ELLIOTT 900

Volume 2 : PROGRAMMING INFORMATION

Part 6 : APPLIED PROGRAMMING

Section 2 : ALGOL MATRIX PACKAGE (ALMAT)

CONTENTS

Chapter 1: INTRODUCTION

1.1 Purpose

The Matrix package contains a set of procedures which perform many of the standard operations of matrix arithmetic. Each matrix to be operated on is held in a two dimensional real array.

1.2 Configuration

The minimum configuration is a basic 900 series processor with 8192 word store.

On this minimum configuration about 5 of the large procedures could be used together with a realistic sized program and several 15 by 15 matrices. The largest matrix that could be inverted on a basic 8192 word store is about 35 by 35.

On a 16384 word store the corresponding largest matrix would be about 70 by 70.

1.3 Form of Distribution

ALMAT is distributed as a single source code tape of procedures in 900 series Algol code. Not all the procedures will be required for any one program, the unnecessary procedures should be edited out.

1.4 Method of Use

The procedures should be incorporated into the head of the user's Algol program, normally by an editing process which removes unwanted procedures.

Chapter 2: FUNCTIONS

2.1 Parameters

The parameters of the routines are, in general, array names. The vectors and matrices which are used as actual parameters must have been declared in the main program as TWO-DIMENSIONAL arrays with appropriate subscript bounds, e.g., A column - or row - vector A should be declared as A [1:m, 1:1] or A [1:1, 1:m].

The first parameter usually gives the array in which the result of the operation is stored, and subsequent parameters specify the operands. Storage space for results is allocated when the procedure is called, the subscript bounds of the operands then being known. In general, the procedures make all tests for compatibility etc., which are necessary for the performance of the operation intended.

2.2 Summary of Procedures

Each procedure is described by specifying its parameters and method of use in pseudo-Algol form.

Throughout this summary

A, B and C	represent two-dimensional real arrays (matrices)
X	represents a real scalar, e.g., a matrix element
i and j	represents integers used as suffices to indicate particular array elements.

2.2.1 MXDIFF

Procedure	MXDIFF (A) becomes: (B) minus: (C);
Array	A, B, C;
Comment	This procedure subtracts array C from array B and stores the result in array A. A may be the same as either B or C. MXDIFF uses about 250 words of store.;

2.2.2 MXSUM

Procedure MXSUM (A) becomes: (B) plus: (C);
Array A, B, C;
Comment this procedure adds array B to array C and stores the result in array A. A may be the same as either B or C.
MXSUM uses about 250 words of store.;

2.2.3 MXCOPY

Procedure MXCOPY (A) becomes: (B);
Array A, B;
Comment this procedure which is used by mxquot copies array B, the copy becoming array A. MXCOPY uses about 180 words of store.;

2.2.4 MXNEG

Procedure MXNEG (A) becomes minus: (B);
Array A, B;
Comment this procedure negates array B. The result is stored in array A. Array A may be the same as array B;
MXNEG uses about 190 words of store.;

2.2.5 MXPROD

Procedure MXPROD (A) becomes: (B) times: (C);
Array A, B, C;
Comment this procedure forms the matrix product of arrays B and C. The result is stored in array A. A must not be the same array as B or C. If the arrays are incompatible the message "mxprod error" is displayed and the program is terminated.
MXPROD uses about 420 words of store.;

2.2.6 MXTRANS

Procedure MXTRANS (A) becomes transpose of:(B);
Array A, B;

Comment this procedure forms the transpose of array B and stores the result in array A. Array A may not be the same as array B. If A and B are incompatible the message "mxtrans error" is displayed and the program is terminated.
MXTRANS uses about 280 words of store.;

2.2.7 SCPROD

Procedure SCPROD (A) becomes A times the scalar: (x);
Value x; real x; array A;
Comment this procedure multipies array A by the scalar x in situ.
SCPROD uses about 90 words of store.;

2.2.8 MXQUOT

Procedure MXQUOT (B) becomes: (A) to minus one times: (C);
Value A; array A, B, C;
Comment this procedure which uses mxcopy solves a set of simultaneous equations A * B = C in one procedure call.
The method used is Gaussian elimination. The solution array B has the same number of rows and columns as the right hand side, array C.
MXQUOT uses the procedures MXCOPY and SOLVMX. Since the array A is called by value an extra copy of A is made in store. Where store is critical the procedure SOLVMX may be used directly, saving time and store space. SOLVMX (A, B) solves A * X = B, placing the result in B and overwriting the original values in A.
If at any stage the ratio of the new pivot to the greatest pivot is less in magnitude than 10^{-6}, the following message is displayed:

MXQUOT: SINGULAR STAGE∝SIZE∫ PIVRATIOγ

and WAIT is entered. It is possible to continue but the results are unlikely to contain any significant figures.

If the arrays are incompatible, the message

MXQUOT ERROR

is displayed and the program is terminated.
MXQUOT uses about 20 words of store and the procedures MXCOPY and SOLVMX. SOLVMX uses about 460 words and the procedures PERMROWS and CROUT, which occupy about 700 words.;

2.2.9 INVMX

Procedure INVMX (A);
Array A;
Comment this procedure inverts the non-singular array A in situ. The program uses the Gaussian elimination method, searching for the maximum element in each column and using these elements as pivots. If at any stage the ratio of the new pivot to the greatest pivot is less in magnitude than 10^{-6}, the following message is displayed:

INVMX: SINGULAR STAGE∝ SIZE∕ PIVRATIO∀

and a data wait is entered. It is possible to continue, but the results are unlikely to contain any correct significant figures. Failure at the first stage is specially distinguished. If A is not square the message "invmx error" is displayed and the program is terminated.
INVMX uses about 630 words of store, and the procedures PERMROWS and CROUT. PERMROWS and CROUT use about 700 words;

2.2.10 PRINTMX

Procedure PRINTMW (A);
Array A;

the format current when the procedure is called.
The format is specified before the procedure call, e.g. :

```
punch (2)
prefix
freepoint (4)
printmx (A)
```

Each row is printed on a new line. No row or column numbers are printed nor is any facility included for the printing of large matrices. PRINTMX uses about 80 words of store.;

In cases where the number of columns is too great for the matrix to be printed on one level either of the following output procedures may be used:

2.2.11 PRINTCOL

Procedure	PRINTCOL (A);
Array	A;
Comment	this procedure prints the array A by columns. The elements of a column are all output on the same line and each column is output on a new line. The format for printing is set before the procedure is called. PRINTCOL uses about 70 words of store.;

2.2.12 MXOUTPUT

Procedure	MXOUTPUT (A, m, n,);
Value	m, n;
integer	m, n;
Array	A;
Comment	this procedure prints the array A by rows (see example) together with its row and column numbers. The row and column numbers are printed in the format digits (3). If the array has so many columns that it will not fit on the available output sheet, then the procedure will arrange for the array to be output on more than one level (see example)

The parameters of the procedure are:-

A, a real array,
m, the number of characters available across the output sheet and
n, the number of characters occupied by each element of the array.

The format for printing elements of the array is set by the programmer before the procedure is called. From this n, the number of characters per element, may be calculated e. g.
freepoint (t) : t + 2 characters are required by each element.
aligned (r, s) : r + s + 2 characters are required by each element.
MXOUTPUT occupies about 230 words of store and uses the procedures SPACES (30 words) ;

2. 2. 13. READMX

Procedure READMX (A)
Array A;
Comment this procedure reads any real number and places the result in an element of the matrix A. Successive elements are placed in the same row of the matrix until the row has been filled;
Reading takes place on the device current when the procedure is called. The device setting may be altered before readmx is called. e. g. READER (3); READMX (B);
READMX occupies about 80 words of store.

Chapter 3: TIME TAKEN

Some typical times are given below:

SIZE OF MATRIX:	5 BY 5	10 BY 10	20 BY 20
903			
INVMX	10 sec.	55 sec.	5 min 30 sec.
MXPROD	4.5 sec.	32 sec.	3 min 42 sec.
MXSUM	2 sec.	6 sec.	22 sec.
SCPROD	0.5 sec.	2.5 sec.	9 sec.
905 1 microsec store			
INVMX	1.1 sec.	6.2 sec.	36.7 sec.
MXPROD	0.5 sec.	3.6 sec.	24.7 sec.
MXSUM	222 msec.	667msec.	2.4 sec.
SCPROD	56 msec.	280msec.	1.0 sec.
905 2 microsec store			
INVMX	1.9 sec.	10.6 sec.	1 min 3.5 sec.
MXPROD	865 msec.	6.2 sec.	42.3 sec.
MXSUM	385 msec.	1.2 sec.	4.2 sec.
SCPROD	96 msec.	481 msec.	1.7 sec.

Chapter 4: OPTIMISATION

The full package of procedures is provided for the convenience of the user. In the case of procedures such as MXCOPY, the main advantage is to save the programmer the labour of writing out multiple for statements. However, where the procedure MXCOPY is used 3 or more times, its use will save program storage space.

The user who is interested in store economy may easily find improvements to the package. Among the more obvious ways of reducing store used would be to remove some or all of the many compatibility tests. In many cases if these tests are removed but the condition is violated an Algol system runtime error will be given.

The procedures were translated from the Elliott 503 and 4100 Algol matrix packages with suitable changes for use on the 903. They will run on the 4100 unchaged.

Very substantial savings in time could be achieved by converting some of the procedures into SIR code procedures. In most cases the use of code will also save program storage space. The compatibility tests could be expressed in code more elegantly than in Algol.

Chapter 5: EXAMPLES

5.1 Example 1
Evaluate E: = A * B + C^{-1}
where E and C are 5 x 5 matrices
A is a 5 x 10 matrix
and B is a 10 x 5 matrix
A and B are to be read from paper tape and C
is defined by C [i, j] = 1 + 1 / (i + 2* j).

```
EVALUATE;

Comment     (The procedures readmx, mxprod, invmx,
            mxsum and printmx are edited in at this point)
Begin array   a [1 : 5, 1 : 10], b [1 : 10, 1 : 5];
integer       i, j;
```

```
Comment     d is work space;
            for i: = 1 step 1 until 5 do
            for j: = 1 step 1 until 5 do
            c [ i, j] : = 1 + 1/ (i + 2 * j);
        readmx (a); readmx (b);
        mxprod (d, a, b);       comment D: = A*B;
        invmx (c);              comment C: = C^-1
        mxsum (e, c, d);        comment E: = A*B+C^-1
        printmx (e);
End     evaluate
End     matrix package;
```

A tape for the above program could be made up by the following edit

```
IL EVALUATE;
DC;
FL "PROCEDURE" PRINT 2;
FC;
DL "END" MXNEG;
FL "END" MXPROD;
DL "END" SCPROD;
FL "END" INVMX;
DL "END" MXQUOT;
FL "END" PRINTMX;
DL "END" MXOUTPUT;
IH
```

(H) (Halt code)

The tape produced by this edit is input to the Algol Translator (Tape 1). This ends with a halt code and should be followed by input of the program tape at entry point 9.

The program tape starts with:

"BEGIN" "ARRAY" A[

and ends with:

"END" MATRIX PACKAGE;

Alternatively, the program could be copied onto the end of the edited matrix procedures.

5.2 Example 2

Example of output using mxoutput with the format set to freepoint (5)

ROW/COL	1	2	3	4	5	6	7
1	.00000	14.000	21.000	28.000	35.000	42.000	49.000
2	.00000	.00000	21.000	28.000	35.000	42.000	49.000
3	.00000	.00000	.00000	28.000	35.000	42.000	49.000

ROW/COL	8	9	10	11	12	13	14
1	56.000	63.000	70.000	77.000	84.000	91.000	98.000
2	56.000	63.000	70.000	77.000	84.000	91.000	98.000
3	56.000	63.000	70.000	77.000	84.000	91.000	98.000

ROW/COL	15	16	17	18	19	20
1	105.00	112.00	119.00	126.00	133.00	140.00
2	105.00	112.00	119.00	126.00	133.00	140.00
3	105.00	112.00	119.00	126.00	133.00	140.00

END OF MATRIX

Chapter 6: STORE USED

Approximate figures for the store used by each procedure are given in chapter 2.2

The total store taken by all the procedures is about 3950, including scalar variables. However, it is most unlikely that all the procedure will be used in one program.