i,j)(x_1,\ldots,x_{s-1},b) = C(i,j)(x_1,\ldots,x_{s-1})$.

Note that the special case k = -1 is permitted, in which case D(x) = 1, A is the m by n zero matrix, and H = C is computed.

Algorithm

(1) [Initialize.] X ← A; Y ← C; H ← ( ).

(2) [Begin next row.] ADV(W,X); ADV(T,Y); E ← ( ).

(3) [Interpolate to get next row element.] ADV(U,W); ADV(V,T); F ← CPINT(p,U,b,V,D,s); E ← PFL(F,E); if $W \neq ( )$, go to (3).

(4) [Prefix next row.] H ← PFL(INV(E),H); if $X \neq ( )$, go to (2).

(5) [Return.] H ← INV(H); return H.

Computing time: $\lesssim mn(k+2)(\Pi_{i=1}^{s-1}\mu_i)$, for $A \in M(m,n,P^*(m_1,\ldots,m_{s-1},k))$ and $C \in M(m,n,P^*(m_1,\ldots,m_{s-1}))$, and $\mu_i = m_i + 1$.

2.3 A Modular Algorithm for Matrix Multiplication

Five algorithms are described, three of which are the outer (or main), middle, and inner portions of the modular matrix multiplication algorithm, the remaining two computing integer coefficient bounds and degree bounds, respectively. The outer algorithm applies mod-p homomorphisms and the Chinese Remainder Algorithm to compute the product of two matrices over $I[x_1,\ldots,x_s]$, $s \geq 0$. This algorithm employs the middle algorithm to compute the products of pairs of matrices over $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$, using evaluation and interpolation. The middle algorithm eventually applies the inner algorithm, which computes the product of two matrices over GF(p) by the classical algorithm. The three main algorithms are given in the order: outer, middle, inner.

C = MMPY(A,B)

A and B are matrices over $I[x_1,\ldots,x_s]$, $s \geq 0$, where A is m by n and B is n by q. C is the m by q matrix product AB.

Note that this algorithm requires the list PRIME of odd single-precision primes.

Algorithm

(1) [Initialize.] $B' \leftarrow$ MTRAN(B); $Z \leftarrow$ PRIME; $m \leftarrow$ LENGTH(A); $q \leftarrow$ LENGTH(B').

(2) [Construct the m by q zero matrix C.] $C \leftarrow (\,)$; for i = 1, m do: ($D \leftarrow (\,)$; for j = 1, q do: ($D \leftarrow$ PFL(0,D)); $C \leftarrow$ PFL(D,C)); if MZERO(A) = 1 or MZERO(B) = 1, go to (8).

(3) [Finish initializing Garner's method.] $V \leftarrow$ MVLIST(A); $s \leftarrow$ LENGTH(V); $I \leftarrow$ PFA(1,0); $J \leftarrow$ MMCB(A,B').

(4) [Apply a mod-p mapping, if available.] If $Z = (\,)$, (print error message; stop); ADV(p,Z); $A^* \leftarrow$ MMOD(p,A,s); $B^* \leftarrow$ MMOD(p,B',s).

(5) [Apply evaluation mapping algorithm.] $C^* \leftarrow$ MCPMPY(p,A^*,B^*); erase A^*, B^*.

(6) [Apply Garner's method.] $C' \leftarrow$ MGARN(I,C,p,C^*,V); erase C,C^*; $C \leftarrow C'$.

(7) [Recompute I and test.] $T \leftarrow$ PFA(p,0); $U \leftarrow$ IPROD(I,T); erase I,T; $I \leftarrow U$; if ICOMP(I,J) $\neq 1$, go to (4); erase V,I,J.

(8) [Return.] Erase B'; return C.

Computing time: $\lesssim L(nde) \cdot [mnq(\Pi_{i=1}^{s}\kappa_i) + mn\{L(d) \cdot (\Pi_{i=1}^{s}\mu_i) + \Sigma_{i=1}^{s}(\Pi_{j=1}^{i}\mu_j)(\Pi_{j=i}^{s}\kappa_j)\} + nq\{L(e) \cdot (\Pi_{i=1}^{s}\nu_i) + \Sigma_{i=1}^{s}(\Pi_{j=1}^{s}\nu_j)(\Pi_{j=i}^{s}\kappa_j)\} + mq\{(\Pi_{i=1}^{s}\kappa_i)(L(de) + \Sigma_{i=1}^{s}\kappa_i)\}] \lesssim L(nde)\kappa^s(mnq + mn(\mu + L(d)) + nq(\nu + L(e)) + mq(\kappa + L(de)))$, for $A \in M(m,n,P(d,m_1,\ldots,m_s))$, $B \in M(n,q,P(e,n_1,\ldots,n_s))$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $\mu = \max \mu_i$, $\nu = \max \nu_i$, $\kappa_i = \mu_i + \nu_i$, and $\kappa = \max \kappa_i$.

K = MMCB(**A**,B)

A and B are matrices over $I[x_1,\ldots,x_s]$, $s \geq 0$, where A is m by n and represented rowwise and B is n by q and represented columnwise. K is an infinite-precision integer such that $K/2 \geq$ norm(C(i,j)), for all elements of C = AB.

Algorithm

(1) [Initialize.] X ← A; Y ← B; K ← 0; n ← LENGTH(FIRST(X)); I ← ((),(),...,()), a list of length n; J ← ((),(),...,()), a list of length n.

(2) [Begin column bound list for A.] ADV(R,X); S ← I.

(3) [Compare next row element.] ADV(U,R); U ← PNORMF(U); V ← FIRST(S); h ← ICOMP(U,V); if h = 1, (erase V; ALTER(U,S); if h ≠ 1, erase U; S ← TAIL(S); if S ≠ ( ), go to (3); if X ≠ ( ), go to (2).

(4) [Begin row bound list for B.] ADV(R,Y); S ← J.

(5) [Compare next column element.] ADV(U,R); U ← PNORMF(U); V ← FIRST(S); h ← ICOMP(U,V); if h = 1, (erase V; ALTER(U,S)); if h ≠ 1, erase U; S ← TAIL(S); if S ≠ ( ), go to (5); if Y ≠ ( ), go to (4).

(6) [Compute and return norm bound.] DECAP(U,I); DECAP(V,J); R ← IPROD(U,V); erase U and V; S ← ISUM(K,R); erase K

and R; $K \leftarrow S$; if $I \neq ( )$, go to (6); $U \leftarrow PFA(2,0)$; $V \leftarrow$ IPROD(U,K); erase U, K; $K \leftarrow V$; return K.

Computing time: $\lesssim mn \cdot L(d) \cdot (\prod_{i=1}^{s} \mu_i) + nq \cdot L(e) \cdot (\prod_{i=1}^{s} \nu_i) + n \cdot L(d)L(e) + nL(n)$, where $A \in M(m,n,P(d,m_1,\ldots,m_s))$, $B \in M(n,q,P(e,n_1,\ldots,n_s))$, $\mu_i = m_i + 1$, and $\nu_i = n_i + 1$.

C = MCPMPY(p,A,B)

p is a prime number and A and B are matrices over $GF(p)[x_1, \ldots, x_s]$, $s \geq 0$, where A is m by n and represented rowwise and B is n by q and represented columnwise. C is the m by q matrix product AB.

Algorithm

(1) [Initialize.] m ← LENGTH(A); q ← LENGTH(B).

(2) [Compute product over GF(p).] If TYPE(FIRST(A)) ≠ 0, go to (3); C ← MCMPY(p,A,B); return C.

(3) [Construct the m by q zero matrix C.] C ← ( ); for i = 1,m do: (D ← ( ); for j = 1, q do: (D ← PFL(0,D)); C ← PFL(D,C)); if MZERO(A) = 1 or MZERO(B) = 1, return C.

(4) [Finish initializing the interpolation.] s ← MCPNV(A); k ← MCPMDB(A,B); i ← 0; D - PFA(0,PFA(1,0)).

(5) [Apply another evaluation mapping, if one exists.] If i = p, print error message and stop; $A^* \leftarrow$ MCPEVL(p,A,i,s); $B^* \leftarrow$ MCPEVL(p,B,i,s).

(6) [Apply algorithm recursively.] $C^* \leftarrow$ MCPMPY(p,A^*,B^*); erase A^* and B^*.

(7) [Apply incremental interpolation.] $C' \leftarrow$ MCPINT(p,C,i,C^*,D,s); erase C, C^*; $C \leftarrow C'$; if $i < k$, go to (8); erase D; return C.

(8) [Recompute polynomial D for next interpolation.] $E \leftarrow$ PFA(1,PFA(1,PFA(CDIF(p,0,i),0))); $W \leftarrow$ CPPROD(p,D,E); erase D and E; $D \leftarrow W$; $i \leftarrow i + 1$; go to (5).

Computing time: for $s = 0$, the maximum computing time is $\sim mnq$ and, for $s \geq 1$, it is $\lesssim mnq(\Pi_{i=1}^{s} \kappa_i) + mn\{\Sigma_{i=1}^{s}(\Pi_{j=1}^{i} \mu_j)(\Pi_{j=i}^{s} \kappa_j)\} + nq\{\Sigma_{i=1}^{s}(\Pi_{j=1}^{i} \nu_j)(\Pi_{j=i}^{s} \kappa_j)\} + mq\{(\Pi_{i=1}^{s} \kappa_i)(\Sigma_{i=1}^{s} \kappa_i)\} \lesssim \kappa^s(mnq + mn\mu + nq\nu + mq\kappa)$, for $A \in M(m,n,P^*(m_1,\ldots,m_s))$, $B \in M(n,q,P^*(n_1,\ldots,n_s))$, $\mu_i = m_i + 1$, $\nu_i = n_i + 1$, $\mu = \max \mu_i$, $\nu = \max \nu_i$, $\kappa_i = \mu_i + \nu_i$, and $\kappa = \max \kappa_i$.

z = MCPMDB(A,B)

A and B are matrices over $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, where A is m by n and represented rowwise and B is n by q and represented columnwise. z is a non-negative Fortran integer which bounds the degrees in the main variable x_s of the elements of the product AB.

Algorithm

(1) [Initialize.] $X \leftarrow A$; $Y \leftarrow B$; $z \leftarrow 0$.

(2) [Begin degree bound for next row.] ADV(R,X); $W \leftarrow Y$.

(3) [Get degree of next row element.] $S \leftarrow R$; ADV(T,W).

(4) [Compare degree of next term of inner product.] ADV(U,S); ADV(V,T); if $U = 0$ or $V = 0$, go to (5); $e \leftarrow \deg(U) + \deg(B)$; if $e > z$, $z \leftarrow e$.

(5) [Termination tests and return.] If $S \neq ( )$, go to (4); if $W \neq ( )$, go to (3); if $X \neq ( )$, go to (2); return z.

Computing time: the maximum computing time is $\sim mnq$.

C = MCMPY(p,A,B)

A and B are matrices over GF(p), where A is m by n and represented rowwise and B is n by q and represented columnwise. C is the m by q matrix product AB.

Algorithm

(1) [Initialize.] X ← CINV(A); Y ← CINV(B); C ← ( ); V ← X.

(2) [Begin next row of product.] ADV(R,V); D ← ( ); W ← Y.

(3) [Begin to compute next element as an inner product.] S ← R; ADV(T,W); e ← 0.

(4) [Add next term of inner product.] e ← CSUM(p,e,CPROD(p, FIRST(S),FIRST(T))); S ← TAIL(S); T ← TAIL(T); if S ≠ ( ), go to (4); D ← PFA(e,D); if W ≠ ( ), go to (3); C ← PFL(D,C); if V ≠ ( ), go to (2).

(5) [Return.] Erase X,Y; return C.

Computing time: the maximum computing time is ~mnq.

2.4 Auxiliary Algorithms for Solving Linear Equations

Various supporting algorithms are described, which will be required by the modular algorithm for solving linear equations. The first performs the lexicographical comparison of two integral vectors.

$$z = \text{VCOMP}(H,K)$$

H and K are lists of Fortran integers, either possibly null. z is the Fortran integer -1, 0, or 1 depending on whether $H < K$, $H = K$, or $H > K$, respectively.

Algorithm

(1) [Initialize.] $S \leftarrow H$; $T \leftarrow K$.

(2) [Test if H exhausted.] If $S \neq ( )$, go to (3); if $T \neq ( )$, go to (5); $z \leftarrow 0$; return z.

(3) [Test if K exhausted.] If $T \neq ( )$, go to (4); go to (6).

(4) [Compare next elements.] ADV(u,S); ADV(v,T); $w \leftarrow u - v$; if $w = 0$, go to (2); if $w > 0$, go to (6).

(5) [$H < K$ return.] $z \leftarrow -1$; return z.

(6) [$H > K$ return.] $z \leftarrow 1$; return z.

Computing time: the maximum computing time is $\sim \min(m,n) + 1$, for H of length m and K of length n.

B = MZCON(A)

A is an m by n matrix over $I[x_1,\ldots,x_s]$, $s \geq 0$, or $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, for some prime p. B is an m by n zero matrix.

Algorithm

(1) [Initialize.] B ← ( ); S ← A.

(2) [Begin next row.] ADV(U,S); X ← ( ).

(3) [Prefix next row element.] X ← PFL(0,X); U ← TAIL(U); if U ≠ ( ), go to (3); B ← PFL(X,B); if S ≠ ( ), go to (2).

(4) [Return.] B ← INV(B); return B.

Computing time: ~ mn.

z = MEQ(A,B)

A and B are m by n matrices over $I[x_1,\ldots,x_s]$, $s \geq 0$. z is the Fortran integer 1 or 0 depending on whether $A = B$ or $A \neq B$.

Algorithm

(1) [Initialize.] $X \leftarrow A$; $Y \leftarrow B$.

(2) [Obtain next rows.] ADV(S,X); ADV(T,Y).

(3) [Test next row elements.] $C \leftarrow$ PDIF(FIRST(S), FIRST(T)); if $C \neq 0$, go to (4); $S \leftarrow$ TAIL(S); $T \leftarrow$ TAIL(T); if $S \neq ( )$, go to (3); if $X \neq ( )$, go to (2); $z \leftarrow 1$; return z.

(4) [Unequal return.] Erase C; $z \leftarrow 0$; return z.

Computing time: $\lesssim mn(L(d) + L(e))(\Pi_{i=1}^{s}\mu_i + \Pi_{i=1}^{s}\nu_i)$, for $A \in M(m,n,P(d,m_1,\ldots,m_s))$, $B \in M(m,n,P(e,n_1,\ldots,n_s))$, $\mu_i = m_i + 1$, and $\nu_i = n_i + 1$.

z = MCPEQ(p,A,B)

p is a prime number and A and B are m by n matrices over $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$. z is the Fortran integer 1 or 0 depending on whether $A = B$ or $A \neq B$, respectively.

Algorithm

(1) [Initialize.] $X \leftarrow A$; $Y \leftarrow B'$.

(2) [Obtain next rows.] ADV(S,X); ADV(T,Y).

(3) [Test next row elements.] $C \leftarrow$ CPDIF(p,FIRST(S), FIRST(T)); if $C \neq 0$, go to (4); $S \leftarrow$ TAIL(S); $T \leftarrow$ TAIL(T); if $S \neq ( )$, go to (3); if $X \neq ( )$, go to (2); $z \leftarrow 1$; return z.

(4) [Unequal return.] Erase C; $z \leftarrow 0$; return z.

Computing time: $\lesssim mn(\Pi_{i=1}^{s}\mu_i + \Pi_{i=1}^{s}\nu_i)$, for $A \in M(m,n,P^*(m_1,\ldots,m_s))$, $B \in M(m,n,P^*(n_1,\ldots,n_s))$, $\mu_i = m_i + 1$, and $\nu_i = n_i + 1$.

Z = NULCON(n,J,P,W)

n is a positive Fortran integer, J is the RE sequence of an m by n RRE matrix F of rank r: $1 \leq r < n$ over $I[x_1,\ldots,x_s]$, $s \geq 0$, or $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, for some p. -P is the common diagonal value of F and W is the non-diagonal part such that (J,-P,W) is the compact representation of F. Z is the n by n-r null space basis matrix for F, as defined by formula (2) of Section 1.

Algorithm

(1) [Initialize.] L ← J; W' ← CINV(W); r ← LENGTH(J); t ← r - LENGTH(W); if t > 0, apply W' ← PFL(0,W') t times; W' ← INV(W'); e ← n-r; Z ← ( ); ADV(h,L); I ← 0; k ← 0.

(2) [Begin next row of Z.] i ← i+1; if i > n, go to (5); if i = h, go to (4).

(3) [Construct next row using P.] k ← k+1; U ← ( ); t ← e-k; if t > 0, do U ← PFL(0,U) t times; U ← PFL(BORROW(P),U); t ← k-1; if t > 0, do U ← PFL(0,U) t times; Z ← PFL(U,Z); go to (2).

(4) [Construct next row from row of W'.] DECAP(U,W'); U ← INV(CINV(U)); t ← e - LENGTH(U); if t > 0, do U ← PFL(0,U) t times; Z ← PFL(U,Z); if L ≠ ( ), ADV(h,L); go to (2).

(5) [Return.] Z ← INV(Z); return Z.

Computing time: the maximum computing time is $\sim n(n-r)$.

b = DBRRE(J,C)

C is an m by n nonzero matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 1$, and J is the RE sequence of C. b is a positive Fortran integer which bounds the degrees in the main variable x_s of the elements of any RRE form for C which might be computed in the modular algorithm for solving linear equations.

Algorithm

(1) [Initialize.] X ← MTRAN(C); Y ← X; K ← J; e ← 0; f ← 0; L ← ( ); i ← 0.

(2) [Obtain next column of C.] If Y = ( ), go to (5); ADV(U,Y); i ← i+1; if K = ( ), go to (4); if i ≠ FIRST(K), go to (4); g ← 0.

(3) [Get maximum degree of pivot column.] ADV(T,U); if T ≠ 0, (d ← deg(T); if d > g, g ← d); if U ≠ ( ), go to (3); L ← PFA(g,L); K ← TAIL(K); go to (2).

(4) [Get maximum degree of non-pivot column.] ADV(T,U); if T ≠ 0, (d ← deg(T); if d > f, f ← d); if U ≠ ( ), go to (4); go to (2).

(5) [Compute sum and minimum of elements of list L.] DECAP(d,L); e ← e+d; if d < g, g ← d; if L ≠ ( ), go to (5).

(6) [Complete computation of degree bound.] b ← e; h ← f-g; if h > 0, b ← b+h; erase X; return b.

Computing time: ~ mn.

2.5 A Modular Algorithm for Solving Linear Equations

This algorithm is presented as three separate subalgorithms: an outer algorithm employing mod-p homomorphisms and the Chinese Remainder Algorithm, a middle algorithm employing evaluation homomorphisms and interpolation, and an inner algorithm which employs a Gaussian elimination algorithm in computing the determinantal RRE form for any nonzero matrix over GF(p). These algorithms are presented in the order: outer, middle, inner.

$$G = \text{PLES}(n, C)$$

n is a positive Fortran integer and C is an m by n' matrix over $I[x_1, \ldots, x_s]$, $s \geq 0$, where $n < n'$. C is the augmented matrix for a linear system $AX = B$, where A is m by n and nonzero and B is m by q. G is the list (D,Y,Z), representing a general solution to the system $AX = B$, if the system is consistent, or G is the null list ( ), if the system is inconsistent.

Note that this algorithm requires the list PRIME of distinct odd primes.

Algorithm

(1) [Initialize.] $X \leftarrow \text{MVLIST}(C)$; $L \leftarrow \text{PRIME}$; $r \leftarrow 0$.

(2) [Apply mod-p mapping φ_p.] If $L = ( )$, print error message and stop; otherwise, ADV(p,L); $C^* \leftarrow$ MMOD(p,C,s); if MZERO(C^*) = 1, (erase C^*; go to (2)).

(3) [Compute an RRE form F^* for C^* by the evaluation mapping algorithm.] $W^* \leftarrow$ CPRRE(p,C^*); DECAP(J^*,W^*); DECAP(I^*,W^*); DECAP(D^*,W^*); DECAP(V^*,W^*).

(4) [Rejection tests.] $r^* \leftarrow$ LENGTH(J^*); if $r^* < r$, go to (5); if $r^* > r$, go to (6); $u \leftarrow$ VCOMP(J^*,J); if $u = 1$, go to (5); if $u = -1$, go to (6); $u \leftarrow$ VCOMP(I^*,I); if $u = 1$, go to (5); if $u = -1$, go to (6); erase J^*,I^*; go to (7).

(5) [Discard φ_p.] Erase J^*, I^*; if $X \neq ( )$, erase D^*; if $V^* \neq ( )$, erase V^*; go to (2).

(6) [Discard previously retained mod-p mappings and initialize interpolation.] If $r > 0$, (erase J,I,D,V,E); $J \leftarrow J^*$; $I \leftarrow I^*$; $D \leftarrow 0$; $V \leftarrow ( )$; if $V^* \neq ( )$, $V \leftarrow$ MZCON(V^*); $E \leftarrow$ PFA(1,0); $r \leftarrow r^*$.

(7) [Retain φ_p and apply Garner's Method.] $D' \leftarrow$ CPGARN(E, D,p,D^*,X); if $X \neq ( )$, erase D^*; if $V \neq ( )$, ($V' \leftarrow$ MGARN(E,V,p,V^*,X); erase V^*); $T \leftarrow$ PFA(p,0); $E' \leftarrow$ IPROD(T,E); erase T,E; $E \leftarrow E'$.

(8) [Equality test.] $T \leftarrow$ PDIF(D,D'); erase D; $D \leftarrow D'$; $u \leftarrow 1$ if $V \neq ( )$, ($u \leftarrow$ MEQ(V,V'); erase V; $V \leftarrow V'$); if $u = 1$ and $T = 0$, go to (9); erase T; go to (2).

(9) [Substitution test.] If $r > n$, ($G \leftarrow ( )$; erase D; go to (12)); $n' \leftarrow$ LENGTH(FIRST(C)); $D' \leftarrow$ PNEG(D); $Z' \leftarrow$ NULCON(n',J,D',V); $T \leftarrow$ MMPY(C,Z'); $u \leftarrow$ MZERO(T); erase D', T; if $u = 0$, (erase Z'; go to (2)).

(10) [Inconsistent system test.] $J \leftarrow$ INV(J); if FIRST(J) $> n$, (erase Z'; $G \leftarrow ( )$; erase D; go to (12)); $Y \leftarrow ( )$; $Z \leftarrow ( )$; $h \leftarrow n-r-1$; $u \leftarrow n$.

(11) [Construct general solution.] $u \leftarrow u-1$; DECAP(T,Z'); if $h \geq 0$, ($Z \leftarrow$ PFL(T,Z); if $h > 0$, do $T \leftarrow$ TAIL(T) h times; $W \leftarrow T$; $T \leftarrow$ TAIL(T); SSUCC(0,W)); $Y \leftarrow$ PFL(T,Y); if $u > 0$, go to (11); erase Z'; $G \leftarrow$ PFL(D,PFL(INV(Y),PFL(INV(Z),0))).

(12) [Final erasures and return.] Erase J,I,V,E,X; return G.

Computing time: for $s = 0$, $\lesssim (L(n') + rL(rd)) \cdot [m(n')^2 + (m+n')r(n'-r+1)L(rd)]$ and, for $s > 0$, $\lesssim (L(n') + rL(rd))r^s(\Pi_{i=1}^s \mu_i)$ $[m(n')^2 + (m+n')r(n'-r+1)(L(rd) + \Sigma_{i=1}^s \mu_i)$, for $C \in M(m,n',P(d, m_1,\ldots,m_s))$, C of rank r, and $\mu_i = m_i + 1$. If $n' \sim m$ and $n-r \lesssim 1$, then $\lesssim m^{s+3}L(md)(\Pi_{i=1}^s \mu_i)(m + q(L(d) + \Sigma_{i=1}^s \mu_i)) \lesssim m^{s+3}$ $\mu^s L(md)\,(m + q(\mu + L(d))) \lesssim m^{s+4}\,\mu^s\, L(md)(\mu + L(d))$, for $s \geq 0$, where $\mu = \max \mu_i$, and $n' = n + q$.

W = CPRRE(p,C)

p is a prime number and C is an m by n nonzero matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$. W is the list (J,I,D,V), where J is the RE sequence J_C, I is the permutation I_C, D is the common diagonal value of the determinantal RRE form $\overline{\overline{C}}$ for C, and V is the non-diagonal part of the compact representation of $\overline{\overline{C}}$. The list C is erased.

Algorithm

(1) [Apply Gaussian elimination algorithm] s ← MCPNV(C); if s > 0, go to (2); W ← CRRE(p,C); return W.

(2) [Initialize evaluation mapping algorithm.] r ← 0; a ← p; h ← 0.

(3) [Apply evaluation mapping Ψ_a.] a ← a-1; if a < 0, print error message and stop; otherwise, C^* ← MCPEVL(p,C,a,s); if MZERO(C^*) = 1, (erase C^*; go to (3)).

(4) [Apply algorithm recursively.] W^* ← CPRRE(p,C^*); DECAP(J^*,W^*); DECAP(I^*,W^*); DECAP(D^*,W^*); DECAP(V^*,W^*).

(5) [Rejection tests.] r^* ← LENGTH(J^*); if $r^* < r$, go to (6); if $r^* > r$, go to (7); u ← VCOMP(J^*,J); if u = 1, go to (6); if u = -1, go to (7); u ← VCOMP(I^*,I); if u = 1, go to (6); if u = -1, go to (7); erase J^*, I^*; go to (8).

(6) [Discard Ψ_a.] Erase J^*, I^*, V^*; if $s > 1$, erase D^*, go to (3).

(7) [Discard previously retained evaluation mappings and initialize interpolation.] If $r > 0$, (erase J, I, D, V, E); $J \leftarrow J^*$; $I \leftarrow I^*$; $D \leftarrow 0$; $V \leftarrow ( )$; if $V^* \neq ( )$, $V \leftarrow MZCON(V^*)$; $E \leftarrow PFA(0, PFA(1,0))$; $r \leftarrow r^*$.

(8) [Retain Ψ_a and apply interpolation.] $D' \leftarrow CPINT(p,D,a,D^*, E,s)$; if $s > 1$, erase D^*; if $V \neq ( )$, ($V' \leftarrow MCPINT(p,V,a, V^*,E,s)$; erase V^*); $T \leftarrow PFA(1,PFA(1,PFA(CDIF(p,0,a),0)))$; $E' \leftarrow CPPROD(p,T,E)$; erase T,E; $E \leftarrow E'$; if $h = 0$, go to (9); erase D; $D \leftarrow D'$; if $V \neq ( )$, (erase V; $V \leftarrow V'$); go to (12).

(9) [Equality test.] $T \leftarrow CPDIF(p,D,D')$; erase D; $D \leftarrow D'$; $u \leftarrow 1$; if $V \neq ( )$, ($u \leftarrow MCPEQ(p,V,V')$; erase V; $V \leftarrow V'$); if $u = 1$ and $T = 0$, to to (10); erase T; go to (3).

(10) [Substitution test.] $n \leftarrow LENGTH(FIRST(C))$; if $r = n$, ($h \leftarrow 1$; go to (11)); $D' \leftarrow CPNEG(p,D)$; $Z \leftarrow NULCON(n,J, D',V)$; $T \leftarrow MCPMPY(p,C,Z)$; $u \leftarrow MZERO(T)$; erase D',Z,T; if $u = 0$, go to (3); $h \leftarrow 1$.

(11) [Compute degree bound.] $b \leftarrow DBRRE(J,C)$.

(12) [Degree test.] If $CPDEG(E) \leq b$, go to (3).

(13) [Construct output list and return.] Erase E,C; $W \leftarrow PFL(J,PFL(I,PFL(D,PFL(V,0))))$; return W.

Computing time: $\sim mnr$, if $s = 0$, and $\lesssim r^s(\Pi_{i=1}^s \mu_i)[mn^2 + (m+n)(n-r+1)r(\Sigma_{i=1}^s \mu_i)]$, if $s \geq 1$, for $C \in M(m,n,P^*(m_1,\ldots,m_s))$, C of rank r and $\mu_i = m_i + 1$. If $m \sim n \sim r$ then this is $\lesssim m^{s+3}\mu^{s+1}$, for $\mu = \max \mu_i$.

W = CRRE(p,A)

p is a prime number and A is an m by n nonzero matrix over GF(p), whose list representation does not overlap another list. W is the list (J,I,d,V), where J is the RE sequence J_A of A, I is the permutation I_A, d is the common diagonal value of the determinantal RRE form $\overline{\overline{A}}$ for A, and V is the non-diagonal part of the compact representation of $\overline{\overline{A}}$. The list A is erased.

Algorithm

(1) [Initialize.] B' ← A; m ← LENGTH(B'); n ← LENGTH(FIRST(B')); I ← ( ); J ← ( ); d ← 1; V ← ( ); B ← ( ); I' ← ( ); for k = 1, m do: (I' ← PFA(m+1-k,I')); z ← 0.

(2) [Begin next step of triangularization.] z ← z+1; B" ← B'; B' ← ( ); I" ← I'; I' ← ( ).

(3) [Search column z for next pivot element.] DECAP(F,B"); DECAP(e,F); DECAP(k,I"); if e ≠ 0, go to (4); I' ← PFA(k,I'); if F ≠ ( ), B' ← PFL(F,B'); if B" ≠ ( ), go to (3); go to (8).

(4) [Update d, J, I, and B, and begin elimination.] d ← CPROD(p,e,d); J ← PFA(z,J); I ← PFA(k,I); B ← PFL(F,B); if F = ( ), go to (8); e ← CRECIP(p,e); G ← F.

(5) [Transform pivot row.] ALTER(CPROD(p,e,FIRST(G)), G); G ← TAIL(G); if G ≠ ( ), go to (5).

(6) [Begin to transform next row, if any.] If $B'' = ( )$, go to (8); DECAP(G,B"); DECAP(u,G); $B' \leftarrow$ PFL(G,B'); if $u = 0$, go to (6); $H \leftarrow F$.

(7) [Recompute next element of row G.] $s \leftarrow$ CPROD(p,u,FIRST(H)); $t \leftarrow$ CDIF(p,FIRST(G),s); ALTER(t,G); $G \leftarrow$ TAIL(G); $H \leftarrow$ TAIL(H); if $G \neq ( )$, go to (7); go to (6).

(8) [Test for end of triangularization.] $B' \leftarrow$ INV(B'); $I' \leftarrow$ CONC(INV(I'),I"); if $B' \neq ( )$, go to (2); $I \leftarrow$ CONC(INV(I),I'); if $B'' \neq ( )$, erase B".

(9) [Initialize splitting of row.] If $B = ( )$, ($U \leftarrow V$; $V \leftarrow ( )$; go to (15)); DECAP(H,B); $H \leftarrow$ INV(H); $\bar{H} \leftarrow ( )$; $M \leftarrow ( )$; $z \leftarrow n$; $L \leftarrow J$.

(10) [Split row.] If $H = ( )$, go to (11); DECAP(u,H); $k \leftarrow$ FIRST(L); if $z = k$, ($M \leftarrow$ PFA(u,M); $L \leftarrow$ TAIL(L)); if $z \neq k$, $\bar{H} \leftarrow$ PFA(u,$\bar{H}$); $z \leftarrow z-1$; go to (10).

(11) [Initialize diagonalization of row.] $T \leftarrow V$; $\bar{H} \leftarrow$ INV($\bar{H}$).

(12) [Prepare transformation by next row of V.] If $M = ( )$, go to (14); DECAP(u,M); if $T = ( )$, go to (12); ADV(S,T); $H \leftarrow \bar{H}$.

(13) [Transform by next row of V.] If $S = ( )$, go to (12); ADV(x,S); $w \leftarrow$ FIRST(H); $w \leftarrow$ CDIF(p,w,CPROD(p,u,x); ALTER(w,H); $H \leftarrow$ TAIL(H); go to (13).

(14) [Prefix row to V.] If $\bar{H} \neq ( )$, $V \leftarrow PFL(\bar{H}, V)$; go to (9).

(15) [Compute next row of non-diagonal part.] If $U = ( )$, go to (17); DECAP(T,U); $S \leftarrow ( )$.

(16) [Multiply next row element by $\delta(A)$.] DECAP(e,T); $g \leftarrow$ CPROD(p,d,e); $S \leftarrow$ PFA(g,S); if $T \neq ( )$, go to (16); $V \leftarrow$ PFL(S,V); go to (15).

(17) [Construct output list and return.] $V \leftarrow$ INV(V); $J \leftarrow$ INV(J); $W \leftarrow$ PFL(J,PFL(I,PFA(d,PFL(V,0)))); return W.

Computing time: the maximum computing time is $\sim mnr$, where A is of rank r.

2.6 A Modular Algorithm for Determinant Calculation

As for matrix multiplication and linear equations solution, this algorithm is presented as outer, middle, and inner subalgorithms, in that order. The outer algorithm requires an additional algorithm for computing an integer coefficient bound, CBDET, while the middle algorithm employs DBRRE to compute a degree bound. The inner algorithm is a Gaussian elimination algorithm for computing determinants of matrices over GF(p).

D = PDET(A)

A is an m by n nonzero matrix over $I[x_1,\ldots,x_s]$, $s \geq 0$. D is the determinant of A, an element of $I[x_1,\ldots,x_s]$.

Note that this algorithm requires the list PRIME of distinct odd primes.

Algorithm

(1) [Initialize.] $D \leftarrow 0$; $V \leftarrow$ MVLIST(A); $s \leftarrow$ LENGTH(V); $J \leftarrow$ CBDET(A); $I \leftarrow$ PFA(1,0); $L \leftarrow$ PRIME; $h \leftarrow 0$.

(2) [Apply mod-p mapping φ_p.] If L = ( ), (print error message; stop); ADV(p,L); $A^* \leftarrow$ MMOD(p,A,s); if MZERO(A^*) = 1, (erase A^*; $D^* \leftarrow 0$; go to (4)).

(3) [Apply evaluation mapping algorithm.] $D^* \leftarrow CPDET(p,A^*)$.

(4) [Apply Garner's Method.] If $D^* \neq 0$, $h \leftarrow 1$; if $h = 0$, go to (5); $D' \leftarrow CPGARN(I,D,p,D^*,V)$; erase D; if $V \neq 0$, erase D^*; $D \leftarrow D'$.

(5) [Integer coefficient bound test.] $T \leftarrow PFA(p,0)$; $U \leftarrow IPROD(I,T)$; erase I,T; $I \leftarrow U$; if $ICOMP(I,J) \neq 1$, go to (2).

(6) [Final erasures and return.] Erase V,I,J; return D.

Computing time: for $s = 0$, $\lesssim m^3L(md)(L(d) + r)$; for $s = 1$, $\lesssim m^3L(md)\mu_1(L(d) + mr + m\mu_1)$; and for $s \geq 2$, $\lesssim m^3L(md)(\Pi_{i=1}^{s}\mu_i)(L(d) + m^s r + m \cdot \Sigma_{i=1}^{s}\mu_i m^{s-i}) \lesssim m^3\mu^s L(md)(L(d) + m^s(r+\mu))$, if $r < m$, and $\lesssim m^{s+2}L(md)(\Pi_{i=1}^{s}\mu_i)(L(d) + m^2 + m\mu_1 + \Sigma_{i=2}^{s}\mu_i) \lesssim m^{s+2}\mu^s L(md)(L(d) + m(m+\mu))$, if $r = m$. These hold for $A \in M(m,m,P(d,m_1,\ldots,m_s))$, A of rank r, $\mu_i = m_i + 1$, and $\mu = \max \mu_i$.

J = CBDET(A)

A is an m by m matrix over $I[x_1,\ldots,x_s]$, $s \geq 0$. J is an infinite-precision integer such that J/2 bounds the magnitudes of the integer coefficients of the determinant of A.

Algorithm

(1) [Initialize.] B ← A; m ← LENGTH(B); J ← PFA(2,0).

(2) [Begin to compute maximum norm of next row.] ADV(C,B); K ← 0.

(3) [Compare norm of next row element.] ADV(D,C); L ← PNORMF(D); u ← ICOMP(L,K); if u = 1, (erase K; K ← L); if u ≠ 1, erase L; if C ≠ ( ), go to (3).

(4) [Recompute coefficient bound.] L ← IPROD(J,K); erase J,K; J ← L; if B ≠ ( ), go to (2).

(5) [Final computations for bound.] K ← IFACT(m); L ← IPROD(K,J); erase K,J; J ← L; return J.

Computing time: $\lesssim m^2L(d)(L(md) + \Pi_{i=1}^{s}\mu_i) + m^2L(m)^2$, for A ε M(m,m,P(d, $m_1,\ldots,m_s$)) and $\mu_i = m_i + 1$.

D = CPDET(p,A)

p is a prime number and A is an m by m nonzero matrix over $GF(p)[x_1,\ldots,x_s]$, $s \geq 0$. D is the determinant of A, an element of $GF(p)[x_1,\ldots,x_s]$. The list A is erased.

Algorithm

(1) [Get number of variables.] If MZERO(A) = 1, (D ← 0; erase A; return D); s ← MCPNV(A).

(2) [Apply Gaussian elimination algorithm.] If s > 0, go to (3); D ← CDET(p,A); return D.

(3) [Initialize evaluation mapping algorithm.] h ← 0; m ← LENGTH(A); J ← ( ); for i = 1, m do: (J ← PFA(m+1-i,J)); b ← DBRRE(J,A); erase J; i ← 0; E ← PFA(0,PFA(1,0)); D ← 0.

(4) [Apply evaluation mapping Ψ_i.] If i = p, (print error message; stop); $A^* \leftarrow$ MCPEVL(p,A,i,s).

(5) [Apply algorithm recursively.] $D^* \leftarrow$ CPDET(p,A^*); if $D^* \neq 0$, h ← 1; if h = 0, go to (7).

(6) [Interpolation step.] D' ← CPINT(p,D,i,D^*,E,s); erase D; if s > 1, erase D^*; D ← D'.

(7) [Degree test and return.] If CPDEG(E) = b, (erase A, E; return D).

(8) [Initialize next interpolation step.] T ← PFA(1,PFA(1,PFA(CDIF(p,0,a,),0))); U ← CPROD(p,T,E); erase T,E; E ← U; i ← i+1; go to (4).

Computing time: for $s = 0$, the maximum computing time is $\sim m^2 r$; for $s = 1$, it is $\lesssim m^3 \mu_1 (r+\mu_1)$; and for $s \geq 2$, it is $\lesssim m^3 (\Pi_{i=1}^{s} \mu_i)(m^{s-1} r + \Sigma_{i=1}^{s} \mu_i m^{s-i})$, if $r < m$, and $\lesssim m^{s+1} (\Pi_{i=1}^{s} \mu_i)(m^2 + m\mu_1 + \Sigma_{i=2}^{s} \mu_i)$, if $r = m$. These hold for $A \in M(m,m,P^*(m_1,\ldots,m_s))$, A of rank r, and $\mu_i = m_i + 1$. Letting $\mu = \max \mu_i$, for $s \geq 2$, it is $\lesssim m^{s+2} \mu^s (r+\mu)$, $r \leq m$.

d = CDET(p,A)

p is a prime number and A is an m by m nonzero matrix over GF(p). The list A may not overlap any other list. d is the determinant of A, an element of GF(p) and, hence, a Fortran integer. The list A is erased.

Algorithm

(1) [Initialize.] d ← 1; t ← 0; B' ← A.

(2) [Begin next elimination step.] B" ← B' ← ( ).

(3) [Search next column for pivot.] DECAP(F,B"); DECAP(e,F); if e ≠ 0, go to (4); if F ≠ ( ); B' ← PFL(F,B'); t ← t+1; if B" ≠ ( ), go to (3); d ← 0; erase B'; return d.

(4) [Recompute d and begin elimination.] d ← CPROD(p,e,d); if F = ( ), go to (8); e ← CRECIP(p,e); G ← F.

(5) [Transform pivot row.] ALTER(CPROD(p,e,FIRST(G)),G); G ← TAIL(G); if G ≠ ( ), go to (5).

(6) [Begin transforming next row, if any.] If B" = ( ), go to (8); DECAP(G,B"); DECAP(u,G); B' ← PFL(G,B'); if u = 0, go to (6); H ← F.

(7) [Recompute next element of row G.] s ← CPROD(p,u,FIRST(H)); v ← CDIF(p,s,FIRST(G)); ALTER(t,G); G ← TAIL(G); H ← TAIL(H); if G ≠ ( ), go to (7); go to (6).

(8) [Complete determinant calculation, if elimination finished.]

Erase F; $B' \leftarrow INV(B')$; if $B' \neq ( )$, go to (2); $u \leftarrow 0$; QR(u,t,2); if $t = 1$, $d \leftarrow p-d$; return d.

Computing time: the maximum computing time is $\sim m^2 r$, for A of rank r.

2.7 Matrix Inversion and Null Space Basis Calculation

Two of the more important applications of the modular algorithm for linear equations solution are given below. Each is simply a skeleton which structures its input as input to PLES, evokes PLES, and constructs its output from the output of PLES. The first algorithm computes a matrix inverse for a matrix A, which consists of an integer or polynomial D and a matrix Y such that (1/D)AY is the identity matrix.

X = MINV(A)

A is an m by m nonzero matrix over $I[x_1,\ldots,x_s]$, $s \geq 0$. If rank(A) = m, X is the list (D,Y) representing a matrix inverse for A, where D is in $I[x_1,\ldots,x_s]$ and Y is an m by m matrix over $I[x_1,\ldots,x_s]$. If rank(A) < m, X is the null list.

Algorithm

(1) [Initialize.] T ← A; C ← ( ); j ← 0; P ← PFA(1,0); m ← LENGTH(T); L ← MVLIST(T).

(2) [Construct integer or polynomial 1.] If L = ( ), go to (3); DECAP(V,L); P = PFL(V,PFL(0,0))); go to (2).

(3) [Begin next row of augmented matrix C.] ADV(U,T); R ← ( ); j ← j+1.

(4) [Prefix element of A to row R.] ADV(W,U); R ← PFL(BORROW(W),R); if U ≠ ( ), go to (4).

(5) [Suffix elements of identity matrix to row R.] k ← j-1; if k > 0, do R ← PFL(0,R) k times; R ← PFL(BORROW(P),R); k ← m-j; if k > 0, do R ← PFL(0,R) k times; C ← PFL(INV(R),C); if T ≠ ( ), go to (3); C ← INV(C).

(6) [Compute a matrix inverse.] X ← PLES(m,C); if X = ( ), go to (7); T ← TAIL(X); U ← TAIL(T); SSUCC(0,T); erase U.

(7) [Return.] Erase P,C; return X.

Computing time: $\leq (L(m) + rL(rd))r^s (\pi_{i=1}^{s} \mu_i) [m^3 + mr(2m-r)(L(rd) + \Sigma_{i=1}^{s} \mu_i)]$, for A in $M(m,m,P(d,m_1,\ldots,m_s))$, $\mu_i = m_i + 1$ and $\mu = \max \mu_i$. If $m - r \leq 1$, this is $\leq m^{s+4} L(md)(\Pi_{i=1}^{s} \mu_i)(L(md) + \Sigma_{i=1}^{s} \mu_i) \leq m^{s+4} \mu^s L(md)(\mu + L(md))$.

Z = NULSP(A)

A is an m by n nonzero matrix of rank r over $I[x_1,\ldots,x_s]$, $s \geq 0$. If $r < n$, Z is an n by n-r null space basis matrix for A. If $r = n$, Z is the null list.

Algorithm

(1) [Initialize.] T ← A; C ← ( ).

(2) [Begin next row of augmented matrix C.] ADV(U,T); R ← ( ).

(3) [Construct matrix C.] ADV(V,U); R ← PFL(BORROW(V),R); if U ≠ ( ), go to (3); R ← PFL(0,R); C ← PFL(INV(R),C); if T ≠ ( ), go to (2); C ← INV(C).

(4) [Obtain null space basis.] n ← LENGTH(FIRST(A)); G ← PLES(n,C); DECAP(D,G); DECAP(Y,G); DECAP(Z,G); erase D,Y,C; return Z.

Computing time: for $s = 0$, $\lesssim (L(n) + rL(rd))[mn^2 + (m+n)rL(rd)]$ and, for $s \geq 1$, $\lesssim (L(n) + rL(rd))r^s(\Pi_{i=1}^{s}\mu_i)[mn^2 + (m+n)r(n-r+1)(L(rd) + \Sigma_{i=1}^{s}\mu_i)]$, for $A \in M(m,n,P(d,m_1,\ldots,m_s))$ and $\mu_i = m_i + 1$. If $n \sim m \sim r$, then $\lesssim m^{s+3}L(md)(\Pi_{i=1}^{s}\mu_i)(m + (n-r+1)(L(md) + \Sigma_{i=1}^{s}\mu_i))$ $\lesssim m^{s+3}\mu^s L(md)(m + (n-r+1)(\mu + L(md))$, for $s \geq 0$, where $\mu = \max \mu_i$.

3. EMPIRICAL COMPUTING TIMES

Three classes of computations were selected for empirical computing time studies involving a series of representative cases. The computations performed were determinant calculation, solving a single system of linear equations, and matrix inversion, the Fortran programs used being PDET, PLES, and MINV, respectively. The SAC-1 implementation on the UNIVAC 1108 computer under EXEC 8 at the University of Maryland was used. Matrices of randomly generated infinite-precision integers, univariate and bivariate polynomials were used as inputs for each of the three computations. All times are given in seconds.

3.1 Determinant Calculation

Tables 1, 2, and 3 give the times for computing determinants of m by m matrices. Table 1 gives the times for randomly generated integers in the intervals $[-2^k+1, 2^k-1]$, for $k = 4, 8, 16, 32$. Table 2 gives the times for randomly generated univariate polynomials with integer coefficients in the interval $[-15, +15]$ and degrees at most t. Table 3 gives the times for randomly generated bivariate polynomials with integer coefficients in the interval $[-15, +15]$ and degrees in each variable at most t. All matrices generated had non-zero determinants.

Table 1

PDET for Integer Entries

k \ m	5	10	15
4	0.12	0.51	1.89
8	0.14	0.82	2.84
16	0.21	1.30	4.20
32	0.39	2.33	8.08

Table 2

PDET for Univariate Polynomial Entries

t \ m	4	6	8	10
1	0.20	0.82	1.58	3.91
2	0.32	1.47	2.97	7.78
3	0.53	2.37	4.72	12.35
4	0.68	3.22	6.16	16.42
5	0.81	3.92	7.92	21.56

Table 3

PDET for Bivariate Polynomial Entries

t \ m	3	4	5
1	0.48	0.96	3.30
2	1.79	3.28	12.01
3	3.32	14.64	28.60

3.2 Solving Systems of Linear Equations

Tables 4, 5, and 6 give the times for solving a single system of linear equations with an m by m coefficient matrix. The entries of the m by m+1 matrix inputs to PLES are of the same classes as for Tables 1, 2, and 3, respectively. All system matrices generated had unique solutions.

Table 4

PLES for Integer Entries

k \ m	5	10	15
4	0.23	1.09	3.21
8	0.38	1.56	4.49
16	0.57	3.12	8.56
32	1.11	6.58	18.45

Table 5

PLES for Univariate Polynomial Entries

t \ m	4	6	8	10
1	1.29	3.44	8.21	15.04
2	2.68	7.14	19.21	33.96
3	4.33	12.41	33.01	64.01
4	5.66	16.59	45.60	89.76
5	7.91	22.65	64.24	119.70

Table 6

PLES for Bivariate Polynomial Entries

t \ m	3	4	5
1	5.79	11.52	24.79
2	24.80	61.29	115.95
3	46.22	129.63	263.75

3.3 Matrix Inversion

Tables 7, 8, and 9 give the times for inverting an m by m matrix. The entries of these matrix inputs to MINV are of the same classes as for Tables 1, 2, and 3, respectively. The inverses existed for all matrices tried.

Table 7

MINV for Integer Entries

k \ m	5	10	15
4	0.47	3.26	10.37
8	0.71	4.76	18.35
16	1.10	8.96	33.88
32	2.08	21.86	106.98

Table 8

MINV for Univariate Polynomial Entries

t \ m	4	6	8	10
1	2.39	8.95	28.89	60.78
2	4.62	18.84	66.93	151.25
3	7.43	32.15	114.32	---
4	11.25	47.39	163.38	---
5	14.46	81.71	235.29	---

Table 9

MINV for Bivariate Polynomial Entries

t \ m	3	4	5
1	9.78	26.32	57.89
2	33.17	91.97	242.67
3	62.14	199.31	---

The three cases missing from Table 8 and the one case missing from Table 9 were omitted due to space restrictions. For example, the inverse of the 10 by 10 matrix of univariate polynomials of degree at most 5 would require at least 26000 cells for computer representation. The inverse of the 5 by 5 matrix of bivariate polynomials with degree in each variable at most 3 would require over 22500 cells.

4. A TEST PROGRAM

In this section a test program for the SAC-1 Linear Algebra System is given, accompanied by two simple subprograms for reading in and writing out matrices. Also given is the output for a typical run. This program can be used to test out a user's implementation of the Polynomial Linear Algebra System. In fact, the test run described below employed all the programs of this system except those for classical matrix arithmetic.

The main program can read in any number of matrices C with multivariate polynomial entries and perform the desired computation for C from among the following:

(a) compute the determinant D of C by PDET;

(b) compute a general solution (D,Y,Z) for the system of linear equations AX = B by PLES, where C is the augmented system matrix (A,B);

(c) compute the inverse of C by MINV;

(d) compute a basis for the null space of C by NULSP.

The main loop begins at the statement labeled 1, reading in integers K, M, N, NN. If K = 0, the run is terminated, after the final erasures. If K > 0, an M by N matrix C is read in, using MREAD, and is written out, using MWRITE. (These input-

output routines, written in USASI Fortran, are described below.) The values 1, 2, 3, 4 of K specify computations (a), (b), (c), (d), respectively, and require the following corresponding conventions for M, N, and NN:

(a') K = 1: M = N and NN is ignored;

(b') K = 2: the coefficient matrix A is M by NN;

(c') K = 3: M = N and NN is ignored;

(d') K = 4: NN is ignored.

Following each computation the results are written out, again using MWRITE for matrices included in the output.

The test program includes the following features which are common to SAC-1 main programs employing the Polynomial System, [4], and the Modular Arithmetic System, [6]. The four Common blocks, TR1, TR2, TR3, and TR4, introduced in [2], [4], [3], and [7], respectively, are declared. All the elements of these common blocks, and all variables and function subprograms used by the program, are declared type integer. The arrays SPACE and ARRAY are dimensioned. SPACE is used by BEGIN to initially construct the available space list AVAIL and ARRAY is used by GENPR to construct the list PRIME of Fortran integer primes. The size of these arrays in general depends on the size of the computation being

performed. The values of the implementation parameters used are those for the UNIVAC 1108 implementation of SAC-1. Specifically, since two words per cell are required, SPACE is dimensioned to 10000 to obtain the 5000 cells. The base BETA for the infinite-precision integers is 2^{33}; hence, 33 bits of a 36-bit word are used for each digit. Also, each prime generated by GENPR is $\geq 2^{31} + 1$. The standard input and output units are IN = 5 and OUT = 6, respectively. The symbol list SYMLST for the Polynomial System is initialized to the null list.

We now give functional descriptions for the two matrix input and output routines.

C = MREAD(u,m,n)

u is an input unit number and m and n are positive integers; all are Fortran integers. C is an m by n matrix, represented rowwise, whose entries are multivariate polynomials with infinite-precision integer coefficients. The polynomial (or integer) entries C_{ij} of C are read in row-by-row from row 1 to row m in the order: $C_{11},\ldots,C_{1n},C_{21},\ldots,C_{2n},\ldots,C_{m1},\ldots,C_{mn}$. These elements are read from consecutive records on unit u using PREAD, each beginning in column 1 of the record immediately following the last record of the preceding element. If an attempt to read an element

results in reading an end-of-file or in a syntactic error, an error message is printed and the run is terminated. Note that by interchanging row and column, and m and n, in the above description, a columnwise representation can be read in.

MWRITE(u,C)

u is an output unit number, a Fortran integer, and C is an m by n matrix of multivariate polynomial entries with infinite-precision integer coefficients, assumed to be represented rowwise. The elements C_{ij} of C are written out row-by-row from row 1 to row m in the order: $C_{11},\ldots,C_{1n},C_{21},\ldots,C_{2n},\ldots,C_{m1},\ldots,C_{mn}$. Each polynomial (or integer) is written on unit u using PWRITE. The elements of row i are preceded by a record of blanks, followed by a record containing the identification "ROW i", followed by another blank record. Note that C may be represented columnwise, in which case column i will be mislabeled "ROW i".

The routines MREAD and MWRITE are not part of the Linear Algebra System but are included for the user's convenience.

Three test cases were run on the UNIVAC 1108, all for matrices C with univariate polynomial entries: (1) a determinant calculation, (2) the solution of a system of linear equations, whose coefficient matrix has a non-trivial null space, and (3) a matrix inversion.

The total execution time was approximately 22 seconds, the times for (1), (2), and (3) being roughly 2.5, 7.5 and 12 seconds, respectively. A separate null space basis computation is not included, since a null space basis was computed as part of case (2).

Four cards were read at the statement labeled 1 containing the values:

K	M	N	NN
1	5	5	
2	3	6	5
3	5	5	
0			

Immediately following each of the first three cards were the cards containing the M*N polynomials. For these cases, each polynomial was contained on a single card. The 25, 18, and 25 polynomials read in for cases (1), (2), and (3), respectively, are listed below as part of the program output in the format specified by MWRITE. This test run can be duplicated by punching these polynomials on cards in the format and order given in the output, remembering to begin each in column 1, and preceding each group of polynomials for a matrix with a card containing the values of K, M, N, and NN.

Test Program

```
      INTEGER GENPR,LENGTH,MINV,MREAD,NULSP,PDET,PLES
      INTEGER ARRAY,C,D,G,I,IN,K,L,M,N,NN,OUT,SPACE,X,Y,Z
      INTEGER AVAIL,STAK,RECORD,BETA,SYMLST,PRIME
      DIMENSION SPACE(10000),ARRAY(50)
      COMMON /TR1/AVAIL,STAK,RECORD(72)
      COMMON /TR2/SYMLST
      COMMON /TR3/BETA
      COMMON /TR4/PRIME
      CALL BEGIN(SPACE,10000)
      PRIME=GENPR(ARRAY,50,2**31+1)
      BETA=2**33
      SYMLST=0
      IN=5
      OUT=6
      I=0
1     READ 903, K,M,N,NN
903   FORMAT(4I5)
      IF (K.EQ.0) GO TO 60
      I=I+1
      PRINT 905, I
905   FORMAT(9H1CASE NO.,I3)
      C=MREAD(IN,M,N)
      PRINT 907, M,N
907   FORMAT(//4H THE,I4,3H BY,I4,13H INPUT MATRIX/)
      CALL MWRITE(OUT,C)
      GO TO (10,20,30,40), K
10    PRINT 910
910   FORMAT(//14H DETERMINANT D/)
      D=PDET(C)
      CALL PWRITE(OUT,D)
      CALL PERASE(D)
      GO TO 50
20    PRINT 920, NN
920   FORMAT(//30H LINEAR SYSTEM SOLUTION  (NN =,I4,1H))
      G=PLES(NN,C)
      IF (G.EQ.0) GO TO 26
      CALL DECAP(D,G)
      CALL DECAP(Y,G)
      CALL DECAP(Z,G)
      PRINT 921
921   FORMAT(/14H DENOMINATOR D/)
      CALL PWRITE(OUT,D)
      CALL PERASE(D)
      PRINT 922
922   FORMAT(/22H PARTICULAR SOLUTION Y)
      CALL MWRITE(OUT,Y)
      CALL MERASE(Y)
23    IF (Z.EQ.0) GO TO 25
      PRINT 923
923   FORMAT(/19H NULL SPACE BASIS Z)
      CALL MWRITE(OUT,Z)
      CALL MERASE(Z)
      GO TO 50
25    PRINT 925
925   FORMAT(22H NULL SPACE IS TRIVIAL)
      GO TO 50
26    PRINT 926
```

```
926   FORMAT(/15H DOES NOT EXIST)
      GO TO 50
30    PRINT 930
930   FORMAT(//15H MATRIX INVERSE)
      X=MINV(C)
      IF (X.EQ.0) GO TO 26
      CALL DECAP(D,X)
      CALL DECAP(Y,X)
      PRINT 931
931   FORMAT(/23H DETERMINANT D (+ OR -)/)
      CALL PWRITE(OUT,D)
      CALL PERASE(D)
      PRINT 932
932   FORMAT(/19H ADJOINT Y (+ OR -))
      CALL MWRITE(OUT,Y)
      CALL MERASE(Y)
      GO TO 50
40    Z=NULSP(C)
      GO TO 23
50    CALL MERASE(C)
      GO TO 1
60    CALL ERLA(PRIME)
      CALL ERASE(SYMLST)
      L=LENGTH(AVAIL)
      PRINT 960, L
960   FORMAT(//16H LENGTH(AVAIL) =,I8)
      STOP
      END
```

Matrix Input/Output Programs

```
      INTEGER FUNCTION MREAD(U,M,N)
      INTEGER A,B,C,I,J,M,N,U
      INTEGER INV,PFL,PREAD
      A=0
      DO 4 I=1,M
      B=0
      DO 3 J=1,N
      C=PREAD(U)
      IF (C.EQ.-1) GO TO 5
      IF (C.EQ.-2) GO TO 6
3     B=PFL(C,B)
4     A=PFL(INV(B),A)
      MREAD=INV(A)
      RETURN
5     PRINT 905 ,I,J
905   FORMAT(/10H EOF AT A(,I4,1H,,I4,1H))
      STOP
6     PRINT 906 ,I,J
906   FORMAT(/20H SYNTAX ERROR FOR A(,I4,1H,,I4,1H))
      STOP
      END
```

```
      SUBROUTINE MWRITE(U,A)
      INTEGER A,B,C,I,M,T,U
      INTEGER LENGTH
      M=LENGTH(A)
      T=A
      I=0
2     I=I+1
      WRITE (U,901) I
901   FORMAT(/4H ROW,I4/)
      CALL ADV(B,T)
3     CALL ADV(C,B)
      CALL PWRITE(U,C)
      IF (B.NE.0) GO TO 3
      IF (T.NE.0) GO TO 2
      RETURN
      END
```

Test Program Output

```
CASE NO.  1

THE   5 BY   5 INPUT MATRIX

ROW   1

(-13X**5-11X**3+1X**2-11X**1-8X**0)
(-2X**5+1X**4+10X**3-7X**2+1X**1+10X**0)
(+7X**5-12X**4+5X**3+7X**2+9X**1+5X**0)
(+7X**5-14X**4+12X**3+2X**2-9X**1+14X**0)
(+1X**5-13X**4+7X**3+7X**2-11X**1+14X**0)

ROW   2

(-9X**4+12X**3-3X**2-9X**1+4X**0)
(-12X**5+11X**4+9X**3-8X**2+12X**1)
(+14X**5-12X**4-4X**3-1X**2+6X**1-4X**0)
(+15X**5+3X**4+4X**3-6X**2-2X**1+4X**0)
(+15X**5+7X**4+6X**3+12X**2-12X**1-9X**0)

ROW   3

(-7X**5-11X**4+11X**3+3X**1+10X**0)
(+5X**5-11X**4+13X**3-9X**2-5X**1+8X**0)
(-13X**5-6X**4-6X**3+13X**1+15X**0)
(+15X**5-10X**4-5X**2-14X**1+1X**0)
(-8X**5+12X**4+4X**3+10X**2+2X**1+8X**0)

ROW   4

(+11X**5+7X**4-8X**3-15X**2-4X**1-5X**0)
(+3X**5+7X**4-9X**2-2X**1-13X**0)
(+9X**5-12X**4-1X**3-2X**2-9X**1-5X**0)
(+11X**5-15X**4+14X**3-10X**2+7X**1+5X**0)
(-11X**5+12X**4+10X**3-9X**2+11X**1)

ROW   5

(+1X**5+5X**4+9X**2+7X**1)
(-10X**5+6X**4-2X**2-7X**1-7X**0)
(+14X**5+12X**4+2X**3-9X**2+3X**1+3X**0)
(+10X**5+6X**4+13X**3+15X**2+9X**1+1X**0)
(-5X**5-3X**4+1X**3+1X**2-11X**1+11X**0)


DETERMINANT D

(+1317513X**25-13914336X**24+30773300X**23-46871200X**22+6944510X**21-3027837X**20-17268679X**19+27071738X**18-41847964X**17+3224836X**16+8717685X**15-25399734X**14+3099302X**13+18674812X**12-1801567X**11+25341253X**10+25631958X**9+13575937X**8+19152565X**7+3909794X**6-1527515X**5+651547X**4-4393987X**3-1701176X**2+867808X**1-182943X**0)
```

CASE NO. 2

THE 3 BY 6 INPUT MATRIX

ROW 1

(+6X**3+12X**2-12X**1-9X**0)
(+3X**3+10X**2+15X**1+7X**0)
(-7X**3-11X**2+11X**1)
(+13X**3-9X**2-5X**1+8X**0)
(+13X**3+15X**2+5X**1-11X**0)
(-13X**3-6X**2-6X**1)

ROW 2

(-5X**2-14X**1+1X**0)
(+2X**3+8X**2+15X**1-10X**0)
(-8X**3+12X**2+4X**1+10X**0)
(-8X**3-15X**2-4X**1-5X**0)
(-2X**3-13X**2+11X**1+7X**0)
(+3X**3+7X**2-9X**0)

ROW 3

(-1X**3-2X**2-9X**1-5X**0)
(+7X**3+5X**2+9X**1-12X**0)
(+11X**3-15X**2+14X**1-10X**0)
(+10X**3-9X**2+11X**1)
(+7X**3-11X**1+12X**0)
(+1X**3+5X**2+9X**0)

LINEAR SYSTEM SOLUTION (NN = 5)

DENOMINATOR D

(+478X**9+1416X**8+1446X**7-1345X**6-6609X**5+2725X**4-5945X**3+2001X**2
-2782X**1-2260X**0)

PARTICULAR SOLUTION Y

ROW 1

(-1270X**9-1771X**8-2473X**7+7867X**6+2583X**5+9514X**4-629X**3-801X**2+
1992X**1)

ROW 2

(+121X**9-52X**8-2604X**7-492X**6+840X**5+5209X**4+3183X**3+2724X**2+130
2X**1)

ROW 3

(-149X**9-271X**8+591X**7+1852X**6+2073X**5-54X**4+314X**3-3301X**2+2793
X**1+2034X**0)

```
ROW    4

+0

ROW    5

+0

NULL SPACE BASIS 2

ROW    1

(+1570X**9+2596X**8+5406X**7-4239X**6+7247X**5-1419X**4-3025X**3+3062X**
2-2866X**1+1410X**0)
(+1108X**9+3010X**8+4327X**7-7404X**6-9135X**5-3163X**4+12652X**3-5826X*
*2+7648X**1-1090X**0)

ROW    2

(+112X**9-168X**8+1840X**7+1340X**6+400X**5+4224X**4-3691X**3+2066X**2-1
527X**1-770X**0)
(+322X**9+499X**8+2251X**7+3662X**6-5964X**5-6022X**4-404X**3-7194X**2+5
051X**1+2150X**0)

ROW    3

(+506X**9+2160X**8+3990X**7+4135X**6+1983X**5+5827X**4-29X**3+2960X**2+4
096X**1+219X**0)
(+200X**9+1185X**8+1728X**7-1051X**6-3097X**5+3305X**4+10456X**3-3212X**
2-5169X**1+677X**0)

ROW    4

(-478X**9-1416X**8-1446X**7+1345X**6+6609X**5-2725X**4+5945X**3-2001X**2
+2782X**1+2260X**0)
+0

ROW    5

+0
(-478X**9-1416X**8-1446X**7+1345X**6+6609X**5-2725X**4+5945X**3-2001X**2
+2782X**1+2260X**0)
```

```
CASE NO.  3

THE   5 BY   5 INPUT MATRIX

ROW   1

(-6X**2+9X**1-10X**0)
(-6X**2-10X**1-1X**0)
(-11X**2+10X**1-6X**0)
(-14X**2+14X**1+9X**0)
(+8X**2+1X**1+7X**0)

ROW   2

(-9X**2+4X**1+11X**0)
(-10X**2-4X**1+15X**0)
(+12X**2+8X**1-13X**0)
(+1X**1+5X**0)
(+4X**0)

ROW   3

(+4X**2+3X**1-12X**0)
(+7X**2+15X**1-8X**0)
(-15X**2+7X**1-3X**0)
(-11X**2+15X**1-10X**0)
(+2X**2+5X**1+8X**0)

ROW   4

(-11X**2+9X**1-7X**0)
(-15X**2-3X**0)
(+1X**2+10X**1+12X**0)
(+15X**2+8X**1+15X**0)
(+3X**2+9X**1-3X**0)

ROW   5

(-9X**2+8X**1-6X**0)
(+8X**2+13X**1-3X**0)
(-5X**2+13X**0)
(+12X**2+12X**1+11X**0)
(-11X**2-10X**1-8X**0)


MATRIX INVERSE

DETERMINANT D (+ OR -)

(+106804X**10-2146701X**9-240343X**8+3218301X**7-1667524X**6-1727090X**5
+901265X**4+3390663X**3+2171211X**2+751912X**1+81892X**0)

ADJOINT Y (+ OR -)

ROW   1
```

```
(-26858X**8+67859X**7+219338X**6+9[illegible]9X**5-1126Y**4-69475X**3-61235X**2-
10532X**1+156^**0)
(+19268X**8+114213X**7+35263X**6+59189X**5+16459X**4+192608X**3+93412X**
2-25340X**1-6768X**0)
(+38990X**8+30263X**7-60294X**6+38016X**5+65702Y**4-62476X**3-122762X**2
-71552X**1-9576X**0)
(+11066X**8+1439X**7-57276X**6-151209X**5-134443X**4-4455X**3+49061X**2+
52860X**1+7148X**0)
(-9426X**8+88151X**7+96009X**6-68195X**5-7443X**4+60764X**3+1912X**2-918
04X**1-15504X**0)

ROW   2

(+39057X**8-15487X**7-68383X**6+63116X**5+56602Y**4-77115X**3-64193X**2-
43212X**1-10211X**0)
(-40876X**8-16618X**7+34827X**6+8591X**5-78900X**4+40987Y**3+22280X**2+7
5468X**1+16217X**0)
(-66291X**8-20869X**7+104534X**6-15908X**5-94287X**4+36516Y**3+122383X**
2+87533X**1+15758X**0)
(-20721X**8+49214X**7+15800X**6-71659X**5+42230Y**4-4434X**3+26922X**2-2
3994X**1+381X**0)
(+10701X**8-54899X**7+12510X**6+61749X**5-48954Y**4-2956X**3-45603X**2+8
6969X**1+14789X**0)

ROW   3

(+12404X**8+51035X**7+12309X**6+43381X**5-90321Y**4-163480X**3-133486X**
2-52798X**1-11538X**0)
(-10712X**8-118762X**7+109462X**6-25711X**5-48943X**4-70760X**3-11680X**
2+7970X**1+6070X**0)
(-26000X**8-10928X**7+83787X**6-31573X**5-44634Y**4-12734X**3+43981X**2+
29158X**1+9024X**0)
(-8968X**8+37534X**7-19651X**6-107873X**5+10486Y**4+110987Y**3+130576X**
2+52690X**1+13070X**0)
(+1848X**8+25658X**7+12074X**6-39015X**5-28991X**4-31416Y**3-44266X**2-1
0240X**1-2938X**0)

ROW   4

(+7869X**8+19026X**7+56194X**6-87368X**5-47299Y**4+127893Y**3+73617X**2+
34771X**1+5867X**0)
(-14700X**8+100598X**7-23449X**6+25788X**5+46653Y**4+17752Y**3+100326X**
2-241X**1-4137X**0)
(-18311X**8+52256X**7-1518X**6-77243X**5+32802Y**4+39535Y**3+2473X**2-35
631Y**1-7378X**0)
(-721X**8-71398X**7-8102X**6+59720X**5-26704Y**4+6286Y**3-39616Y**2+1516
5X**1+2155X**0)
(+2197X**8-6329X**7-3854X**6+50764X**5-44917Y**4-24093Y**3+150499Y**2-60
66X**1-5121X**0)

ROW   5

(+53326X**8-82495X**7-58940X**6+157297X**5-54037Y**4+61338Y**3-43070X**2
-12807X**1-6970X**0)
(-56660X**8+59371X**7+54821X**6-151675X**5+137651Y**4-52976Y**3+131035X**
2+12741X**1+3170X**0)
(-88270X**8+32318X**7+122558X**6-140948X**5-791Y**4+71001Y**3+118960X**2
```

```
+16952X**1+5792X**0)
(-20834X**8-50622X**7+96507X**6+67616X**5-158745X**4+9222X**3+29817X**2+
63453X**1+18698X**0)
(+7342X**8+66051X**7-123755X**6+17001X**5+194199X**4-183026X**3+45437X**
2-61921X**1-15970X**0)


LENGTH(AVAIL) =     5000
```

5. FORTRAN PROGRAM LISTINGS

```
      INTEGER FUNCTION CBDET(A)
      INTEGER A,B,C,D,J,K,L,M,U
      INTEGER ICOMP,IFACT,IPROD,LENGTH,PFA,PNORMF
10    B=A
      M=LENGTH(B)
      J=PFA(2,0)
20    CALL ADV(C,B)
      K=0
30    CALL ADV(D,C)
      L=PNORMF(D)
      U=ICOMP(L,K)
      IF (U.NE.1) GO TO 31
      CALL ERLA(K)
      K=L
      GO TO 32
31    CALL ERLA(L)
32    IF (C.NE.0) GO TO 30
40    L=IPROD(J,K)
      CALL ERLA(J)
      CALL ERLA(K)
      J=L
      IF (B.NE.0) GO TO 20
50    K=IFACT(M)
      CBDET=IPROD(K,J)
      CALL ERLA(K)
      CALL ERLA(J)
      RETURN
      END



      INTEGER FUNCTION CDET(P,A)
      INTEGER A,BP,BPP,D,E,F,G,H,I,P,T,V
      INTEGER CDIF,CPROD,CRECIP,FIRST,INV,PFL,TAIL
1     BP=A
      D=1
      I=0
2     BPP=BP
      BP=0
3     CALL DECAP(F,BPP)
      CALL DECAP(E,F)
      IF (E.NE.0) GO TO 4
      IF (F.NE.0) BP=PFL(F,BP)
      I=I+1
      IF (BPP.NE.0) GO TO 3
      CDET=0
      CALL MCPERS(BP)
      RETURN
4     D=CPROD(P,E,D)
```

```
      IF (F.EQ.0) GO TO 8
      E=CRECIP(P,E)
      G=F
5     CALL ALTER(CPROD(P,E,FIRST(G)),G)
      G=TAIL(G)
      IF (G.NE.0) GO TO 5
6     IF (BPP.EQ.0) GO TO 8
      CALL DECAP(G,BPP)
      CALL DECAP(J,G)
      BP=PFL(G,BP)
      IF (U.EQ.0) GO TO 6
      H=F
7     T=CDIF(P,FIRST(G),CPROD(P,U,FIRST(H)))
      CALL ALTER(T,G)
      G=TAIL(G)
      H=TAIL(H)
      IF (G.NE.0) GO TO 7
      GO TO 6
8     CALL ERLA(F)
      BP=INV(BP)
      IF (BP.NE.0) GO TO 2
      U=0
      CALL QR(U,I,2)
      IF (I.EQ.1) D=P-D
      CDET=D
      RETURN
      END



      INTEGER FUNCTION CPDET(PP,AA)
      INTEGER A,AA,ASTAR,B,D,DP,DSTAR,E,H,I,J,M,P,PP,RET,S,T,U
      INTEGER CDET,CDIF,CPDEG,CPINT,CPPROD,DBRRE,LENGTH,MCPEVL,MCPNV
      INTEGER MZERO,PFA
      A=AA
      P=PP
      RET=1
      GO TO 10
1     CPDET=D
      RETURN
10    IF (MZERO(A).EQ.0) GO TO 11
      D=0
      CALL MCPERS(A)
      GO TO (1,51), RET
11    S=MCPNV(A)
20    IF (S.GT.0) GO TO 30
      D=CDET(P,A)
      GO TO (1,51), RET
30    H=0
      M=LENGTH(A)
      J=0
      DO 31 I=1,M
31    J=PFA(M+1-I,J)
      B=DBRRE(J,A)
      CALL ERLA(J)
```

```
      I=0
      E=PFA(0,PFA(1,0))
      D=0
40    IF (I.LT.P) GO TO 42
      PRINT 41
41    FORMAT(52H ELEMENTS OF GF(P) EXHAUSTED. ALGORITHM CPDET FAILS.)
      STOP
42    ASTAR=MCPEVL(P,A,I,S)
50    CALL STACK3(A,S,I)
      CALL STACK3(D,E,B)
      CALL STACK2(H,RET)
      RET=2
      A=ASTAR
      GO TO 10
51    DSTAR=D
      CALL UNSTK2(H,RET)
      CALL UNSTK3(D,E,B)
      CALL UNSTK3(A,S,I)
      IF (DSTAR.NE.0) H=1
      IF (H.EQ.0) GO TO 70
60    DP=CPINT(P,D,I,DSTAR,E,S)
      CALL CPERAS(D)
      IF (S.GT.1) CALL CPERAS(DSTAR)
      D=DP
70    IF (CPDEG(E).NE.B) GO TO 80
      CALL MCPERS(A)
      CALL ERLA(E)
      GO TO (1,51), RET
80    T=PFA(1,PFA(1,PFA(CDIF(P,0,I),0)))
      U=CPPROD(P,T,E)
      CALL CPERAS(T)
      CALL CPERAS(E)
      E=U
      I=I+1
      GO TO 40
      END



      INTEGER FUNCTION CPRRE(P,CC)
      INTEGER A,B,C,CC,CSTAR,D,DP,DSTAR,E,EP,H,I,ISTAR,J,JSTAR,N,P
      INTEGER R,RET,RSTAR,S,T,U,V,VP,VSTAR,W,WSTAR,Z,ZP
      INTEGER CDIF,CPDEG,CPDIF,CPINT,CPNEG,CPPROD,CRRE,DBRRE,FIRST
      INTEGER LENGTH,MCPEQ,MCPEVL,MCPINT,MCPMPY,MCPNV,MTRAN,MZCON
      INTEGER MZERO,NULCON,PFA,PFL,VCOMP
      C=CC
      RET=1
      GO TO 10
1     CPRRE=W
      RETURN
10    S=MCPNV(C)
      IF (S.GT.0) GO TO 20
      W=CRRE(P,C)
      GO TO (1,41), RET
20    R=0
```

```
      A=P
      H=0
30    A=A-1
      IF (A.GE.0) GO TO 32
      PRINT 31
31    FORMAT(52H ELEMENTS OF GF(P) EXHAUSTED. ALGORITHM CPRRE FAILS.)
      STOP
32    CSTAR=MCPEVL(P,C,A,S)
      IF (MZERO(CSTAR).EQ.0) GO TO 40
      CALL MCPERS(CSTAR)
      GO TO 30
40    CALL STACK3(C,S,A)
      CALL STACK3(R,I,J)
      CALL STACK3(D,V,E)
      CALL STACK3(H,B,RET)
      C=CSTAR
      RET=2
      GO TO 10
41    WSTAR=W
      CALL UNSTK3(H,B,RET)
      CALL UNSTK3(D,V,E)
      CALL UNSTK3(R,I,J)
      CALL UNSTK3(C,S,A)
      CALL DECAP(JSTAR,WSTAR)
      CALL DECAP(ISTAR,WSTAR)
      CALL DECAP(DSTAR,WSTAR)
      CALL DECAP(VSTAR,WSTAR)
50    RSTAR=LENGTH(JSTAR)
      IF (RSTAR-R) 60,51,70
51    IF (VCOMP(JSTAR,J)) 70,52,60
52    IF (VCOMP(ISTAR,I)) 70,53,60
53    CALL ERLA(JSTAR)
      CALL ERLA(ISTAR)
      GO TO 80
60    CALL ERLA(JSTAR)
      CALL ERLA(ISTAR)
      IF (VSTAR.NE.0) CALL MCPERS(VSTAR)
      IF (S.GT.1) CALL CPERAS(DSTAR)
      GO TO 30
70    IF (R.EQ.0) GO TO 71
      CALL ERLA(J)
      CALL ERLA(I)
      CALL CPERAS(D)
      IF (V.NE.0) CALL MCPERS(V)
      CALL CPERAS(E)
71    R=RSTAR
      J=JSTAR
      I=ISTAR
      D=0
      V=0
      IF (VSTAR.NE.0) V=MZCON(VSTAR)
      E=PFA(0,PFA(1,0))
80    DP=CPINT(P,D,A,DSTAR,E,S)
      IF (S.GT.1) CALL CPERAS(DSTAR)
      IF (V.EQ.0) GO TO 81
```

```
      VP=MCPINT(P,V,A,VSTAR,E,S)
      CALL MCPERS(VSTAR)
81    T=PFA(1,PFA(1,PFA(CDIF(P,C,A),0)))
      EP=CPPROD(P,T,E)
      CALL CPERAS(T)
      CALL CPERAS(E)
      E=EP
      IF (H.EQ.0) GO TO 90
      CALL CPERAS(D)
      D=DP
      IF (V.EQ.0) GO TO 120
      CALL MCPERS(V)
      V=VP
      GO TO 120
90    T=CPDIF(P,D,DP)
      CALL CPERAS(D)
      D=DP
      U=1
      IF (V.EQ.0) GO TO 91
      U=MCPEQ(P,V,VP)
      CALL MCPERS(V)
      V=VP
91    IF (U.EQ.1 .AND. T.EQ.0) GO TO 100
      CALL CPERAS(T)
      GO TO 30
100   N=LENGTH(FIRST(C))
      IF (R.NE.N) GO TO 101
      H=1
      GO TO 110
101   DP=CPNEG(P,D)
      Z=NULCON(N,J,DP,V)
      ZP=MTRAN(Z)
      T=MCPMPY(P,C,ZP)
      U=MZERO(T)
      CALL MCPERS(ZP)
      CALL CPERAS(DP)
      CALL MCPERS(Z)
      CALL MCPERS(T)
      IF (U.EQ.0) GO TO 30
      H=1
110   B=DBRRE(J,C)
120   IF (CPDEG(E).LE.B) GO TO 30
130   CALL CPERAS(E)
      CALL MCPERS(C)
      W=PFL(J,PFL(I,PFL(D,PFL(V,0))))
140   GO TO (1,41), RET
      END
```

```
      INTEGER FUNCTION CRRE(P,A)
      INTEGER A,B,BP,BPP,D,E,F,G,H,HB,I,IP,IPP,J,K,L,M,N,P
      INTEGER S,T,U,V,W,X,Z
      INTEGER CDIF,CONC,CPROD,CRECIP,FIRST,INV,LENGTH,PFA,PFL,TAIL
10    BP=A
      M=LENGTH(BP)
      N=LENGTH(FIRST(BP))
      I=0
      J=0
      D=1
      V=0
      B=0
      IP=0
      DO 11 K=1,M
      U=M+1-K
11    IP=PFA(U,IP)
      Z=0
20    Z=Z+1
      BPP=BP
      BP=0
      IPP=IP
      IP=0
30    CALL DECAP(F,BPP)
      CALL DECAP(E,F)
      CALL DECAP(K,IPP)
      IF (E.NE.0) GO TO 40
      IP=PFA(K,IP)
      IF (F.NE.0) BP=PFL(F,BP)
      IF (BPP.NE.0) GO TO 30
      GO TO 80
40    D=CPROD(P,E,D)
      J=PFA(Z,J)
      I=PFA(K,I)
      B=PFL(F,B)
      IF (F.EQ.0) GO TO 80
      E=CRECIP(P,E)
      G=F
50    CALL ALTER(CPROD(P,E,FIRST(G)),G)
      G=TAIL(G)
      IF (G.NE.0) GO TO 50
60    IF (BPP.EQ.0) GO TO 80
      CALL DECAP(G,BPP)
      CALL DECAP(U,G)
      BP=PFL(G,BP)
      IF (U.EQ.0) GO TO 60
      H=F
70    T=CDIF(P,FIRST(G),CPROD(P,U,FIRST(H)))
      CALL ALTER(T,G)
      G=TAIL(G)
      H=TAIL(H)
      IF (G.NE.0) GO TO 70
      GO TO 60
80    BP=INV(BP)
      IP=CONC(INV(IP),IPP)
      IF (BP.NE.0) GO TO 20
```

```
      I=CONC(INV(I),IP)
      IF (BPP.NE.0) CALL ERASE(BPP)
90    IF (B.NE.0) GO TO 91
      U=V
      V=0
      GO TO 150
91    CALL DECAP(H,B)
      H=INV(H)
      HBAR=0
      M=0
      Z=N
      L=J
100   IF (H.EQ.0) GO TO 110
      CALL DECAP(J,H)
      K=FIRST(L)
      IF (Z.NE.K) GO TO 101
      M=PFA(U,M)
      L=TAIL(L)
      GO TO 102
101   HBAR=PFA(U,HBAR)
102   Z=Z-1
      GO TO 100
110   T=V
      HBAR=INV(HBAR)
120   IF (M.EQ.0) GO TO 140
      CALL DECAP(U,M)
      IF (T.EQ.0) GO TO 120
      CALL ADV(S,T)
      H=HBAR
130   IF (S.EQ.0) GO TO 120
      CALL ADV(X,S)
      W=FIRST(H)
      W=CDIF(P,W,CPROD(P,U,X))
      CALL ALTER(W,H)
      H=TAIL(H)
      GO TO 130
140   IF (HBAR.NE.0) V=PFL(HBAR,V)
      GO TO 90
150   IF (U.EQ.0) GO TO 170
      CALL DECAP(T,U)
      S=0
160   CALL DECAP(E,T)
      G=CPROD(P,D,E)
      S=PFA(G,S)
      IF (T.NE.0) GO TO 160
      V=PFL(S,V)
      GO TO 150
170   V=INV(V)
      J=INV(J)
      CRRE=PFL(J,PFL(I,PFA(D,PFL(V,0))))
      RETURN
      END
```

```
      INTEGER FUNCTION DBRRE(J,C)
      INTEGER C,D,E,F,G,H,I,J,K,L,T,U,X,Y
      INTEGER CPDEG,FIRST,MTRAN,PFA,TAIL
10    X=MTRAN(C)
      Y=X
      K=J
      E=0
      F=0
      L=0
      I=0
20    IF (Y.EQ.0) GO TO 50
      CALL ADV(U,Y)
      I=I+1
      IF (K.EQ.0) GO TO 40
      IF (I.NE.FIRST(K)) GO TO 40
      G=0
30    CALL ADV(T,J)
      IF (T.EQ.0) GO TO 31
      D=CPDEG(T)
      IF (D.GT.G) G=D
31    IF (U.NE.0) GO TO 30
      L=PFA(G,L)
      K=TAIL(K)
      GO TO 20
40    CALL ADV(T,J)
      IF (T.EQ.0) GO TO 41
      D=CPDEG(T)
      IF (D.GT.F) F=D
41    IF (U.NE.0) GO TO 40
      GO TO 20
50    CALL DECAP(D,L)
      E=E+D
      IF (D.LT.G) G=D
      IF (L.NE.0) GO TO 50
60    DBRRE=E
      H=F-G
      IF (H.GT.0) DBRRE=DBRRE+H
      CALL MCPERS(X)
      RETURN
      END



      INTEGER FUNCTION MCMPY(P,A,B)
      INTEGER A,B,C,D,E,P,R,S,T,V,W,X,Y
      INTEGER CINV,CPROD,CSUM,FIRST,PFA,PFL,TAIL
1     X=CINV(A)
      Y=CINV(B)
      C=0
      V=X
2     CALL ADV(R,V)
      D=0
      W=Y
3     S=R
      CALL ADV(T,W)
```

```
      E=0
4     E=CSUM(P,E,CPROD(P,FIRST(S),FIRST(T)))
      S=TAIL(S)
      T=TAIL(T)
      IF (S.NE.0) GO TO 4
      D=PFA(E,D)
      IF (W.NE.0) GO TO 3
      C=PFL(D,C)
      IF (V.NE.0) GO TO 2
5     CALL ERASE(X)
      CALL ERASE(Y)
      MCMPY=C
      RETURN
      END


      INTEGER FUNCTION MCPEVL(P,A,B,S)
      INTEGER A,B,C,D,P,S,U,W,X
      INTEGER CPEVAL,INV,PFA,PFL
1     X=A
      C=0
2     CALL ADV(W,X)
      D=0
3     CALL ADV(U,W)
      IF (U.NE.0) U=CPEVAL(P,U,B)
      IF (S.EQ.1) D=PFA(U,D)
      IF (S.NE.1) D=PFL(U,D)
      IF (W.NE.0) GO TO 3
4     C=PFL(INV(D),C)
      IF (X.NE.0) GO TO 2
5     MCPEVL=INV(C)
      RETURN
      END


      INTEGER FUNCTION MCPEQ(P,A,B)
      INTEGER A,B,C,P,S,T,X,Y
      INTEGER CPDIF,FIRST,TAIL
1     X=A
      Y=B
2     CALL ADV(S,X)
      CALL ADV(T,Y)
3     C=CPDIF(P,FIRST(S),FIRST(T))
      IF (C.NE.0) GO TO 4
      S=TAIL(S)
      T=TAIL(T)
      IF (S.NE.0) GO TO 3
      IF (X.NE.0) GO TO 2
      MCPEQ=1
      RETURN
4     CALL CPERAS(C)
      MCPEQ=0
      RETURN
      END
```

```
      SUBROUTINE MCPERS(A)
      INTEGER A,B,C,K,T
      INTEGER COUNT,FIRST,TYPE
1     T=TYPE(FIRST(A))
2     IF (A.EQ.0) RETURN
      K=COUNT(A)-1
      IF (K.EQ.0) GO TO 21
      CALL SCOUNT(K,A)
      RETURN
21    CALL DECAP(B,A)
3     K=COUNT(B)-1
      IF (K.EQ.0) GO TO 31
      CALL SCOUNT(K,B)
      GO TO 2
31    CALL DECAP(C,B)
      IF (T.EQ.1) CALL CPERAS(C)
      IF (B.NE.0) GO TO 3
      GO TO 2
      END



      INTEGER FUNCTION MCPINT(P,A,B,C,D,S)
      INTEGER A,B,C,D,E,F,H,P,S,T,U,V,W,X,Y
      INTEGER CPINT,INV,PFL
1     X=A
      Y=C
      H=0
2     CALL ADV(W,X)
      CALL ADV(T,Y)
      E=0
3     CALL ADV(U,W)
      CALL ADV(V,T)
      F=CPINT(P,U,B,V,D,S)
      E=PFL(F,E)
      IF (W.NE.0) GO TO 3
4     H=PFL(INV(E),H)
      IF (X.NE.0) GO TO 2
5     MCPINT=INV(H)
      RETURN
      END



      INTEGER FUNCTION MCPMDB(A,B)
      INTEGER A,B,E,R,S,T,U,V,W,X,Y,Z
      INTEGER CPDEG
1     X=A
      Y=B
      Z=0
2     CALL ADV(R,X)
      W=Y
3     S=R
      CALL ADV(T,W)
4     CALL ADV(U,S)
```

```
      CALL ADV(V,T)
      IF (U.EQ.0 .OR. V.EQ.0) GO TO 5
      E=CPDEG(U)+CPDEG(V)
      IF (E.GT.Z) Z=E
5     IF (S.NE.0) GO TO 4
      IF (W.NE.0) GO TO 3
      IF (X.NE.0) GO TO 2
      MCPMDB=Z
      RETURN
      END


      INTEGER FUNCTION MCPMPY(P,AA,BB)
      INTEGER A,AA,ASTAR,B,BB,BSTAR,C,CP,CSTAR,D,E,I,J,K,M,P,Q,RET,S,W
      INTEGER CDIF,CPPROD,FIRST,LENGTH,MCMPY,MCPEVL,MCPINT,MCPMDB
      INTEGER MCPNV,MZERO,PFA,PFL,TYPE
      A=AA
      B=BB
      RET=1
      GO TO 10
91    MCPMPY=C
      RETURN
10    M=LENGTH(A)
      Q=LENGTH(B)
20    IF (TYPE(FIRST(A)).NE.0) GO TO 30
      C=MCMPY(P,A,B)
      GO TO 90
30    C=0
      DO 32 I=1,M
      D=0
      DO 31 J=1,Q
31    D=PFL(0,D)
32    C=PFL(D,C)
      IF (MZERO(A).EQ.1 .OR. MZERO(B).EQ.1) GO TO 90
40    S=MCPNV(A)
      K=MCPMDB(A,B)
      I=0
      D=PFA(0,PFA(1,0))
50    IF (I.LT.P) GO TO 52
      PRINT 51
51    FORMAT(53H ELEMENTS OF GF(P) EXHAUSTED. ALGORITHM MCPMPY FAILS. )
      STOP
52    ASTAR=MCPEVL(P,A,I,S)
      BSTAR=MCPEVL(P,B,I,S)
60    CALL STACK3(A,B,C)
      CALL STACK3(D,I,K)
      CALL STACK2(S,RET)
      A=ASTAR
      B=BSTAR
      RET=2
      GO TO 10
61    CSTAR=C
      ASTAR=A
      BSTAR=B
```

```
      CALL UNSTK2(S,RET)
      CALL UNSTK3(D,I,K)
      CALL UNSTK3(A,B,C)
      CALL MCPERS(ASTAR)
      CALL MCPERS(BSTAR)
70    CP=MCPINT(P,C,I,CSTAR,D,S)
      CALL MCPERS(C)
      CALL MCPERS(CSTAR)
      C=CP
      IF (I.LT.K) GO TO 80
      CALL CPERAS(D)
      GO TO 90
80    E=PFA(1,PFA(1,PFA(CDIF(P,0,I),0)))
      W=CPPROD(P,D,E)
      CALL CPERAS(D)
      CALL CPERAS(E)
      D=W
      I=I+1
      GO TO 50
90    GO TO (91,61),RET
      END



      INTEGER FUNCTION MCPNV(A)
      INTEGER A,B,C,D
      INTEGER CPNV,FIRST,TYPE
1     B=A
      IF (TYPE(FIRST(B)).EQ.1) GO TO 2
      MCPNV=0
      RETURN
2     CALL ADV(C,B)
3     CALL ADV(D,C)
      IF (D.NE.0) GO TO 4
      IF (C.NE.0) GO TO 3
      GO TO 2
4     MCPNV=CPNV(D)
      RETURN
      END



      INTEGER FUNCTION MDIF(A,B)
      INTEGER A,B,C,D,E,F,X,Y
      INTEGER FIRST,INV,PDIF,PFL,TAIL
1     X=A
      Y=B
      C=0
2     CALL ADV(E,X)
      CALL ADV(F,Y)
      D=0
3     D=PFL(PDIF(FIRST(E),FIRST(F)),D)
      E=TAIL(E)
      F=TAIL(F)
      IF (E.NE.0) GO TO 3
```

```
4       C=PFL(INV(D),C)
        IF (X.NE.0) GO TO 2
5       MDIF=INV(C)
        RETURN
        END


        INTEGER FUNCTION MEQ(A,B)
        INTEGER A,B,C,S,T,X,Y
        INTEGER FIRST,PDIF,TAIL
1       X=A
        Y=B
2       CALL ADV(S,X)
        CALL ADV(T,Y)
3       C=PDIF(FIRST(S),FIRST(T))
        IF (C.NE.0) GO TO 4
        S=TAIL(S)
        T=TAIL(T)
        IF (S.NE.0) GO TO 3
        IF (X.NE.0) GO TO 2
        MEQ=1
        RETURN
4       CALL PERASE(C)
        MEQ=0
        RETURN
        END


        SUBROUTINE MERASE(A)
        INTEGER A,B,C,K
        INTEGER COUNT
1       IF (A.EQ.0) RETURN
        K=COUNT(A)-1
        IF (K.EQ.0) GO TO 11
        CALL SCOUNT(K,A)
        RETURN
11      CALL DECAP(B,A)
2       K=COUNT(B)-1
        IF (K.EQ.0) GO TO 21
        CALL SCOUNT(K,B)
        GO TO 1
21      CALL DECAP(C,B)
        CALL PERASE(C)
        IF (B.NE.0) GO TO 2
        GO TO 1
        END
```

```
      INTEGER FUNCTION MGARN(Q,B,P,A,L)
      INTEGER A,B,C,D,E,L,P,Q,S,T,U,V,X,Y
      INTEGER CPGARN,INV,PFL
1     X=B
      Y=A
      C=0
2     CALL ADV(S,X)
      CALL ADV(T,Y)
      D=0
3     CALL ADV(U,S)
      CALL ADV(V,T)
      E=CPGARN(Q,U,P,V,L)
      D=PFL(E,D)
      IF (S.NE.0) GO TO 3
4     C=PFL(INV(D),C)
      IF (X.NE.0) GO TO 2
5     MGARN=INV(C)
      RETURN
      END


      INTEGER FUNCTION MINV(A)
      INTEGER A,C,I,J,K,L,M,P,R,T,U,V,W,X
      INTEGER BORROW,INV,LENGTH,MVLIST,PFA,PFL,PLES,TAIL
10    T=A
      C=0
      J=0
      P=PFA(1,0)
      M=LENGTH(T)
      L=MVLIST(T)
20    IF (L.EQ.0) GO TO 30
      CALL DECAP(V,L)
      P=PFL(V,PFL(P,PFA(0,0)))
      GO TO 20
30    CALL ADV(U,T)
      R=0
      J=J+1
40    CALL ADV(W,U)
      R=PFL(BORROW(W),R)
      IF (U.NE.0) GO TO 40
50    K=J-1
      IF (K.EQ.0) GO TO 52
      DO 51 I=1,K
51    R=PFL(0,R)
52    R=PFL(BORROW(P),R)
      K=M-J
      IF (K.EQ.0) GO TO 54
      DO 53 I=1,K
53    R=PFL(0,R)
54    C=PFL(INV(R),C)
      IF (T.NE.0) GO TO 30
      C=INV(C)
60    X=PLES(M,C)
      IF (X.EQ.0) GO TO 70
```

```
      T=TAIL(X)
      U=TAIL(T)
      CALL SSUCC(0,T)
      CALL ERASE(U)
70    MINV=X
      CALL PERASE(P)
      CALL MERASE(C)
      RETURN
      END


      INTEGER FUNCTION MMCB(A,B)
      INTEGER A,B,H,I,J,K,L,N,R,S,U,V,X,Y
      INTEGER FIRST,ICOMP,IPROD,ISUM,LENGTH,PFA,PFL,PNORMF,TAIL
10    X=A
      Y=B
      K=0
      N=LENGTH(FIRST(X))
      I=0
      DO 11 L=1,N
11    I=PFL(0,I)
      J=0
      DO 12 L=1,N
12    J=PFL(0,J)
20    CALL ADV(R,X)
      S=I
30    CALL ADV(U,R)
      U=PNORMF(U)
      V=FIRST(S)
      H=ICOMP(U,V)
      IF (H.NE.1) GO TO 31
      CALL ERLA(V)
      CALL ALTER(U,S)
      GO TO 32
31    CALL ERLA(U)
32    S=TAIL(S)
      IF (S.NE.0) GO TO 30
      IF (X.NE.0) GO TO 20
40    CALL ADV(R,Y)
      S=J
50    CALL ADV(U,R)
      U=PNORMF(U)
      V=FIRST(S)
      H=ICOMP(U,V)
      IF (H.NE.1) GO TO 51
      CALL ERLA(V)
      CALL ALTER(U,S)
      GO TO 52
51    CALL ERLA(U)
52    S=TAIL(S)
      IF (S.NE.0) GO TO 50
      IF (Y.NE.0) GO TO 40
60    CALL DECAP(U,I)
      CALL DECAP(V,J)
```

```
      R=IPROD(U,V)
      CALL ERLA(U)
      CALL ERLA(V)
      S=ISUM(K,R)
      CALL ERLA(K)
      CALL ERLA(R)
      K=S
      IF (I.NE.0) GO TO 60
      U=PFA(2,0)
      V=IPROD(U,K)
      CALL ERLA(U)
      CALL ERLA(K)
      MMCB=V
      RETURN
      END



      INTEGER FUNCTION MMOD(P,A,S)
      INTEGER A,B,C,P,R,S,T,X
      INTEGER CPMOD,INV,PFA,PFL
1     X=A
      B=0
2     CALL ADV(T,X)
      C=0
3     CALL ADV(R,T)
      R=CPMOD(P,R)
      IF (S.EQ.0) C=PFA(R,C)
      IF (S.NE.0) C=PFL(R,C)
      IF (T.NE.0) GO TO 3
4     B=PFL(INV(C),B)
      IF (X.NE.0) GO TO 2
5     MMOD=INV(B)
      RETURN
      END



      INTEGER FUNCTION MMPY(AA,BB)
      INTEGER A,AA,ASTAR,BB,BP,BSTAR,C,CP,CSTAR,D,I,J,M,P,PRIME
      INTEGER Q,S,T,U,V,Z
      INTEGER ICOMP,IPROD,LENGTH,MCPMPY,MOARN,MMCB,MMOD,MTRAN,MVLIST
      INTEGER MZERO,PFA,PFL
      COMMON /TR4/PRIME
10    A=AA
      BP=MTRAN(BB)
      Z=PRIME
      M=LENGTH(A)
      Q=LENGTH(BP)
20    C=0
      DO 22 I=1,M
      D=0
      DO 21 J=1,Q
21    D=PFL(0,D)
22    C=PFL(D,C)
```

```
30    V=MVLIST(A)
      IF (MZERO(A).EQ.1 .OR. MZERO(BP).EQ.1) GO TO 80
      S=LENGTH(V)
      I=PFA(1,0)
      J=MMCB(A,BP)
40    IF (Z.NE.0) GO TO 42
      PRINT 41
41    FORMAT(48H LIST OF PRIMES EXHAUSTED. ALGORITHM MMPY FAILS. )
      STOP
42    CALL ADV(P,Z)
      ASTAR=MMOD(P,A,S)
      BSTAR=MMOD(P,BP,S)
50    CSTAR=MCPMPY(P,ASTAR,BSTAR)
      CALL MCPERS(ASTAR)
      CALL MCPERS(BSTAR)
60    CP=MGARN(I,C,P,CSTAR,V)
      CALL MERASE(C)
      CALL MCPERS(CSTAR)
      C=CP
70    T=PFA(P,0)
      U=IPROD(I,T)
      CALL ERLA(I)
      CALL ERLA(T)
      I=U
      IF (ICOMP(I,J).NE.1) GO TO 40
      CALL ERASE(V)
      CALL ERLA(I)
      CALL ERLA(J)
80    CALL MERASE(BP)
      MMPY=C
      RETURN
      END


      INTEGER FUNCTION MPROD(A,B)
      INTEGER A,B,C,D,E,F,G,R,S,T,V,W,X,Y
      INTEGER CINV,FIRST,INV,MTRAN,PFL,PPROD,PSUM,TAIL
1     X=CINV(A)
      Y=INV(MTRAN(B))
      C=0
      V=X
2     CALL ADV(R,V)
      W=Y
      D=0
3     S=R
      CALL ADV(T,W)
      E=0
4     F=PPROD(FIRST(S),FIRST(T))
      G=PSUM(E,F)
      CALL PERASE(E)
      CALL PERASE(F)
      E=G
      S=TAIL(S)
      T=TAIL(T)
```

```
      IF (S.NE.0) GO TO 4
5     D=PFL(E,D)
      IF (W.NE.0) GO TO 3
6     C=PFL(D,C)
      IF (V.NE.0) GO TO 2
7     CALL ERASE(X)
      CALL ERASE(Y)
      MPROD=C
      RETURN
      END



      INTEGER FUNCTION MSUM(A,B)
      INTEGER A,B,C,D,E,F,X,Y
      INTEGER FIRST,INV,PFL,PSUM,TAIL
1     X=A
      Y=B
      C=0
2     CALL ADV(E,X)
      CALL ADV(F,Y)
      D=0
3     D=PFL(PSUM(FIRST(E),FIRST(F)),D)
      E=TAIL(E)
      F=TAIL(F)
      IF (E.NE.0) GO TO 3
4     C=PFL(INV(D),C)
      IF (X.NE.0) GO TO 2
5     MSUM=INV(C)
      RETURN
      END



      INTEGER FUNCTION MTRAN(A)
      INTEGER A,B,C,D,F,G,I,N,S,T
      INTEGER BORROW,CINV,FIRST,LENGTH,PFA,PFL,TAIL,TYPE
1     G=CINV(A)
      F=G
      C=FIRST(F)
      T=TYPE(C)
      N=LENGTH(C)
      B=0
      DO 15 I=1,N
15    B=PFL(0,B)
2     CALL ADV(C,F)
      S=B
3     CALL ADV(D,C)
      IF (T.EQ.0) CALL ALTER(PFA(D,FIRST(S)),S)
      IF (T.NE.0) CALL ALTER(PFL(BORROW(D),FIRST(S)),S)
```

```
      S=TAIL(S)
      IF (S.NE.0) GO TO 3
      IF (F.NE.0) GO TO 2
4     CALL ERASE(G)
      MTRAN=B
      RETURN
      END


      INTEGER FUNCTION MVLIST(A)
      INTEGER A,B,C,X
      INTEGER PVLIST
1     X=A
2     CALL ADV(B,X)
3     CALL ADV(C,B)
      IF (C.NE.0) GO TO 4
      IF (B.NE.0) GO TO 3
      GO TO 2
4     MVLIST=PVLIST(C)
      RETURN
      END


      INTEGER FUNCTION MZCON(A)
      INTEGER A,B,S,U,X
      INTEGER INV,PFL,TAIL
1     B=0
      S=A
2     CALL ADV(U,S)
      X=0
3     X=PFL(U,X)
      U=TAIL(U)
      IF (U.NE.0) GO TO 3
      B=PFL(X,B)
      IF (S.NE.0) GO TO 2
4     MZCON=INV(B)
      RETURN
      END


      INTEGER FUNCTION MZERO(A)
      INTEGER A,X,Y
      INTEGER FIRST,TAIL
1     X=A
      MZERO=0
2     CALL ADV(Y,X)
3     IF (FIRST(Y).NE.0) RETURN
      Y=TAIL(Y)
      IF (Y.NE.0) GO TO 3
      IF (X.NE.0) GO TO 2
      MZERO=1
      RETURN
      END
```

```
      INTEGER FUNCTION NULCON(N,J,P,W)
      INTEGER E,H,I,J,K,L,N,P,R,S,T,U,W,WP,Z
      INTEGER BORROW,CINV,COUNT,INV,LENGTH,PFL
10    L=J
      WP=CINV(W)
      R=LENGTH(J)
      T=R-LENGTH(W)
      IF (T.EQ.0) GO TO 12
      DO 11 S=1,T
11    WP=PFL(0,WP)
12    WP=INV(WP)
      E=N-R
      Z=0
      CALL ADV(H,L)
      I=0
      K=0
20    I=I+1
      IF (I.GT.N) GO TO 50
      IF (I.EQ.H) GO TO 40
30    K=K+1
      U=0
      T=E-K
      IF (T.EQ.0) GO TO 32
      DO 31 S=1,T
31    U=PFL(0,U)
32    U=PFL(BORROW(P),U)
      T=K-1
      IF (T.EQ.0) GO TO 34
      DO 33 S=1,T
33    U=PFL(0,U)
34    Z=PFL(U,Z)
      GO TO 20
40    CALL DECAP(U,WP)
      IF (U.NE.0) CALL SCOUNT(COUNT(U)-1,U)
      U=INV(CINV(U))
      T=E-LENGTH(U)
      IF (T.EQ.0) GO TO 42
      DO 41 S=1,T
41    U=PFL(0,U)
42    Z=PFL(U,Z)
      IF (L.NE.0) CALL ADV(H,L)
      GO TO 20
50    NULCON=INV(Z)
      RETURN
      END


      INTEGER FUNCTION NULSP(AA)
      INTEGER A,AA,C,D,G,N,R,T,U,V,Y,Z
      INTEGER BORROW,FIRST,INV,LENGTH,PFL,PLES
      A=AA
10    T=A
      C=0
20    CALL ADV(U,T)
```

```
      R=0
30    CALL ADV(V,U)
      R=PFL(BORROW(V),R)
      IF (U.NE.0) GO TO 30
      R=PFL(0,R)
      C=PFL(INV(R),C)
      IF (T.NE.0) GO TO 20
      C=INV(C)
40    N=LENGTH(FIRST(A))
      G=PLES(N,C)
      CALL DECAP(D,G)
      CALL DECAP(Y,G)
      CALL DECAP(Z,G)
      CALL PERASE(D)
      CALL MERASE(Y)
      CALL MERASE(C)
      NULSP=Z
      RETURN
      END



      INTEGER FUNCTION PDET(AA)
      INTEGER A,AA,ASTAR,D,DP,DSTAR,H,I,J,L,P,PRIME,S,T,U,V
      INTEGER CBDET,CPDET,CPGARN,ICOMP,IPROD,LENGTH,MMOD,MVLIST
      INTEGER MZERO,PFA
      COMMON /TR4/PRIME
      A=AA
10    D=0
      V=MVLIST(A)
      S=LENGTH(V)
      J=CBDET(A)
      I=PFA(1,0)
      L=PRIME
      H=0
20    IF (L.NE.0) GO TO 22
      PRINT 21
21    FORMAT(48H LIST OF PRIMES EXHAUSTED. ALGORITHM PDET FAILS.)
      STOP
22    CALL ADV(P,L)
      ASTAR=MMOD(P,A,S)
      IF (MZERO(ASTAR).EQ.0) GO TO 30
      CALL MCPERS(ASTAR)
      DSTAR=0
      GO TO 40
30    DSTAR=CPDET(P,ASTAR)
40    IF (DSTAR.NE.0) H=1
      IF (H.EQ.0) GO TO 50
      DP=CPGARN(I,D,P,DSTAR,V)
      CALL PERASE(D)
      IF (V.NE.0) CALL CPERAS(DSTAR)
      D=DP
50    T=PFA(P,0)
      U=IPROD(I,T)
      CALL ERLA(I)
```

```
      CALL EPLA(T)
      I=U
      IF (ICOMP(I,J).NE.1) GO TO 20
60    PDET=D
      CALL ERLA(I)
      CALL ERLA(J)
      CALL ERASE(V)
      RETURN
      END



      INTEGER FUNCTION PLES(NN,CC)
      INTEGER C,CC,CSTAR,D,DP,DSTAR,E,EP,G,H,I,ISTAR,J,JSTAR,K,L,N,NN
      INTEGER NP,P,PRIME,R,RSTAR,S,T,U,V,VP,VSTAR,W,WSTAR,X,Y,Z,ZP
      INTEGER CPGARN,CPRRE,FIRST,INV,IPROD,LENGTH,MEQ,MGARN,MMOD
      INTEGER MMPY,MVLIST,MZCON,MZERO,NULCON,PDIF,PFA,PFL,PNEG
      INTEGER TAIL,VCOMP
      COMMON /TR4/PRIME
      N=NN
      C=CC
10    X=MVLIST(C)
      S=LENGTH(X)
      L=PRIME
      R=0
20    IF (L.NE.0) GO TO 22
      PRINT 21
21    FORMAT(49H LIST OF PRIMES EXHAUSTED.  ALGORITHM PLES FAILS.)
      STOP
22    CALL ADV(P,L)
      CSTAR=MMOD(P,C,S)
      IF (MZERO(CSTAR).EQ.0) GO TO 30
      CALL MCPERS(CSTAR)
      GO TO 20
30    WSTAR=CPRRE(P,CSTAR)
      CALL DECAP(JSTAR,WSTAR)
      CALL DECAP(ISTAR,WSTAR)
      CALL DECAP(DSTAR,WSTAR)
      CALL DECAP(VSTAR,WSTAR)
40    RSTAR=LENGTH(JSTAR)
      IF (RSTAR-R) 50,41,60
41    IF (VCOMP(JSTAR,J)) 60,42,50
42    IF (VCOMP(ISTAR,I)) 60,43,50
43    CALL ERLA(JSTAR)
      CALL ERLA(ISTAR)
      GO TO 70
50    CALL ERLA(JSTAR)
      CALL ERLA(ISTAR)
      IF (X.NE.0) CALL CPERAS(DSTAR)
      IF (VSTAR.NE.0) CALL MCPERS(VSTAR)
      GO TO 20
60    IF (R.EQ.0) GO TO 61
      CALL ERLA(J)
      CALL ERLA(I)
      CALL PERASE(D)
```

```
      IF (V.NE.0) CALL MERASE(V)
      CALL ERLA(E)
61    R=RSTAR
      J=JSTAR
      I=ISTAR
      D=0
      V=0
      IF (VSTAR.NE.0) V=MZCON(VSTAR)
      E=PFA(1,0)
70    DP=CPGARN(E,D,P,DSTAR,X)
      IF (X.NE.0) CALL CPERAS(DSTAR)
      IF (V.EQ.0) GO TO 71
      VP=MGARN(E,V,P,VSTAR,X)
      CALL MCPERS(VSTAR)
71    T=PFA(P,0)
      EP=IPROD(T,E)
      CALL ERLA(T)
      CALL ERLA(E)
      E=EP
80    T=PDIF(D,DP)
      CALL PERASE(D)
      D=DP
      U=1
      IF (V.EQ.0) GO TO 81
      U=MEQ(V,VP)
      CALL MERASE(V)
      V=VP
81    IF (U.EQ.1 .AND. T.EQ.0) GO TO 90
      CALL PERASE(T)
      GO TO 20
90    IF (R.LE.N) GO TO 91
      G=0
      CALL PERASE(D)
      GO TO 120
91    NP=LENGTH(FIRST(C))
      DP=PNEG(D)
      ZP=NULCON(NP,J,DP,V)
      T=MMPY(C,ZP)
      U=MZERO(T)
      CALL PERASE(DP)
      CALL MERASE(T)
      IF (U.EQ.1) GO TO 100
      CALL MERASE(ZP)
      GO TO 20
100   J=INV(J)
      IF (FIRST(J).LE.N) GO TO 101
      CALL MERASE(ZP)
      G=0
      CALL PERASE(D)
      GO TO 120
101   Y=0
      Z=0
      H=N-R-1
      U=N
110   U=U-1
```

```
      CALL DECAP(T,ZP)
      IF (H.LT.0) GO TO 113
      Z=PFL(T,Z)
      IF (H.EQ.0) GO TO 112
      DO 111 K=1,H
111   T=TAIL(T)
112   W=T
      T=TAIL(T)
      CALL SSUCC(0,W)
113   Y=PFL(T,Y)
      IF(U.GT.0) GO TO 110
      CALL MEPASE(ZP)
      G=PFL(D,PFL(INV(Y),PFL(INV(Z),0)))
120   CALL ERLA(J)
      CALL ERLA(I)
      IF (V.NE.0) CALL MERASE(V)
      CALL ERLA(E)
      CALL ERASE(X)
      PLES=G
      RETURN
      END



      INTEGER FUNCTION VCOMP(H,K)
      INTEGER H,K,S,T,U,V,W
1     S=H
      T=K
2     IF (S.NE.0) GO TO 3
      IF (T.NE.0) GO TO 5
      VCOMP=0
      RETURN
3     IF (T.NE.0) GO TO 4
      GO TO 6
4     CALL ADV(U,S)
      CALL ADV(V,T)
      W=U-V
      IF (W.EQ.0) GO TO 2
      IF (W.GT.0) GO TO 6
5     VCOMP=-1
      RETURN
6     VCOMP=1
      RETURN
      END
```

REFERENCES

[1] Fortran vs. Basic Fortran - A Programming Language for Information Processing on Automatic Data Processing Systems, CACM, Vol. 7, No. 10 (Oct. 1964), 591-625.

[2] Collins, G. E., The SAC-1 List Processing System, University of Wisconsin Computer Sciences Department Technical Report No. 129 (34 pages), July 1971.

[3] Collins, G. E., and J. R. Pinkert, The Revised SAC-1 Integer Arithmetic System, University of Wisconsin Computing Center Technical Report No. 9 (50 pages), Nov. 1968.

[4] Collins, G. E., The SAC-1 Polynomial System, University of Wisconsin Computer Sciences Department Technical Report No. 115 (66 pages), March 1971.

[5] Collins, G. E., The SAC-1 Rational Function System, University of Wisconsin Computing Center Technical Report No. 8 (32 pages), September 1971.

[6] Collins, G. E., L. E. Heindel, E. Horowitz, M. T. McClellan, and D. R. Musser, The SAC-1 Modular Arithmetic System, University of Wisconsin Computing Center Technical Report No. 10 (50 pages), June 1969.

[7] Collins, G. E., and E. Horowitz, The SAC-1 Partial Fraction Decomposition and Rational Function Integration System, University of Wisconsin Computer Sciences Department Technical Report No. 80 (47 pages), Feb. 1970.

[8] Collins, G. E., and L. E. Heindel, The SAC-1 Polynomial Real Zero System, University of Wisconsin Computing Center Technical Report No. 19 (72 pages), August 1970.

[9] Collins, G. E., The SAC-1 Polynomial GCD and Resultant System, University of Wisconsin Computer Sciences Department Technical Report No. 145 (94 pages), February 1972.

[10] Collins, G. E. and D. R. Musser, The SAC-1 Polynomial Factorization System, University of Wisconsin Computer Sciences Department Technical Report, in preparation.

[11] McClellan, M. T., The Exact Solution of Systems of Linear Equations with Polynomial Coefficients, University of Wisconsin Computer Sciences Department Technical Report No. 136 (Ph.D. thesis), (258 pages), September 1971.

[12] Musser, D. R., Algorithms for Polynomial Factorization, University of Wisconsin Computer Sciences Department Technical Report No. 134 (Ph.D. thesis), (174 pages), September 1971.

INDEX TO ALGORITHMS

BIBLIOGRAPHIC DATA SHEET	1. Report No. WIS-CS-154-72	2.	3. Recipient's Accession No.
4. Title and Subtitle The SAC-1 Polynomial Linear Algebra System			5. Report Date April 1972 6.
7. Author(s) G. E. Collins and M. T. McClellan			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Computer Sciences Department 1210 West Dayton Street, University of Wisconsin Madison, Wisconsin 53706			10. Project/Task/Work Unit No. 11. Contract/Grant No. GJ-239, GJ-30125X
12. Sponsoring Organization Name and Address National Science Foundation Washington, D. C.			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

This system is the tenth in a series of subsystems comprising the SAC-1 System for Symbolic and Algebraic Calculation. The present subsystem consists of programs implementing modular algorithms for linear equations solution, matrix inversion, determinant calculation, null space basis generation, and matrix multiplication, all for matrices with integer or polynomial entries. For each program in the system is given a functional specification, an algorithm description, an analytical computing time, and a Fortran program listing. Empirically observed computing times for some of the key programs are presented. Also, a test program is supplied as an aid in implementing the system and to illustrate its use.

17. Key Words and Document Analysis. 17a. Descriptors

Polynomials, Linear Algebra, Linear Equations, Determinants, Matrices, Matrix Inversion, Null Space, Vector Spaces, Basis Vectors, SAC-1, Fortran, Algorithms, Modular Arithmetic, Computing Time Analysis, Computational Complexity, Computational Algebra, Symbol Manipulation.

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Available to public.	UNCLASSIFIED	110
	20. Security Class (This Page) UNCLASSIFIED	22. Price