The ALPAK System for Nonnumerical Algebra on a Digital Computer — III: Systems of Linear Equations and a Class of Side Relations

By J. P. HYDE

(Manuscript received March 6, 1964)

This is the third and last in a series of papers describing the ALPAK system for nonnumerical algebra on a digital computer. The first paper[1] *is concerned with polynomials in several variables and truncated power series with polynomial coefficients. The second paper*[2] *is concerned with rational functions in several variables and truncated power series with rational-function coefficients. The present paper discusses systems of linear equations with rational-function coefficients and a certain class of side relations.*

The ALPAK system has been programmed within the BE-SYS-4 monitor system on the IBM 7090 computer, but the language and concepts are machine independent. Several practical applications are described in Ref. 1.

This paper is divided into six sections. The first two assume that the reader has no knowledge of computers or computer programming and the last four assume that the reader is familiar with basic computer programming and Refs. 1 and 2. Section I is a general description of ALPAK and this paper; Section II discusses the different forms in which a linear system can occur, including canonical form; Section III describes the ALPAK linear system operations for converting these forms; Section IV discusses side relations; Section V describes list naming operations; and Section VI discusses possible future developments and improvements.

I. INTRODUCTION

This is the third and last in a series of papers describing the ALPAK system, a programming system for performing routine manipulations of algebraic expressions on a digital computer. The system can perform the operations of addition, subtraction, multiplication, division, sub-

stitution, and differentiation. The first paper[1] is concerned with polynomials in several variables and truncated power series with polynomial coefficients. The second paper[2] is concerned with rational functions in several variables and truncated power series with rational-function coefficients. The present paper describes the ALPAK facilities for manipulating and solving by Gaussian elimination systems of equations linear in certain variables with coefficients which are rational functions of other variables. The facilities for handling a certain class of side relations are also described.

The ALPAK system has been programmed within the BE-SYS-4 monitor system on the IBM 7090 computer, but the language and concepts are machine independent. Several practical applications are described in Ref. 1.

This paper is divided into six sections, of which the first two do not presuppose any knowledge of computers or computer programming and the last four assume that the reader is familiar with the basic concepts of computer programming and Refs. 1 and 2. Section I is a general description of ALPAK and deals with basic concepts. Section II describes the different forms in which a linear system can occur, including especially the canonical form of a linear system. Section III discusses the ALPAK linear system operations for converting these forms.

Section IV describes the way in which ALPAK has been programmed to simplify a rational function, using a certain class of side relations. The most important relations in the allowed class are of the form $X^2 = C$ (C a rational function independent of X). This includes in particular $i^2 = -1$ and $s^2 = 1 - c^2$ where s and c can stand for $\sin \alpha$ and $\cos \alpha$, respectively. The simplification is done by a special rearrangement of the ALPAK format statement and has certain limitations.

Section V discusses list naming operations, a convenient set of auxiliary operations for handling arguments of ALPAK subroutines which are lists (one-dimensional arrays). Finally, Section VI discusses possible future developments and improvements.

1.1 *An Example of the ALPAK Language*

The simplicity of handling linear systems by ALPAK is illustrated in the process of solving the following system of two linear equations, *EQ1* and *EQ2*, in two unknowns, *X1* and *X2*, with polynomials in *a* as coefficients.

$$\text{EQ1: } 3aX1 + 2aX2 - 1 = 0$$

$$\text{EQ2: } 2aX1 + 5a^2X2 - 3 = 0$$

We first extract a coefficient matrix, SYS, for the equations with -1 and -3 moved to the right side.

$$\text{SYS:} \left\| \begin{matrix} 3a & 2a & 1 \\ 2a & 5a^2 & 3 \end{matrix} \right\|.$$

The matrix is then put into canonical form using Gaussian elimination.[3,4]

$$\text{SYS:} \left\| \begin{matrix} 1 & 0 & \dfrac{-6 + 5a}{-4a + 15a^2} \\ 0 & 1 & \dfrac{7}{-4a + 15a^2} \end{matrix} \right\|.$$

The fact that the original coefficient matrix and the canonical form matrix both have the name *SYS* does not imply that they are equal but rather that the latter replaces the former *physically* in the computer. The expressions for the unknowns are then extracted from the coefficient matrix.

$$\text{X1:} \quad \frac{-6 + 5a}{-4a + 15a^2}$$

$$\text{X2:} \quad \frac{7}{-4a + 15a^2}.$$

The following program illustrates how these operations are performed by ALPAK.

```
SYS     SYSRES     2,2
```

Reserve space in the computer for the physical representation of the set of system coefficients which will be obtained from two linear equations in two unknowns and name the set *SYS*.

```
        SYSFRM     SYS,(EQ1,EQ2),=2
```

Extract the 2×3 coefficient matrix from the equations *EQ1* and *EQ2* and place it in *SYS*. The "=2" says that there are two unknowns and the third column of the matrix is used for the terms of the equations which are independent of the unknowns.

```
        SYSPRT     SYS
```

Print the system coefficients.

SYSCFM SYS

Put the system into canonical form (described in next section) using Gaussian elimination. If the system were triangular, the row selection strategy would cause this to be done in the obvious way. The canonical system retains the same name as the original system.

SYSSLV (X1,X2),SYS

Fill *X1* and *X2* with the solutions for the unknowns in *SYS*. The operation SYSSLV assumes that *SYS* is in canonical form.

RFNPRT X1
RFNPRT X2

Print *X1* and *X2*.

X1
X2

These are names of single cells in memory which will be filled in with "pointers"* to the physical representations of the solutions in the computer.

The usefulness of the linear system operations was demonstrated in a problem from queuing theory, proposed by L. Takacs,† in which a truncated power series of 813 terms was involved in forming a system of nine linear equations in nine unknowns. One of these unknowns was the third moment of a probability distribution. Its numerator had 200 terms in five variables with maximum degrees 1, 1, 3, 7, and 9 and its denominator had 39 terms in two variables with maximum degrees 7 and 10.

II. LINEAR SYSTEMS

In this section are discussed the different forms of linear systems as they are dealt with in the ALPAK context. It is important in writing ALPAK programs to remember what these forms are. The next section discusses the ALPAK subroutines for changing one form to another.

* See Ref. 2, p. 795.
† See Ref. 1, pp. 2090–2092.

2.1 *System of Equations*

A linear system of m equations in n unknowns, x_j, is a set of m rational functions, (1), of v variables ($v \geqq n$), each of which is linear in the x_j and is implicitly equal to zero.

$$\sum_{j=1}^{n} \lambda_{ij} x_j - c_i = 0 \qquad (1 \leqq i \leqq m). \tag{1}$$

Thus for each i in (1) the λ_{ij} are the coefficients of the x_j and together with c_i may be thought of as $n + 1$ rational functions with a common denominator.

2.2 *System Coefficients*

Consider (1) written in the form:

$$\sum_{j=1}^{n} \lambda_{ij} x_j = c_i \qquad (1 \leqq i \leqq m). \tag{2}$$

The λ_{ij} and the c_i of (2) shall be referred to as the *system coefficients* of the linear system (1). In ALPAK they form an array of $m(n + 1)$ rational functions stored row-wise and forwards.

2.3 *System Canonical Form*

Let $x_{a_1}, \cdots, x_{a_r}$ be a subset of the unknowns $x_1, \cdots, x_n$ which we shall call the *dependent set*, and let $x_{a_{r+1}}, \cdots, x_{a_n}$ be the remaining unknowns, which we shall call the *independent set*. The dependent set is said to be *valid* if r is the rank of the system and if the associated columns of system coefficients are linearly independent over the field to which they belong. The system

$$x_{a_i} + \sum_{j=r+1}^{n} \lambda_{ij} x_{a_j} = c_i \qquad (1 \leqq i \leqq r) \tag{3}$$

and its array of system coefficients are both said to be in *canonical form* with respect to such a dependent set.* It can be shown that for any linear system and a given valid dependent set, there exists a unique canonical form which is obtainable from the original system and which is satisfied by the same values of the x_{a_i}. One obtains this canonical form by Gaussian elimination; i.e., operating on the system by suitably chosen row operations and column interchanges.† When there is a choice of row interchange, the row with the most zero coefficients is selected to minimize the work involved. If, in the derived canonical form, $r < m$,

* The dependent set in (3) is clearly valid.

† See Refs. 3 and 4.

the last $m - r$ rows should be of the form $0 = 0$. If they are not, the system is said to be *inconsistent*. If $r < n$, the system is said to be *singular*.

2.4 *System Solution*

The solution of a linear system is a set of r rational functions, (4), of v variables ($v \geqq n - r$), each of which is linear in the x_{a_j} ($r + 1 \leqq j \leqq n$) and is implicitly equal to x_{a_i} .

$$x_{a_i} = c_i - \sum_{j=r+1}^{n} \lambda_{ij} x_{a_j} \qquad (1 \leqq i \leqq r). \tag{4}$$

The solution is easily produced once the system is in canonical form, and if the system is nonsingular the solution is of the form $x_{a_i} = c_i$ ($1 \leqq i \leqq n$).

III. LINEAR SYSTEM OPERATIONS

3.1 *General Remarks*

In this section are discussed the ALPAK subroutines for converting the different linear system forms discussed in Section II. The name of a set of system coefficients must be defined by operations SYSNAM or SYSRES if it is to be used in any other operations. This name is the BSS address of a three-word system heading in which are stored the five system parameters. These parameters are the BSS address of the system coefficients, the number of equations, the number of unknowns, an ALPAK format address, and the number of leading variables in this format of which the equations are independent. They are set at assembly time by operations SYSNAM and SYSRES or at run time by operations SYSSET and SYSMPR.

The $m(n + 1)$ system coefficient pointers are stored row-wise and forwards, and a block of $n + 1$ cells must immediately follow to be used by ALPAK as work space. In the ALPAK format statement of the system equations, the n unknowns must have consecutive variable numbers $k + 1$ through $k + n$ ($k \geqq 0$). If $k > 0$, the system equations must be independent of the first k variables, and thus the system coefficients are independent of the first $k + n$ variables. The system parameter *fmt* is normally this ALPAK format statement and is referenced in any system operations involving the names of the unknowns. If it is not supplied by SYSNAM, SYSRES, SYSFRM, or SYSSET, all such operations must refer to variables by number (VARTYP NUM or VARTYP NUM*).

Those arguments of operations SYSFRM, SYSCFM, and SYSSLV which are lists are specified according to the conventions established

in Section V. System parameters and system names are not indexable, but the addresses where they are stored or to be stored are. As in other ALPAK operations, index registers are preserved with the exception of index register four.

3.2 *Notational Conventions*

The following conventions of notation are used in descriptions of instructions. Upper-case letters are used for operation codes (including macro names) and for any parameters which must appear exactly as shown. Dummy parameters are indicated by lower-case letters. A dummy parameter usually stands for the symbolic address of a cell or block of cells in the program where the argument is stored. Those dummy parameters which are the arguments themselves are in boldface. Finally, optional parameters are enclosed in brackets, and parameters which usually have subarguments are enclosed in parentheses. All integer arguments are decimal. By this notation, then, the instructional description

sys	SYSRES	**m**,**n**,[fmt],[k]

specifies certain properties and restrictions about the arguments of the following call:

COEFF	SYSRES	9,9,,INDEP

Thus, only SYSRES must appear exactly as shown and all other parameters are dummies with the third one omitted, as it is optional. The number of equations is nine, but the number of leading variables of which the equations are independent is in the cell whose symbolic address is INDEP.

3.3 *Linear System Operations*

sys	SYSNAM	bss,**m**,**n**,[fmt],[k]	name	(a)
sys	SYSRES	**m**,**n**,[fmt],[k]	reserve	(b)
	SYSPRT	sys	print	(c)
	SYSFRM	sys,(listr),n,[k]	form	(d)
	SYSCFM	sys,[(listv)],[**inc**],[**ids**]	canonical form	(e)
	SYSSLV	(listr),sys,[(listv)]	solve	(f)
	SYSOBT	[(abss)],[(m)],[(n)],[(afmt)], [(k)],sys	obtain parameters	(g)
	SYSSET	sys,[abss],[m],[n],[afmt],[k]	set parameters	(h)
	SYSMPR	sys,[(**op** oper)], [(**op** oper)] [(**op** oper)], [(**op** oper)], [(**op** oper)]	modify parameters	(i)

sys = name of system (symbolic address of heading)
bss = BSS address of the array of system coefficients
abss = address where *bss* is or is to be stored
m = the number of equations in the system
n = the number of unknowns in the system
fmt = the address of the system's ALPAK format statement
afmt = address where *fmt* is or is to be stored
k = the number of leading variables in this format statement of which the system equations are independent
listr = list of rational functions (see Section V)
listv = list of variables (specified in the manner indicated by the last previous VARTYP declaration — see Section V)
(op oper) = a 7090-94 machine operation and an operand separated by a blank
inc = inconsistency return
ids = invalid dependent set return.

3.4 *Descriptions*

(a) sys SYSNAM bss,**m**,**n**,[fmt],[k]

Declare a block of length $(m + 1)$ $(n + 1)$ starting at *bss* to be a set of linear system coefficients and work space, and name it *sys* by reserving remotely a three-word system heading. This heading is filled in with *bss*, *m*, *n*, *fmt*, and *k*. If *fmt* and/or *k* is omitted, the corresponding fields in the system heading are filled in with zeros.

(b) sys SYSRES **m**,**n**,[fmt],[k]

Reserve remotely a block of length $(m + 1)$ $(n + 1)$ for a set of system coefficients and work space, and name the set *sys* by reserving remotely a three-word system heading as in SYSNAM. *sys* is to be filled in at run time (e.g., by SYSFRM).

(c) SYSPRT sys

Print the set *sys* of system coefficients.

(d) SYSFRM sys,(listr),n,[k]

Replace *sys* by the set of system coefficients formed from the set *listr* of system equations and remove the common factors between the coefficients of any given equation (*listr* is destroyed). The contents of *n* and *k* and the number of rational functions in *listr* together with their format are copied into the heading of *sys*. If *k* is not supplied, it

is assumed to be zero. If SYSFRM is not used to fill in *sys*, the system parameters must be filled in with operations (a), (b), or (h).

(e) SYSCFM sys,[(listv)],[inc],[ids]

Replace the set *sys* of system coefficients by its associated canonical set, using Gaussian elimination. *listv* is a list of unknowns (specified in the manner indicated by the last previous VARTYP declaration) to be included in a valid dependent set. If *listv* is not supplied, the list is assumed to be empty. If *sys* is found to be inconsistent, control will be transferred to *inc* (or to the REMARK subroutine if *inc* is not supplied) and *sys* will have a canonical form with an inconsistency. If the set of unknowns in *listv* cannot be included in a valid dependent set, control will be transferred to *ids* (or to the REMARK subroutine if *ids* is not supplied) and *sys* will have a canonical form with some subset of *listv* in the dependent set. At *inc* or *ids* it is possible to call SYSSLV, SYSPRT, or to go to some other part of the program.

(f) SYSSLV (listr),sys,[(listv)]

Replace *listr* (whose length must not be less than that of *listv*) by the solutions for the list of unknowns *listv* (specified in the manner indicated by the last previous VARTYP declaration). *sys* is assumed to be in canonical form. If *listv* is not supplied, all the unknowns in the dependent set are solved for in the order in which they were at the start of SYSCFM.

(g) SYSOBT [(abss)],[(m)],[(n)],[(afmt)],[(k)],sys

Obtain the system coefficient parameters of the system whose name is *sys*. Each optional argument is a memory location in whose address field the parameter is to be stored. Thus the parameter *bss* is stored in the location *abss* specified by SYSOBT, etc. Each optional argument may actually be several arguments, and if an argument is an integer equal to seven or less, it refers to an index register.

(h) SYSSET sys,[abss],[m],[n],[afmt],[k]

Set the system coefficient parameters of the system whose name is *sys* from the locations specified by the bracketed arguments. Thus the parameter *bss* is set to the contents of the location *abss* specified by SYSSET, etc.

(i) SYSMPR sys,[(**op** oper)],[(**op** oper)]

[(**op** oper)], [(**op** oper)], [(**op** oper)]

Modify the system coefficient parameters of the system whose name is *sys* using the 7090-94 machine operations *op* with operands *oper*. Thus, the parameter *bss* is modified by the first operation and operand, *m* is modified by the second, *n* by the third, *fmt* by the fourth, and *k* by the fifth. Each operation and operand may be different. Typically, the operation is ADD or SUB and the operand is the address of some increment or decrement.

IV. SIDE RELATIONS

4.1 *General Remarks*

The ALPAK programmer may find that expressions involving radicals occur in his problem. A radical can be handled by assigning it a variable name and writing the rational functions using this name. Thus in the polynomial $a + 2a\sqrt{3}$, we let $X = \sqrt{3}$ and the expression becomes $a + 2aX$. The problem is that in the outputs of arithmetic operations involving such rational functions, X can have an exponent greater than one and the fact that $X^2 = 3$, $X^3 = 3X$, $X^4 = 9$, $\cdots$ will be ignored. The implicit equation $X^2 = 3$ is called a *side relation* of degree two on X. A subroutine is provided for simplifying rational functions using side relations of the general form

$$X^{2^j} = C \qquad \begin{matrix} (j \text{ an integer} \geqq 1) \\ (C \text{ a rational function independent of } X). \end{matrix} \tag{5}$$

This category includes especially $i^2 = -1$ and $s^2 = 1 - c^2$ where s and c can stand for $\sin \alpha$ and $\cos \alpha$, respectively.

4.2 *Limitations*

Many limitations exist in the present handling of side relations. In the relation $X^n = C$, n must be a power of two and X a single variable. Dependencies between relations are not observed; i.e., $R^2 = 2$ and $S^2 = 3$ and $T^2 = 6$ will not result in the implicit relation of $T = \pm RS$. Moreover, relations are not handled automatically by the lowest-level subroutines, thus causing exponents to grow unnecessarily until simplification is done at main program level. A more sophisticated version of ALPAK would prevent this by including the relations as part of the format statement. To repair these limitations would require a great deal of extra programming, and it turns out in practice that these limitations do not usually matter. The general problem of dependencies is especially difficult, as it involves algebraic extensions of the field of rational functions of several variables.

4.3 *Implementation*

The simplification of a rational function, *RF*, by a side relation $X^{2^j} = C$ is accomplished with the aid of a specially constructed temporary ALPAK format statement. The temporary format is the same as the original one except that the exponent field of X is split into two parts. The right j bits are assigned the name X and the remaining left-hand bits form a temporary exponent field which is assigned any name and stands effectively for X^{2^j}. The rational function C is then substituted for the temporary variable by the call SIDREL. If several side relations are defined on several variables, then a single temporary format statement can be used to split up these variables. There will then be a list of rational functions to be substituted for the temporary variables by a single call to SIDREL (see Section 3.2).

SIDREL rf,(listv),(listr),tfmt

rf = rational function to be simplified
listv = list of temporary variables (specified in the manner indicated by the last previous VARTYP declaration — see Section V)
listr = list of rational functions to be substituted for these variables and specified in the same order (see Section V)
tfmt = address of temporary format.

The format of *rf* after simplification is the format of the items in *listr*, or if none of these items has a format, the format of *rf* is unaltered.

4.4 *Example*

Suppose it is desired to simplify the function *RF* using the side relation $I^2 = -1$. This is done by the following program.

```
FMT     POLCVF    (X,5,Y,5,I,5,Z,21)
                                  Permanent format.
TFMT    POLCVF    (X,5,Y,5,ISQ,4,I,1,Z,21)
                                  Temporary format with I split up
                                  into ISQ with four bits and I with
                                  one bit.
        VARTYP    NAM
        POLSTC    MON,=-1
        SIDREL    RF,ISQ,MON,TFMT
          ⋮
RF
MON
```

Testing equality of rational functions R and S which have been simplified by a side relation should always be done by subtracting and testing for zero as follows:

```
RFNSUB   TEMP,R,S
SIDREL   TEMP,ISQ,MON,TFMT
  :                The side relation applied to R and
                   S is applied to TEMP.
RFNZET   TEMP
                   Test if TEMP is zero.
```

This procedure will recognize that the expressions $(1 + i)/(1 - i)$ and i are equal.

V. LIST NAMING OPERATIONS

5.1 *General Remarks*

Whenever an ALPAK subroutine argument is a list, the list may be specified either by actually listing the contents; e.g.,

```
SYSSLV     (P,Q,R),SYS,(X,Y,Z)
```

or by name; e.g.,

```
SYSSLV     (LISTP,*),SYS,(LISTV,*)
```

Here the asterisk indicates that the list has been specified by name. Both methods may be used within the same command; e.g.,

```
SYSSLV     (P,Q,R)SYS(LISTV,*)
```

This section describes a set of operations LSTNAM, LSTMAK, and LSTRES for assigning names to lists and blocks of storage, thus enabling one subsequently to call them by these names in the appropriate subroutines. The operations LSTOBT, LSTSET, and LSTMPR serve as auxiliary operations. The facilities are especially useful whenever the items of the list are to be filled in at run time or whenever the items do not form a contiguous block in core. A list has two parameters, which are its BES address and its length. These can be set at assembly time by LSTNAM, LSTMAK, and LSTRES or changed at run time by LSTSET and LSTMPR (see Section 3.2). List parameters and list names are not indexable, but the addresses where they are stored or to be stored are. Each item in the specified contents of a list may be tagged. Index registers are preserved with the exception of index register four.

5.2 *List Naming Operations*

ttl	LSTNAM	bes,**lng**,[VAR]	name	(a)
ttl	LSTMAK	(items),[VAR]	make	(b)
ttl	LSTRES	**lng**,[VAR]	reserve	(c)
	LSTOBT	[(abes)],[(lng)],ttl	obtain	(d)
	LSTSET	ttl,[abes],[lng]	set	(e)
	LSTMPR	ttl,[(**op** oper)],[(**op** oper)]	modify	(f)

ttl = name for list
bes = BES address of list
abes = address where "bes" is or is to be stored
lng = length of list
items = contents of list
(**op** oper) = 7090–94 machine operation and an operand separated by a blank.

5.3 *Descriptions*

(a) ttl LSTNAM bes,**lng**,[VAR]

Declare a set of items in a contiguous block of length *lng* to be a list whose BES address is *bes*, name it *ttl*, and set the list parameters to *bes* and *lng*. If *VAR* is present, the list is assumed to consist of variables (specified in the manner indicated by the last previous VARTYP declaration). If *VAR* is not supplied, the list is assumed to consist of rational functions, polynomials, etc. (i.e., of symbolic addresses of pointers.)

(b) ttl LSTMAK (items),[VAR]

Declare the set whose elements are the subarguments in *items* to be a list, name it *ttl*, and set the list parameters accordingly. *VAR* is as described in LSTNAM. The items need not be in a contiguous block as in LSTNAM.

(c) ttl LSTRES **lng**,[VAR]

Reserve remotely a block of length *lng* for a list, name it *ttl*, and set the list parameters to the BES address of the block and *lng*. *VAR* is as described in LSTNAM. The list is to be filled in at run time (e.g., by SYSSLV).

(d) LSTOBT [(abes)],[(lng)],ttl

Store the BES address of the list whose name is *ttl* in location *abes* and store its length in *lng*. The bracketed arguments may actually

consist of several subarguments, and if an argument is an integer equal to seven or less, it refers to an index register.

(e) LSTSET ttl,[abes],[lng]

Set the BES address of the list whose name is *ttl* to the contents of *abes* and set its length to the contents of *lng*.

(f) LSTMPR ttl,[(**op** oper)],[(**op** oper)]

Modify the BES address of the list whose name is *ttl* using the 7090-94 machine operation *op* with operand *oper* specified by the first bracketed argument. Modify the length of the list in a similar manner as indicated by the second bracketed argument. Typically, the operation is ADD or SUB and the operand is the address of some increment or decrement.

5.4 *Example*

The following example shows how list naming can be used to good advantage. We are given polynomials (A_1, $\cdots$, A_y; $y \leqq 15$), a set of variable names (m_1, $\cdots$, m_y; $y \leqq 15$), and polynomials (F_1, $\cdots$, F_y; $y \leqq 15$). It is desired to form a set of polynomials (G_1, $\cdots$, G_y; $y \leqq 15$) in the following way, where m_i: A_i means A_i is substituted for m_i

$$\begin{aligned} G_1 &= F_1(m_1{:}A_1) \\ G_2 &= F_2(m_1{:}A_1, m_2{:}A_2) \\ &\vdots \quad\quad \vdots \\ G_y &= F_y(m_1{:}A_1, m_2{:}A_2, \cdots, m_y{:}A_y). \end{aligned}$$

Assume that the F_i's, A_i's, M_i's, and G_i's are stored forwards in blocks whose BES addresses are F, A, M and G respectively and that the parameter y is in location Y. The following program will perform the substitution.

```
POLS    LSTNAM   A,15
VARS    LSTNAM   M,15
                          Define the lists thus setting the list
                          parameters to (A,15) and (M,15)
        LSTMPR   POLS(SUB  Y) (SUB  =15)
        LSTMPR   VARS(SUB  Y) (SUB  =15)
                          Initialize the list parameters to (A -
                          y,0) and (M - y,0).
        LXA      Y,1
```

```
LOOP    LSTMPR   POLS(ADD  =1) (ADD  =1)
        LSTMPR   VARS(ADD  =1) (ADD  =1)
        POLSST   (G,1) (F,1) (POLS,*) (VARS,*)
        TIX      LOOP,1,1
                          Increment the list parameters by
                          one at each repetition of the above
                          loop.
          ⋮
F       BES      15
A       BES      15
M       BES      15
G       BES      15
Y
```

VI. OUTLOOK

Our experience has shown us that the present handling of linear systems has its limitations. Large linear systems are always difficult to put into canonical form, and even a relatively simple set of system coefficients can grow quite rapidly throughout the course of SYSCFM and cause some form of overflow. This growth becomes coupled with the growth produced by the greatest common divisor algorithm,* thus making the inadequacies of the latter most apparent. The success or failure of SYSCFM depends less on the dimensions of the system and more on the internal structure and size of the individual coefficients. Moreover, it is very difficult to tell by looking at the input array whether the structure and size at a later stage of the reduction will cause trouble. This difficulty is illustrated in that SYSCFM succeeded in reducing a 9×9 array with large, apparently complex, entries,† but failed in a related queuing theory problem to reduce a 10×10 array whose entries averaged only two or three terms. It can at least be said that there will be no GCD problems if the original array consists of all rational numbers.

The subroutine SYSCFM is perhaps too comprehensive. A series of orders which would enable one to perform the Gaussian elimination method a step at a time, leaving the choice of row and column permutations completely up to the user, might be useful. SYSPRT could then be called at any time during the reduction. A routine for evaluating determinants, if available, would enable the solution of nonsingular systems by Cramer's method as an alternative.

* See Ref. 2, pp. 791–794.
† See Ref. 1, pp. 2090–2092.

The growth problem could be reduced by allowing multiple precision polynomial coefficients and by allowing a polynomial to be represented as a product of polynomials (not necessarily irreducible). Thus one could compute the GCD as a product of simpler GCD's. To do this would require the ability to have a data structure hierarchy in the data buffer more complicated than that of a rational function.* This ability would also enable a linear system itself to be such a data structure rather than an array in the main program.

A new version of ALPAK (to be called ALPAKB) is now being developed. Its foundation is a programming system[5] called OEDIPUS (Operating Environment with Dynamic storage allocation, Input-output, Public push down list, Unhurried diagnostics, and Symbolic snaps) which provides for the dynamic storage allocation of such data structures, among other things. ALPAKB will also include multiple precision integer arithmetic which will handle polynomial coefficient overflow.

VII. ACKNOWLEDGMENT

I would like to thank W. S. Brown for many valuable suggestions and discussions concerning every aspect of this paper.

REFERENCES

1. Brown, W. S., The ALPAK System for Nonnumerical Algebra on a Digital Computer — I: Polynomials in Several Variables and Truncated Power Series with Polynomial Coefficients, B.S.T.J., **42,** Sept., 1963, p. 2081.
2. Brown, W. S., Hyde, J. P., and Tague, B. A., The ALPAK System for Nonnumerical Algebra on a Digital Computer — II: Rational Functions of Several Variables and Truncated Power Series with Rational-Function Coefficients, B.S.T.J., **43,** March, 1963, p. 785.
3. Hyde, J. P., unpublished work.
4. Stoll, Robert R., *Linear Algebra and Matrix Theory,* McGraw-Hill, New York, 1952. (See especially Chap. 1.)
5. Brown, W. S., and Leagus, D. C., OEDIPUS: Operating Environment with Dynamic storage allocation, Input-output, Public push down list, Unhurried diagnostics, and Symbolic snaps, to be published.

* See Ref. 2, p. 794.