NATIONAL
ELLIOTT
COMPUTER
APPLICATION
PROGRAM

Reference Number LM21

December 1963

PROVISIONAL

AN APPLICATION OF THE NATIONAL-ELLIOTT 803 TO MATHEMATICS

SMS : A SYSTEM OF MATRIX SUBROUTINES FOR USE WITH THE 803 AUTOCODE

Elliott Computing Division, Borehamwood, Hertfordshire, ELStree 2040

CONTENTS

Introduction

803 SMS (A System of Matrix Subroutines) is a collection of subroutines, written in autocode and intended for matrix computations. When using this system it is possible to write in an ordinary autocode program matrix instructions which refer directly to these matrix subroutines (M-subroutines). The system is used as follows:

(a) A program containing matrix instructions (M-program) is written by the programmer.

(b) This M-program is converted to 803 Autocode by means of the M-translator, which then copies the required subroutines from the M-library tape onto the mnemonic tape it produces, and finally outputs the START instruction.

(c) The 803 Autocode mnemonic tape is then translated in the usual way. Normally, it is unnecessary to perform the M-translation more than once, since any errors occurring may be corrected on the Autocode tape.

When using SMS, integer variables P, P1, ... and V, V1 ....are reserved for parameters. Floating-point variables U, U1 .... are used similarly, and floating-point variables Z, Z1 .... are used as working space.

SMS is not a limited system; it can be enlarged with such special routines as are found necessary.

Program and Machine

The M-translator uses only fixed-point working and can be used on any machine having at least a 4096-word store.

The present M-library tape is intended for use with A103 Issue 3 and is thus only suitable for use on machines with an automatic floating-point unit. All M-library tapes will be marked with the issue of A103 with which they are intended to be used.

The Finnish Cable Company, who originated SMS, have undertaken to maintain it.

A SYSTEM OF MATRIX SUBROUTINES
FOR
ELLIOTT 803 AUTOCODE (803 SMS)

1. The Names of M-subroutines

Every M-subroutine has a name of its own, which makes it possible to refer to this routine.

Example: ADD and MULT are the names of the matrix addition and multiplication subroutines.

Each M-subroutine name has a corresponding Autocode reference number. The M-translator chooses these reference numbers automatically within certain limits.

2. Matrix Instructions

The entry to an M-subroutine for the execution of some matrix operation, is done by a matrix instruction. The matrix instruction is usually divided into two parts, the first being the name of the M-subroutine and the second a list of the parameters by which the execution of the instruction is defined. Comments may also be added to the matrix instruction, if required.

Examples of matrix instructions:

```
'COPY'  (P, Q:N+1, 1)
'INVERT' MATRIX  L
'MULT' A*B = C  (C, A, B:M, N, K)
'READ J1'  (Q:M, K*R:0.0001)
```

The construction of the matrix instruction is as follows:

< matrix instruction > ::=

' < M-subroutine name > ' < comment 1 > lf |

' < M-subroutine name > ' < comment 1 > (< parameter stack >) < comment 2 > lf

<M-subroutine name> ::=

<letter>|<digit>|<M-subroutine name><letter>|

<M-subroutine name><digit>

<parameter stack> ::=

<address parameters>|

<address parameters> : <dimension parameters>|

<address parameters> : <dimension parameters> : <special parameters>

<address parameters> ::= <parameter list>

<dimension parameters> ::= <parameter list>

<special parameters> ::= <parameter list>

<parameter list> ::= <empty>|<parameter word>|

<parameter list> , <parameter word>

<parameter word> ::= <empty>|<right hand side of an arithmetic Autocode instruction>

<comment 1> ::= <A string not containing characters (or lf>

<comment 2> ::= <A string not containing character lf>

The matrix instruction can be labelled with a reference number.

The present M-translator imposes some further limits upon the matrix instruction.

(1) In the M-subroutine name only the first 6 non-shift characters are interpreted. The characters sp and bl are ignored. All other characters except the letters, digits, sp and bl are forbidden.

Example: TRANSPOSE is equivalent to TRANSP and MULT 01 is the same as MULT01. The names MULT A*B, - ADD , DIAG:TRACE are forbidden.

(2) Also in the parameter word only the first 6 non-shift characters are interpreted. All characters except) : , lf are allowed. The characters sp and bl are ignored.

Example: 0.3 , N , N+1 , A1-B13 , - LI*H are permitted parameter words, but A(I+3) and N+12345 are forbidden, the first because it contains) and the second because it contains seven characters. The M-translator interprets them in the wrong form A(I+3 and N+1234.

Warning The M-translator does not check whether the parameter word is really the right-hand side of the arithmetic instruction.

3. Interpretation of Matrix Instructions

The M-translator converts the matrix instructions to mnemonic Autocode.

Thus:

(a) the parts < comment 1 > and < comment 2 > are ignored,

(b) < parameter stack > is converted into a sequence of parameter instructions,

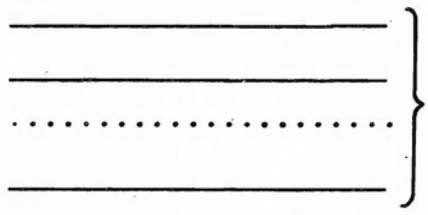
(c) < M-subroutine name > is changed into SUBR-n instruction, where n is the reference number of the corresponding M-subroutine,

(d) < M-subroutine name > and < parameter stack > will be comments on the SUBR n instruction.

A general matrix instruction

'< M-subroutine name >'< comment 1 > (< parameter stack >)< comment 2 >

is thus changed into the form



SUBR n :: <M-subroutine name>(<parameter stack>)

The parameter instructions need closer attention. The parameter instructions corresponding to the general parameter stack

<address parameters>:<dimension parameters>:<special parameters>

or p_o, p_1, p_2... : v_1, v_2, v_3... : u_o, u_1, u_2...

take the form

P0=p_o
P1=p_1
P2=p_2

V1=v_1
V2=v_2
V3=v_3

U0=u_o
U1=u_1
U2=u_2

If a parameter word is empty, the corresponding parameter instruction is left out.

Example: parameter stack

100, Q1, Q2 : M, N-3, K1, , K2 : AI

is converted into the form

```
P0=100
P1=Q1
P2=Q2
V1=M
V2=N-3
V3=K1
V5=K2
U0=AI
```

and the parameter stack

```
::,,,1
```

into the form

```
U3=1.
```

If desired it is possible to write the parameter instructions in the final form directly into the M-program.

Example Instructions

```
12)'READ J1' MATRIX B (Q:M,N-1:0.0001)
```

and

```
12)P=Q
V1=M
V2=N-1
U=0.0001
'READ J1'
```

are equivalent in the M-program.

4. Matrix Storage and Interpretation of Parameters

The M-subroutines need working space for the parameters and the matrix elements. The writer of the M-program is responsible for the reservation of this space in the setting instructions.

In the previous section we saw that different parameter values are specified by P-, V- and U-variables. Of these the P-variables (address parameters) and the V-variables (dimension parameters) are of integer type, but the U-variables (special parameters) are of floating point type.

For the matrices and vectors, floating point variables Z, Z1, Z2,.. are used as working space (matrix space). The matrices are stored by columns in successive Z-locations. In some routines the form of matrix storage is different (e.g. diagonal matrices and symmetric matrices).

The address of the matrix is specified by the P-variables and it is defined in the following way: If the first element of the matrix is in Z(p+1), the address of the matrix is p. In general P is the address of the result matrix and P1, P2,... are the addresses of the operand matrices.

The dimensions of the matrix are specified by the V-variables. Usually V1 is equal to the number of rows and V2 to the number of columns. V-variables are also often used by the M-subroutines in picking up special information from the main program. The U-variables have the same task in the case of special floating point parameters.

The detailed interpretation of the parameters is given in the list of M-subroutines. In the M-subroutine descriptions the following notation is used for the matrices:

(p:m, n) is an m x n matrix, the address of which is p .

Examples

1. Matrix multiplication $(p:m, s) = (p_1 : m, n) * (p_2 : n, s)$ is carried out by the instruction

   'MULT' $(p, p_1, p_2 : m, n, s)$

   which the M-translator converts into the form

   P0=p

   P1=p_1

   P2=p_2

   V1=m

   V2=n

   V3=s

SUBR r:: MULT(p, p_1, p_2 : m, n, s),

where r is the reference number of the MULT-routine chosen by the M-translator.

2. The matrix (p_1 : m, n) is copied into the place of the matrix (p:m, n) by the instruction

'COPY' (p, p_1 : m, n)

3. The matrix (p : n, n) is inverted in situ by the instruction 'INVERT' (p : n, n)

4. The matrix (p : m, n) is read and punched in J*code by the instructions

'READ J2' (p : m, n)

'OUT J2' (p : m, n)

(for a definition of J*code see Appendix 1)

5. Alteration of the Parameter Values in the M-subroutines

The M-subroutines obey the following rules:

(1) The values of the address parameters P, P1, P2, ... are preserved.

(2) The dimension parameters V1 and V2 change so that after the completion of a matrix instruction

V1 = number of rows of the result matrix

V2 = number of columns of the result matrix

(3) The values of other parameters (V, V3, V4, ... U, U1, U2, ..) may be altered within M-subroutines. Further information about this is to be found in each M-subroutine description. With these arrangements useless parameter instructions are avoided when using several matrix subroutines one after another.

Example The matrix (p_1 : n, n) is copied in the place of the matrix (p : n, n) and inverted in situ in this matrix space by

the instructions

'COPY' $(p, p_1 : n, n)$

'INVERT'.

6. Control Instructions

In the M-program, besides matrix instructions, so called control instructions are employed. These are 'MATRIX', 'SCALAR', 'SETM' m and 'START' n.

The 'MATRIX'-instruction announces the transfer from an Autocode block, consisting of ordinary autocode and some special instructions, to a matrix block, consisting only of matrix instructions. (The present M-translator allows the matrix block to contain ordinary autocode, provided the character ' does not appear.)

The 'SCALAR'-instruction announces the opposite transfer, i.e. from the matrix block to the autocode block.

'MATRIX' and 'SCALAR' thus form statement brackets for matrix instructions.

The M-translator always assumes that the M-program starts with an Autocode block, unless otherwise stated by the 'MATRIX'-instruction.

The instruction 'SETM' m, where m is a positive integer, determines that the reference number m is reserved for the next new M-subroutine not mentioned previously in the present M-program. The subsequent new M-subroutines are given the reference numbers m+1, m+2, etc., until a new 'SETM'-instruction is given. The 'SETM'-instruction must be placed in the matrix block and the first 'SETM' must be written before the first matrix instruction of the M-program.

Example Let the first matrix block of the M-program be

```
'MATRIX'
'SETM' 10
'READ J2' (0:M,N)
'READ J2' (M*N)
```

'ADD' (2*P, 0, M*N)

'SETM' 5

'OUT J2'

'SCALAR'

the M-subroutines READ J2, ADD and OUT J2 thus are given the reference numbers 10, 11 and 5.

'START' n is always written at the end of the M-program. It replaces the START n instruction of ordinary autocode programs. 'START' n is not allowed to be in a matrix block.

N.B. TITLE-instructions written in the normal way may cause incorrect activity of the M-program if the character ' appears in the TITLE-instructions, or it may cause the alteration of the title if it contains a line starting with an sp-character. These inconveniences are avoided by writing TITLE-instructions in the form 'TITLE'. 'TITLE'-instruction can be written only in an Autocode block.

7. Setting Instructions

At the beginning of the M-program the setting instructions SETS, SETV, SETF and SETR are written in the same way as in an ordinary autocode program. The M-translator does not in any way change these setting instructions nor does it notice faults in them. Thus the writer of the M-program is responsible for the setting instructions being in order.

The M-translator, however, helps the writer of the M-program in checking the setting instructions, in that when the translating work is finished, information is received about the need of memory space for the M-subroutines and of the minimal setting instructions they require. When writing the M-program space must be reserved:

(1) for all the variables, function subroutines and reference numbers needed in the M-subroutines and appearing in the M-program, in accordance with the proper subroutine descriptions,

and

(2) also for all other variables, functions and reference numbers used in the M-program.

It is a good idea to check, using the control information given by the M-translator, that the setting instructions are sufficient for the M-subroutines and that the reference numbers do not overlap.

For the requirements of most standard subroutines the variable and function setting instructions

SETS P(2)V(20)

SETV U(2)Z(matrix space)

SETF INT SQRT

are sufficient.

However, in the M-subroutines INVERT and INV DET, the requirement of the V-variables is V(10+n), where n is the dimension of the largest matrix to be inverted.

When reserving reference number space by the instruction SETR, it should be observed that 803 SMS needs for its auxiliary routines (see Section 8) 5 reference numbers, which it reserves in accordance with the last 'SETM'-instruction of the M-program. Also, two or more M-subroutines can be united into one subroutine body, whereupon the M-translator has to reserve reference numbers for all M-subroutines belonging to the same subroutine body. Thus more reference numbers may be needed than the number of M-subroutines used in the M-program implies. The reference number requirements of each subroutine appear in the M-subroutine descriptions.

It is recommended that the lowest reference numbers be reserved for the M-program's own purposes and that a single 'SETM'-instruction be given to specify the reference numbers of the M-subroutines after this. It is then impossible for the M-program's own reference numbers and the reference numbers of the M-subroutines to overlap.

8. 803 SMS Basic Routines

In order to facilitate the testing of matrix programs, some Basic Routines are inserted into 803 SMS. These routines are as follows:

ERROR IND	Error Indication
	The M-subroutines can, by the aid of this routine, report the faults appearing in their use.

ERROR STOP	Error Stop
	If the performance of some matrix operation is impossible, after the error indication the ERROR STOP will follow.
EXIT	Exit-routine
	Exit from most of the M-subroutines takes place through the EXIT-routine. The task of this routine is to specify the use of the routines. CHECK PRINT and PRINT VAR.
CHECK PRINT	CHECK PRINT of the RESULT MATRIX
PRINT VAR	PRINT the VALUES of the VARIABLES
	The use of these routines is possible after all the matrix operations, where exit to the main program takes place through the EXIT-routine. The use is controlled from the number generator.

9. Examples of M-programs (see also Appendix 1)

Example 1

To produce $C=(A+B)^t$, where A and B are m x n matrices punched on the tape.

Program: It is presumed that the matrices A and B are punched in *code and C is wanted on the tape in J*code, so that every element is to be punched in the form corresponding to the instruction PRINT Z, 3:5. Let $m \leq 50$ and $n \leq 20$.

```
SETS MNQP(2)V(13)
SETV U(2)Z(2000)
SETF INT
SETR 14

1)READ M
READ N
Q=M*N
```

```
'MATRIX'
'SETM' 2
'READ S2' MATRIX A(0:M, N)
'READ S2' MATRIX B(Q:M, N)
'ADD'              (0, 0, Q:M, N)
'TRANSPOSE'        (Q, 0:M, N)
'OUT J2'  MATRIX C(Q:N, M, 3, 5)
'SCALAR'

STOP
'START' 1
```

Example 2

To solve the systems of linear equations

$$AX = B$$

where the matrices A(m x m) and B(m x n) are known. Further the residual matrix R = AX - B is to be produced.

Program: The matrices A and B are presumed to be punched in J*code with a sum tolerance of 0.00001 and the result matrices X and R are wanted in the same code, the former with print parameters k_1, k_2, which are read from the data tape, and the latter in the form corresponding to the instruction PRINT Z, 9. Further it is required that $m(m+3n) \leq 6000$ and $2m(m+n) \leq 6000$.

```
SETS MNK(2)Q(2)P(2)V(64)
SETV U(2)Z(6000)
SETF INT
SETR 17

1)READ M
READ N
READ K1::PRINT PAR.
READ K2
Q=M*M
Q2=M*N
Q1=Q+Q2
Q2=Q1+Q2
```

```
'MATRIX'
'SETM' 2
'READ J1' A(0:M, M: .00001 )
'READ J2' B(Q:M, N)
'COPY'    A(Q2, 0:M, M)
'INVERT'  A
'MULT'    X(Q1, Q2, Q:M, M, N)
LINES 5
TITLE SOLUTION
'OUT J1'  X(Q1:M, N, K1, K2)
'MULT'  AX(Q2, 0, Q1:M, M, N)
'SUB'     R(Q2, Q2, Q:M, N)
LINES 5
TITLE RESIDUALS
'OUT J1'  R(Q2:M, N, 0, 9)
'SCALAR'

STOP
'START' 1
```

Example 3

To compute the greatest eigenvalue λ of the matrix A (n x n) and the corresponding eigenvector y by the following iteration procedure:

Let

$$y_o = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{and } y_k = \begin{bmatrix} y_{1k} \\ y_{2k} \\ \vdots \\ y_{nk} \end{bmatrix}, \quad k = 1, 2, \ldots,$$

where y_k is obtained from y_{k-1} by

$$x_k = A y_{k-1} ,$$

$$y_k = x_k / a_k ,$$

$$a_k = \max_i \left| x_{ik} \right|$$

Then it can be proved, provided certain conditions concerning the matrix A

are in force, that

$$\lim_{k\to\infty} a_k = \lambda \quad \text{and} \quad \lim_{k\to\infty} y_k = y$$

Program: The matrix A is read in J*code. The iteration is stopped when for the first time $|a_k - a_{k-1}| < \varepsilon$, where ε is a number read from the data tape. Further, it is assumed that $n \leq 50$.

```
SETS NXYP(2)V(13)
SETV EL(1)U(2)Z(2600)
SETF INT
SETR 19

1)READ E::EPSILON
READ N
Y=N*N
X=Y+N
L=0

'MATRIX'
'SETM' 3
'READ J2'  A(0:N, N)
'CLEAR'      (Y:N, 1)
'PUT'        (Y:N, 1, 1, 1:  1.0 )   Y0'=(1, 0, 0, . . . , 0)

2)'MULT'     (X, 0, Y:N, N, 1)       ITERATION
'MAX MOD'    (X:N, 1)
L1=U
'SCAL MULT'  (X:N, 1: 1/U )
'COPY'       (Y, X:N, 1)
U=L1-L
L=L1
JUMP UNLESS E%MOD  U@2

LINE
TITLE EIGENVALUE=
PRINT L, 9
LINES 3
TITLE EIGENVECTOR
'OUT J1'     (Y:N, 1, 0, 9)
'SCALAR'

STOP
'START' 1
```

10. Translating the M-program

The M-program is translated into an ordinary autocode program, an A-program, by means of the M-translator as follows:

(1) Place the M-translator in the tape reader and enter 40 0

(2) Place the M-program in the reader and enter 40 16

Thus the M-translator starts to transform the M-program into an autocode program, which is punched on to the output tape. If the M-program is correct as far as the M-translator is concerned, a WAIT occurs when the 'START' instruction at the end of the program has been read.

(3) Place the M-library tape in the reader and change the setting of the keyboard from even to odd or vice versa.

The M-translator reads the library tape, copying the M-subroutines which are required by the M-program, transforming them into autocode form. When all the necessary M-subroutines have been copied, final information is printed, which for a faultless M-program is as follows:

(As an example the final information from Ex. 1 has been taken)

M-SUBROUTINES:

READS2	2	The reference numbers of the M-subroutines needed by the M-program
ADD	3	
TRANSP	4	
OUTJ2	5	
ERRORI	6	Reference numbers of the Basic routines
EXIT	7	
CHECKP	8	
ERRORS	9	
PRINTV	10	
READS1	11	common program body with READ S2

OUTT1	12	common program body with OUT J2
OUTT2	13	
OUTJ1	14	

NUMBER OF LOCATIONS USED FOR M-SUBROUTINES: 385

SETTING INSTRUCTIONS (MINIMUM):

SETS P(2)V(13)

SETV U(2)Z

SETF INT

SETR 14

11. Error Indications during the M-translation

The M-translator gives clear indications of all noticed errors in the M-program. On finding an error the translation does not usually stop, but the M-program is read as far as the 'START'-instruction which is at the end of the program. Then information about the appearance of mistakes is given by punching 50 *'s and PROGRAM IS NOT CORRECT. The mistakes will be found if the punched tape is interpreted.

Punching 50 characters * and PROGRAM IS NOT CORRECT can also come after the reading of the M-library tape. Then the mistakes can be found by studying the final information.

If the translation stops when reading the M-program, the punched tape has to be interpreted, and the M-program corrected in accordance with the error indications received.

12. Translating the A-program

The mnemonic autocode A-program, which is produced from the M-program by aid of the M-translator, is translated into machine code in the same way as all other autocode programs.

Translation of the A-program must be carried out by the autocode translator which is compatible with the M-library tape used when translating the M-program.

The translation usually stops several times at the auxiliary routine PRINT VAR, depending on the words of the machine code block

```
00  1:00 A
00  2:00 B
00  3:00 C
---------
---------
0026:00 Z
```

which appear in this routine. It stops every time a letter is read which is not in the setting instructions. Changing the sign of the keyboard word will make the translation continue.

In order to be sure that the stop in the translation is due to this reason and not caused by a real error, these words are separated from each other by several bl's.

If, when translating the A-program, the translation stops because of an error in the program, the procedure is the same as that for other autocode programs. If the program has to be corrected, the corrections can be made directly into the A-program, avoiding retranslation of the M-program.

13. Running the M-program

The M-program is run in the same way as an ordinary autocode program. When running the M-program, it is possible to use some extra checking facilities in addition to the autocode system's own checking facilities (CHECK instructions, Trace facility). The checking facilities are as follows:

(1) Several M-subroutines automatically check to see if the given matrix operation can be performed. (e.g. During multiplication by the instruction MULT the position of the result matrix is tested to make sure that it does not overwrite the operand matrices.) Information about these errors is received by means of the routine ERROR IND.

(2) After most of the matrix operations (i.e. those using the EXIT-routine) and after all the error indications given by the ERROR IND routine, it is possible to print the values of the desired variables as intermediate results and then continue the program in the normal way. The routine PRINT VAR prints the value of the variables.

(3) When using the EXIT-routine it is possible to get the result matrices of the matrix operation as intermediate results (CHECK PRINT).

14. Routine ERROR IND

The errors occurring in the M-subroutines are indicated by the routine ERROR IND. The errors are given by punching

ERROR NO.

followed by the number which refers to that particular error. The error numbers are fixed so that the two last figures state the reference number of the M-subroutine.

After the error indication the machine reaches a 'WAIT' instruction. The nature of the error determines the next step. Often it is advisable to use the routine PRINT VAR.

15. Routine PRINT VAR

The task of the routine PRINT VAR is to print the values of the variables used in the M-program. The use of PRINT VAR is regulated by the number generator. The routine PRINT VAR is entered after performing the matrix operation, which uses the EXIT routine, if N1 is odd, or after the error indication received from the routine ERROR IND by changing N1 from even to odd or vice versa. We also come to the routine PRINT VAR by entering 40 16:00 n manually, where n is the reference number of PRINT VAR. First the characters +++ and the reference number of the M-subroutine last used or the error number are punched.

Operation after this is as follows:

(1) Clear the number generator.

(2) Set F2 buttons to the telecode value of the desired variable (letter) in a binary form.

(Example. Variable Z has numerical value 26. This is equivalent to F2 = 32).

Set the N1 buttons to the lowest suffix and the N2 buttons to the greatest suffix of the variable.

(3) Choose the printing mode so that

B = 0 means printing in floating point form with 8 digits,

B = 1 means printing in integer form with 12 digits.

(4) Set F1 = 40

(5) Set F1 = 00

PRINT VAR then prints the values of the variables and returns to the situation before phase (1), when new variables can be printed.

The return from the routine PRINT VAR to the main program is done by carrying out phase (1), setting F2 > 26 and carrying out the phases (4) and (5).

16. Routine CHECK PRINT

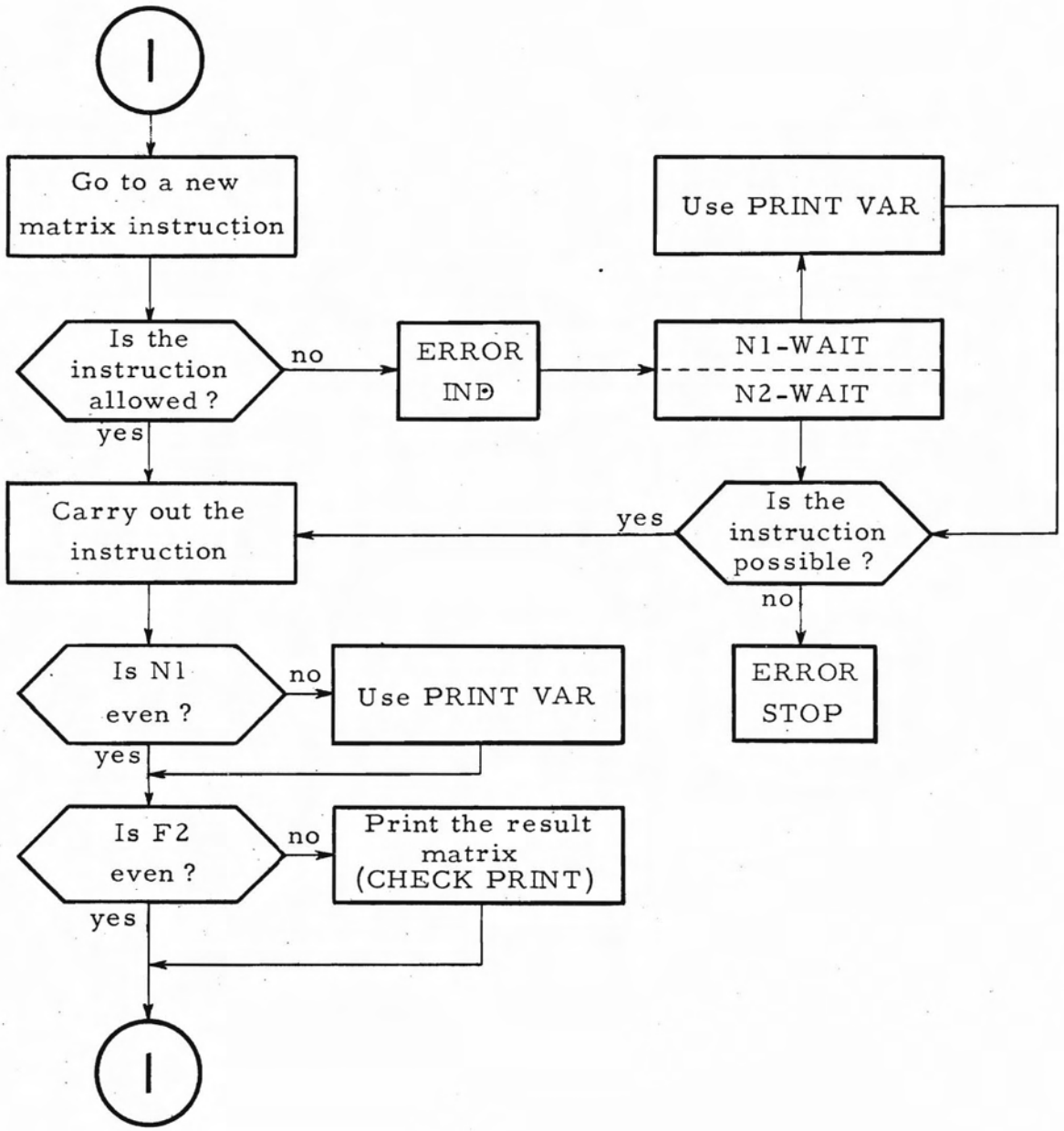
If desired, the result matrices of the M-subroutines using the EXIT routine can be received as intermediate results through the routine CHECK PRINT. This is done, if the last F2 button is depressed when the matrix operation has been carried out. Thus the characters *** and the reference number of the M-subroutine, as well as the matrix itself, are punched in matrix form corresponding to the instruction

'OUT M1' (P:V1, V2, -1, 4, 5)

After the matrix has been printed the main program continues.

17. Routine EXIT

The use of the routines PRINT VAR, CHECK PRINT and ERROR IND is regulated by the EXIT routine, which links most of the M-subroutines to the main program. The scheme below makes the function of the EXIT routine clear.


I
Go to a new matrix instruction
Is the instruction allowed ?
no
ERROR IND
N1-WAIT
N2-WAIT
Use PRINT VAR
yes
Carry out the instruction
yes
Is the instruction possible ?
no
ERROR STOP
Is N1 even ?
no
Use PRINT VAR
yes
Is F2 even ?
no
Print the result matrix (CHECK PRINT)
yes
I

18. Acknowledgement

This program was submitted by S. Mustonen of the Finnish Cable Works Co. Ltd.

Mustonen's original manual was retyped in this form in Elliott Computing Division.

LIST OF M-SUBROUTINES, 1.3.1963

Comments

1. M-subroutines belonging to the same subroutine body are united by brackets in the list. M-subroutines using EXIT-routine are marked by the character X.

2. In all M-programs, in which M-subroutines belonging only to this list are used, the following variable and function setting instructions are sufficient:

SETS P(2)V(20)
SETV U(2)Z(matrix space)
SETF INT SQRT

However, in the instructions INVERT and INV DET the minimum setting for the V-variables is V(10+n), where n is the dimension of the largest matrix being inverted.

Setting INT is needed only when using the instructions OUT J1, OUT J2, OUT T1 and OUT T2.

The function SQRT is needed only in the instructions NORM C and NORM R.

3. The matrix codes *code, J*code and J*Δ code mentioned in connection with the Input-Output operations are defined as follows:

m x n matrix (a_{ij}) is *coded, if it is punched in the form

a_{11}
a_{21}
...
a_{m1} } 1. column
* check sum
a_{12}
a_{22}
...
a_{m2} } 2. column
* check sum
.
.
.

a_{1n}
a_{2n} n. column
$\vdots$
a_{mn}
$*$ check sum

m x n matrix is J*coded, if the punched form is

J=1
1 a_{11}
2 a_{21}
$\vdots$
m a_{m1}
$*$ check sum
J=2
1 a_{12}
2 a_{22}
$\vdots$
m a_{m2}
$*$ check sum
$\vdots$
J=n
1 a_{1n}
2 a_{2n}
$\vdots$
m a_{mn}
$*$ check sum

The symmetric n x n matrix (a_{ij}) is punched in J*Δ code as upper triangular matrix by columns

J=1

1 a_{11}

* check sum ($= a_{11}$)

J=2

1 a_{12}

2 a_{22}

* check sum ($= a_{12}+a_{22}$)

J=3

1 a_{13}

2 a_{23}

3 a_{33}

* check sum ($= a_{13}+a_{23}+a_{33}$)

etc.

The matrix elements must be punched so that they can be read by the READ-instruction for floating point variables. The numbers of columns and of rows that appear in the J* code and the J*Δ code must be punched so that they can be read by the READ-instruction for integer variables.

4. The notation (p:n diag), used in connection with some instructions, means a diagonal matrix which is stored as a vector (p:n, 1) formed by diagonal elements.

5. The order of the M-subroutines in the list is roughly the same as in the M-library tape 1.3.63.

MATRIX INSTRUCTION	OPERATION		NR. OF LOCATIONS	P(.)	V(.)	U(.)	SETF	NR. OF REF. NUMBERS	REMARKS
BASIC ROUTINES	ERROR IND ERROR STOP EXIT CHECK PRINT PRINT VAR		195	0	11	0	-	5	ALWAYS IN A-PROGRAM
'COPY' (p, p_1:m, n)	(p:m, n)=(p_1:m, n)		17	1	5	0	-	1	
'TRANSPOSE'(p, p_1:m, n)	(p:n, m)=(p_1:m, n)'		26	1	5	0	-	1	
'ADD' (p, p_1, p_2:m, n)	(p:m, n)=(p_1:m, n)+(p_2:m, n)	X	28	2	6	0	-	1	ERROR IND if result partly overwrites operands
'SUB' (p, p_1, p_2:m, n)	(p:m, n)=(p_1:m, n)-(p_2:m, n)	X	28	2	6	0	-	1	
'MULT' (p, p_1, p_2:m, n, s)	(p:m, s)=(p_1:m, n)*(p_2:n, s)	X	46	2	8	0	-	1	
'MULT 01' (p, p_1, p_2:m, n, s)	(p:m, s)=(p_1:m, n)*(p_2:s, n)'	X							ERROR IND if result overwrites operands
'MULT 10' (p, p_1, p_2:m, n, s)	(p:n, s)=(p_1:m, n)'*(p_2:m, s)	X	64	2		0	-	3	
'MULT 11' (p, p_1, p_2:m, n, s)	(p:n, s)-(p_1:m, n)'*(p_2:s, m)'	X							
'INVERT' (p:n, n)	(p:n, n)=(p:n, n)$^{-1}$	X	90	0	10	2	-	2	ERROR No. - 100m+i, if i'th row is linearly dependent on the previous rows (m=rej number)
'INV DET'(p:n, n:e, c)	(p:n, n)=(p:n, n)$^{-1}$	X			+n				

The normal Gauss-Jordan elimination with maximum pivot is used

e=lowest bound of pivot

side result: U1=c*det(p:n, n)

MATRIX INSTRUCTION	OPERATION	NR. OF LOCATIONS	P(.)	V(.)	U(.)	SETF	NR. OF REF. NUMBERS	REMARKS
'READ J1' (p:m, n:e)	Read (p:m, n) in J*code sum tolerance=e	57	0	5	2	-	2	FOR ALL INPUT ROUTINES:
'READ J2' (p:m, n)	Read (p:m, n) in J*code sum tolerance set by READ J1 (initially 0.0001)							When a sum error is detected
'READ T1' (p:n, n:e)	Read (p:n, n) in J*Δ code sum tolerance=e	74	0	7	2	-	2	% j d is punched, where j=no. of column d=difference between the read sum and the punched sum
'READ T2' (p:n, n)	Read (p:n, n) in J*Δ code sum tol. set by READ T1 (initially 0.0001)							
'READ S1' (p:m, n:e)	Read (p:m, n) in *code sum tolerance=e	44	0	5	2	-	2	if the sum error can be neglected change the sign of N.G. word
'READ S2' (p:m, n)	Read (p:m, n) in *code sum tol. set by READ S1 (initially 0.0001)							

MATRIX INSTRUCTION	OPERATION	NR. OF LOCATIONS	P(.)	V(.)	U(.)	SETF	NR. OF REF. NUMBERS	REMARKS
'OUT J1' (p:m, n, q_1, q_2)	Print (p:m, n) in J*code print parameters q_1, q_2 meaning that if $q_1>0$ use PRINT Z, q_1:q_2 if $q_1=0$ use PRINT Z, q_2							
'OUT J2' (p:m, n)	Print (p:m, n) in J*code print parameters set by OUT J1 or T1 (initially 5, 5)	111	0	13	1	INT	4	
'OUT T1' (p:n, n, q_1, q_2)	Print (p:n, n) in J*Δ code print parameters q_1, q_2 as in OUT J1							ERROR IND if print parameter is incorrect
'OUT T2' (p:n, n)	Print (p:n, n) in J*Δcode print parameters set by OUT J1 or T1 (initially 5, 5)							
'OUT M1' (p:m, n, q_1, q_2, k)	Print (p:m, n) in matrix form, k numbers on one line print parameters q_1, q_2 meaning that if $q_1>0$ use PRINT Z, q_1:q_2 if $q_1=0$ use PRINT Z, q_2 if $q_1<0$ use PRINT Z, q_2 /	84	0	11	0	-	2	
'OUT M2' (p:m, n)	Print (p:m, n) in matrix form layout set by OUT M1							

MATRIX INSTRUCTION	OPERATION		NR. OF LOCATIONS	P(.)	V(.)	U(.)	SETF	NR. OF REF. NUMBERS	REMARKS
'DELETE' (p:m, n, i, j)	i'th row and j'th column are deleted from (p:m, n)	X	26	0	8	0	-	1	if i=0, only j'th column is deleted
'JOIN' (p, p_1, p_2:m, n, s)	matrices (p_1:m, n) and (p_2:s, n) are joined to form (p:m+s, n)	X	20	2	9	0	-	1	
'DISJOIN' (p, p_1p_2:m, n, s)	inverse operation of JOIN	X	20	2	9	0	-	1	
'CLEAR' (p:m, n)	(p:m, n)=0		9	0	5	0	-	1	
'UNIT' (p:n, n)	(p:n, n)=I		14	0	5	0	-	1	
'DIAGONAL' (p, p_1:n, n)	(p:n diag)=diagonal of (p_1:n, n)	X	13	1	6	0	=	1	
'DIAG ADD' (p, p_1, p_2:n, n)	(p:n, n)=(p_1:n diag) +(p_2:n, n)	X	19	2	7	0	-	1	
'DIAG MULT' (p, p_1, p_2:m, n)	(p:m, n)=(p_1:m diag) *(p_2: m, n)	X	18	2	7	0	-	1	
'MULT DIAG' (p, p_1, p_2:m, n)	(p:m, n)=(p_1:m, n) *(p_2:n diag)	X	15	2	7	0	-	1	

MATRIX INSTRUCTION	OPERATION		NR. OF LOCATIONS	P(.)	V(.)	U(.)	SETF	NR. OF REF. NUMBERS	REMARKS
'ROW' (p, p_1:m, n, i)	(p:n, 1)=i'th row of (p_1:m, n)	X	13	1	6	0	-	1	
'COLUMN' (p, p_1:m, n, j)	(p:m, 1)=j'th column of (p_1:m, n)	X	13	1	6	0	-	1	
'NORM C' (p:m, n)	Normalize (p:m, n) by columns (sums of squares = 1)	X	42	0	8	0	SQRT	2	ERROR IND if column in NORM C or row in NORM R = 0
'NORM R' (p:m, n)	Normalize (p:m, n) by rows (sums of squares = 1)	X							
'SCALP C'(p, p_1, p_2:m, n)	(p:n, 1)=scalar products of columns of (p_1:m, n) and (p_2:m, n)	X	25	2	7	0	-	1	
'SCALP R' (p, p_1, p_2:m, n)	(p:m, 1)=scalar products of rows of (p_1:m, n) and (p_2:m, n)	X	26	2	7	0	-	1	
'ELEMENT' (p:m, n, i, j)	U=a_{ij} V3=i V4=j		7	0	5	0	-	1	a_{ij} = element of i'th row and j'th column of (p:m, n)
'PUT' (p:m, n, i, j, :c)	a_{ij}=c		7	0	5	0	-	1	
'SCAL ADD' (p:m, n:c)	(p:m, n) = $\lceil c+a_{ij} \rceil$		11	0	5	0	-	1	
'SCAL MULT' (p:m, n:c)	(p:m, n) = $\lceil ca_{ij} \rceil$		11	0	5	0	-	1	
'TRACE' (p:n, n)	U=trace (p:n, n)		12	0	5	0	-	1	

MATRIX INSTRUCTION	OPERATION	NR. OF LOCATIONS	P(.)	V(.)	U(.)	SETF	NR. OF REF. NUMBERS	REMARKS
'SUM' (p:m, n, c)	U=c+sum of elements of (p:m, n)	10	0	5	0	-	1	a_{ij} = element of i'th row and j'th column of (p:m, n
'PRODUCT' (p:m, n:c)	U=c*product of elements of (p:m, n)	10	0	5	0	-	1	
'MAX' (p:m, n)	U=max a_{ij} V3=i V4=j	16	0	7	0	-	1	
'MIN' (p:m, n)	U=min a_{ij} V3=i V4=j	16	0	7	0	-	1	
'MAX MOD' (p:m, n)	U=max $\|a_{ij}\|$ V3=i V4=j	19	0	7	1	-	1	
'MIN MOD' (p:m, n)	U=min $\|a_{ij}\|$ V3=i V4=j	20	0	7	1	-	1	