User's guide to fritz laboratory matrix package (flmxpk), September 1970

Sampath N. S. Iyengar
Celal N. Kostem
USER'S GUIDE

TO

FRITZ LABORATORY MATRIX PACKAGE
(FLMXPK)

by

Sampath N. S. Iyengar
Celal N. Kostem

Department of Civil Engineering
Fritz Engineering Laboratory
Lehigh University
Bethlehem, Pennsylvania

September 1970

Fritz Engineering Laboratory Report No. 400.1
# TABLE OF CONTENTS

1. INTRODUCTION · 1
2. GENERAL LIMITATIONS · 1
3. DISCLAIMER · 3
4. DECK SETUP · 3
5. DESCRIPTION OF THE SUBPROGRAMS · 4
   5.1 SUBROUTINE ADD(A,B,C,M,N) · 5
   5.2 SUBROUTINE DETMT(A,DA,N) · 6
   5.3 SUBROUTINE DIAG(A,DA,N) · 6
   5.4 SUBROUTINE EV(A,S,N) · 7
   5.5 SUBROUTINE GEVP(A,B,S,T,N) · 7
   5.6 SUBROUTINE IEV(A,S,N) · 8
   5.7 SUBROUTINE MINV(A,N,DET,NEXCH) · 10
   5.8 SUBROUTINE MOVE(A,B,M,N) · 11
   5.9 SUBROUTINE MULT(A,B,C,L,M,N) · 11
   5.10 SUBROUTINE OUTE(A,I,J,TITLE,TITEL) · 12
   5.11 SUBROUTINE OUTF(A,I,J,TITLE,TITEL) · 14
   5.12 SUBROUTINE OUTG(A,I,J,TITLE,TITEL) · 14
   5.13 SUBROUTINE PMULT(A,B,K,L,X) · 15
   5.14 SUBROUTINE POSTM(A,B,K,L,X) · 16
   5.15 SUBROUTINE RDCBC(A,M,N) · 17
   5.16 SUBROUTINE RDCOLG(A,N,N) · 18
   5.17 SUBROUTINE RDRBR(A,N,N) · 19
   5.18 SUBROUTINE RDROWG(A,M,N) · 19
   5.19 SUBROUTINE SCMUL(A,M,N,X) · 20
   5.20 SUBROUTINE SINV(A,DA,N) · 21
   5.21 SUBROUTINE SOLVE(C,B,A,X,N,M,L,DET,IFLAG) · 22
   5.22 SUBROUTINE SQTR(A,N) · 23
   5.23 SUBROUTINE SUB(A,B,C,M,N) · 24
   5.24 SUBROUTINE TMULT(A,B,C,L,M,N) · 24
   5.25 SUBROUTINE TRANS(A,B,M,N) · 25
6. READY REFERENCE SHEET · 26
7. ACKNOWLEDGMENTS · 28
8. REFERENCE · 28
1. **INTRODUCTION**

The subprograms in this package were developed by Mr. Sampath Iyengar of the Computer Systems Group in Fritz Engineering Laboratory, for use by members of the laboratory. Dr. C. N. Kostem is the Chairman of the Computer Systems Group.

This version supersedes the earlier one (FCMXPK - Fortran Callable Matrix Package) which will be withdrawn by a date to be specified. Mr. Edward T. Manning, Jr. was associated with Mr. Iyengar in the development of FCMXPK.

In this Guide, only a brief description of the function, limitations and requirements of each subprogram is included. More detailed information on how the subprograms were developed will be available with the publication of Reference 1.

The package includes routines for matrix manipulation as under:

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>SUBPROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add matrices</td>
<td>ADD</td>
</tr>
<tr>
<td>Subtract matrices</td>
<td>SUB</td>
</tr>
<tr>
<td>Determinant of matrix</td>
<td>DETMT, MINV, SINV, SOLVE</td>
</tr>
<tr>
<td>Create Diagonal matrix</td>
<td>DIAG</td>
</tr>
<tr>
<td>Eigenvalues of symmetric matrices</td>
<td>EV, IEV, GEVP</td>
</tr>
<tr>
<td>Invert matrix</td>
<td>MINV, SINV</td>
</tr>
<tr>
<td>Copy matrix</td>
<td>MOVE</td>
</tr>
<tr>
<td>Multiply matrices</td>
<td>MULT, PMULT, POSTM, SCMUL, TMULT</td>
</tr>
<tr>
<td>Print matrix</td>
<td>OUTE, OUTF, OUTG</td>
</tr>
<tr>
<td>Read matrix</td>
<td>RDCBC, RDCOLG, RDRBR, RDROWG</td>
</tr>
<tr>
<td>Simultaneous equations</td>
<td>SOLVE</td>
</tr>
<tr>
<td>Transpose matrix</td>
<td>SQTR, TRANS</td>
</tr>
</tbody>
</table>

2. **GENERAL LIMITATIONS**

The subprograms do not provide any diagnostics if operations which are not possible mathematically, such as inversion of a singular matrix,
are attempted. The requirements of the subprograms are not tested prior to or during execution and hence, when the requirements are violated, the answers obtained are, in general, unpredictable and wrong.

Only the following subprograms provide printouts:

i) SUBROUTINE OUTE (A,I,J,TITLE,TITEL)
ii) SUBROUTINE OUTF (A,I,J,TITLE,TITEL)
iii) SUBROUTINE OUTG (A,I,J,TITLE,TITEL)
iv) SUBROUTINE SOLVE (C,B,A,X,N,M,L,DET,IFLAG), only if IFLAG has been set to zero.

SUBROUTINE RDCBC (A,M,N), SUBROUTINE RDCOLG (A,M,N), SUBROUTINE RDRBR (A,M,N) and SUBROUTINE RDROWG (A,M,N) enable reading in values from data cards for the matrix A. All others are 'calculation' subprograms.

In all the subprograms, 'variable' or 'adjustable' dimensions are used for the several arrays. The user must, therefore, prescribe exact dimensions for all his arrays to be handled by this package. Over-dimensioning, except under special circumstances of usage [1], may lead to wrong results. Under-dimensioning invariably produces wrong results or aborts the execution of the program.

For Input/Output operations, card input and printer output are assumed.

The limitations on the sizes of the matrices that can be handled are not due to any feature in programming involved in this package but due to the capacity of the machine used.

The label IYENGAR may not be used in the user's program for any of his labelled COMMON blocks. Similarly, the names of routines in this package may not be used in the user's program for his program or any of the subprograms.
3. **DISCLAIMER**

The burden of proof on the validity and applicability of this package to a particular problem rests with the user, and not the authors. No guarantee is stated or implied that the package will give correct results, or that the mathematical relations and assumptions used are proper and applicable to the problem under consideration by the user. The authors cannot be held responsible for incorrect results or damages resulting from the use of the package, although it is believed that the package is correctly formulated.

The authors welcome suggestions for improvements and notice of any errors. If a correction is possible and implemented, proper publicity will be given to the revised status of the package. Else, the concerned subprogram will be withdrawn.

4. **DECK SETUP**

This package is expected to be available as a permanent file at Lehigh University Computing Center, in due course. At the present time, a prospective user who belongs to Fritz Laboratory may borrow a binary deck and make his own copy. Interested users please contact Mr. Sampath Iyengar or Mr. Karl H. Frank.

A sample deck setup, using the binary deck, is as follows:

Job Card ...
RUN(S)
LOAD(INPUT)
LGO.
7/8/9
User's FORTRAN Program with Subprograms, if any
7/8/9
Binary Deck (FLMXPK)
7/8/9
Data, if any
6/7/8/9
5. DESCRIPTION OF THE SUBPROGRAMS

From the user's viewpoint, there are, in all, 25 routines in this package. These will be described in alphabetical order under the following general headings:

a. Function:

b. Calling Program:

1) Dimensions: Those that are required in the calling program. If the calling program is the main program, the dimensions must be stated in terms of absolute numbers, such as DIMENSION A (10, 15). If it is a subprogram, the dimension statement is either of the same form or of the form DIMENSION A (M,N) where M and N have been initialized through previous operations. (The user is obliged, in the latter instance, to include A, M and N in the argument list of his subprogram).

Most of the arrays used in this package are two-dimensional arrays. Only a few are vectors or one-dimensional arrays. These vectors are needed only for the purpose of storing some intermediate values in some subprograms.

All the subscripted variables handled by this package are 'real' variables. [The solitary exception, vector NEXCH in SUBROUTINE MINV (A,N,DET,NEXCH), needs no special consideration by the user]. Mistakes often occur when this fact is overlooked and the user prescribes an 'integer' name for what is clearly an array of 'real' variables. An example from civil engineering is to refer to the stiffness matrix
as K without a corresponding TYPE statement. An easy solution is to dimension the arrays in the TYPE statement such as REAL K (20,20), TEMP (20). An advantage here is that the user has a free choice of names. Further, a separate DIMENSION statement need not be (in fact, should not be) provided. Inclusion in the TYPE statement of names of arrays which are 'real' even without a TYPE statement, like TEMP in the example, does not hurt in any way.

ii) Definitions: Arrays and variables that must be defined prior to or in the CALL statement.

iii) Values Returned to the Calling Program:

   c. Limitations (if any):

      The general limitations mentioned in Chapter 2 will not be repeated.

   c. or d. Additional Notes (if any):

   c. or d. or e. Examples of CALL Statement:

5.1. SUBROUTINE ADD (A,B,C,M,N)

a. Function:

   Add matrices A and B and store the sum in matrix C. (C = A + B).

b. Calling Program:

   i) Each matrix is of size M rows by N columns.

   ii) Matrices A and B as well as integers M and N must be defined.

   iii) Matrix C is defined in the subprogram.

c. Additional Notes:

   The resultant matrix C may be stored in one of the original matrices. Only in such a case, the specific original matrix will be destroyed.
d. **Examples:**

i) `CALL ADD (A,B,C,15,20)`

ii) `CALL ADD (ABLE, BAKER, CHARLIE, NR, NC)`

iii) `CALL ADD (A,B,A,10,15)`

5.2. **SUBROUTINE DETMT (A,DA,N)**

a. **Function:**

The determinant of the given (square) matrix A is made available to the calling program as DA. \( DA = \text{det}(A) \).

b. **Calling Program:**

i) Matrix A is of size N rows by N columns.

ii) Matrix A and integer N should be defined.

iii) DA is defined in the subprogram.

c. **Limitations:**

The original matrix A is destroyed.

d. **Examples:**

i) `CALL DETMT (A,DA,N)`

ii) `CALL DETMT (ARRAY, DET, 20)`

5.3. **SUBROUTINE DIAG (A,DA,N)**

a. **Function:**

A diagonal matrix A is generated as follows:

Each diagonal element has the value DA, and each off-diagonal element has the value zero.

b. **Calling Program:**

i) Matrix A is of size N rows by N columns.

ii) The value of the diagonal element DA and integer N must be defined.

iii) Matrix A is defined in the subprogram.
c. **Examples:**
  
i) CALL DIAG (A,1.0,15)
  
ii) CALL DIAG (ARRAY,DE,NSIZE)

### 5.4. SUBROUTINE EV (A,S,N)

**a. Function:**

Eigenvalues and eigenvectors of the symmetric matrix A are computed.

**b. Calling Program:**

i) Matrices A and S are of size N rows by N columns.

ii) Matrix A and integer N must be defined.

iii) On return to the calling program, matrix A has, for its diagonal elements, the eigenvalues of the original matrix A and matrix S has, for its columns, the corresponding eigenvectors.

**c. Limitations:**

The original matrix A must be symmetric. It will be destroyed in the subprogram, as the eigenvalues are returned in the same matrix.

**d. Additional Notes:**

The eigenvalues and eigenvectors may be improved further, if so desired, by using SUBROUTINE IEV (A,S,N).

**e. Examples:**

i) CALL EV (A,S,N)

ii) CALL EV (ARRAY,EVEC,10)

### 5.5. SUBROUTINE GEVP (A,B,S,T,N)

**a. Function:**

Eigenvalues and eigenvectors of the given matrix A where

\[ [A] \{X\} = \lambda[B]\{X\} \]

are computed. Both the matrices A and B are symmetric, and further matrix B is also positive-definite.
b. **Calling Program:**

   i) Matrices \( A, B \) and \( S \) are of size \( N \) rows by \( N \) columns. \( T \) is a vector of size \( N \) elements.

   ii) Matrices \( A \) and \( B \), as well as integer \( N \), should be defined.

   iii) The eigenvalues are returned as the diagonal elements of matrix \( A \) and the corresponding eigenvectors as the columns of matrix \( B \). Matrix \( S \) and vector \( T \) are used for storing some intermediate values in computations.

c. **Limitations:**

Matrices \( A \) and \( B \) must be symmetric. Matrix \( B \) must also be positive-definite. Both the original matrices \( A \) and \( B \) are destroyed in the subprogram.

The following subprograms of this package must be available and loaded when this subprogram is used:

   i) SUBROUTINE EV \((A,S,N)\)

   ii) SUBROUTINE MULT \((A,B,C,L,M,N)\)

   iii) SUBROUTINE POSTM \((A,B,K,L,X)\)

   iv) SUBROUTINE TMULT \((A,B,C,L,M,N)\)

d. **Examples:**

   i) CALL GEVP \((A,B,S,T,N)\)

   ii) CALL GEVP \((EVAL,EVEC,S,TEMP,10)\)

5.6. **SUBROUTINE IFV \((A,S,N)\)**

a. **Function:**

   Eigenvalues and eigenvectors of the symmetric matrix \( A \) computed by the use of SUBROUTINE EV \((A,S,N)\) are improved.

b. **Calling Program:**

   Same as in SUBROUTINE EV \((A,S,N)\).
c. **Limitations:**

Same as in SUBROUTINE EV \((A, S, N)\).

d. **Additional Notes:**

The accuracy of calculations in SUBROUTINE EV \((A, S, N)\) is prescribed according to the following scheme. The square root of the sum of the squares of the off-diagonal elements (of the original matrix \(A\)) is computed first. This is called the initial threshold. A final threshold value of one-millionth of such sum is then established. The diagonalization, which is an iterative process, proceeds up to the stage when the absolute value of every off-diagonal element is less than or equal to the final threshold value.

Since the process is iterative, the user has the option to improve the accuracy of the results by successive CALL-s to the subprogram. For reasons explained in the documentation [1], these successive CALL-s must be to SUBROUTINE IEV \((A, S, N)\). The following further rules apply.

1) SUBROUTINE EV \((A, S, N)\) must be CALL-ed once only and before the SUBROUTINE IEV \((A, S, N)\) is CALL-ed.

2) SUBROUTINE IEV \((A, S, N)\) may be CALL-ed subsequently the required number of times to achieve the desired accuracy. If a total number of \(n\) CALL-s are made to (both) the subprograms, each off-diagonal element will be reduced in absolute value to less than \(10^{-6n}\) times the initial threshold.

3) Neither matrix \(A\) nor matrix \(S\) may be altered in the calling program between any two of the above CALL-s to the subprogram.
Trial runs have indicated that the improvement procedure causes small but significant changes in the eigenvectors, and practically no changes in the eigenvalues (apparently because these are already very close to the exact values). An excessive number of improvement cycles may result in an underflow in the machine.

e. Example:

\begin{verbatim}
CALL EV (A,S,N)
\.
CALL IEV (A,S,N)
\.
CALL IEV (A,S,N)
\end{verbatim}

5.7. SUBROUTINE MINV (A,N,DET,NEXCH)

a. Function:
The matrix A is inverted in its own space and its determinant is computed.

b. Calling Program:
i) Matrix A is of size N rows by N columns. Vector NEXCH is of size N elements (see item d below).
ii) Matrix A and integer N should be defined.
iii) The inverse of the original matrix A is returned in A itself.

The value of the determinant of the original matrix A is returned in DET.

c. Limitations:
The original matrix A is destroyed in the subprogram. See also 'Additional Notes' under SUBROUTINE SINV (A,DA,N).

d. Additional Notes:
The vector NEXCH is used for computations only in the subprogram
and the values of its elements are of no consequence to the calling program. Hence, the matching vector in the calling program need not necessarily be a vector of integer elements.

e. **Examples:**

i) CALL MINV (A,N,DET,NEXCH)

ii) CALL MINV (ARRAY,10,DET,TEMP)

5.8. **SUBROUTINE MOVE (A,B,M,N)**

a. **Function:**

Matrix A is copied as matrix B. (B = A)

b. **Calling Program:**

i) Matrices A and B are of size M rows by N columns.

ii) Matrix A and integers M and N should be defined.

iii) Matrix B is defined in the subprogram.

c. **Additional Notes:**

When certain subprograms such as SUBROUTINE DETMT (A,DA,N) of this package are CALL-ed, the original matrices get destroyed in the subprograms. The user may have a need to store the original matrices for further use at a later time. This subprogram meets such a need.

d. **Examples:**

i) CALL MOVE (ARRAY,SAME,10,15)

ii) CALL MOVE (A,B,M,N)

5.9. **SUBROUTINE MULT (A,B,C,L,M,N)**

a. **Function:**

Matrix A is post-multiplied by matrix B to yield matrix C. (C = AB)
b. **Calling Program:**

i) Matrices A, B and C have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>L rows by M columns</td>
</tr>
<tr>
<td>B</td>
<td>M rows by N columns</td>
</tr>
<tr>
<td>C</td>
<td>L rows by N columns</td>
</tr>
</tbody>
</table>

ii) Matrices A, B and integers L, M and N should be defined.

iii) The product matrix C will be defined in the subprogram.

c. **Limitations:**

Matrix C should be distinct from matrices A and B. However, matrices A and B may be identical. See also SUBROUTINE PMULT (A,B,K,L,X) and SUBROUTINE POSTM (A,B,K,L,X).

d. **Examples:**

i) CALL MULT (A,B,C,L,M,N)

ii) CALL MULT (A,A,C,N,N,N)

The two examples below yield wrong results:

iii) CALL MULT (A,B,A,L,M,M)

iv) CALL MULT (A,B,B,L,L,N)

5.10. **SUBROUTINE OUTE (A,I,J,TITLE,TITEL)**

a. **Function:**

Matrix A of size I rows by J columns is printed. Printing begins on a new page. The matrix is labelled at the top of each page with the labels provided by the user. The word CONTINUED in parentheses appears against the label if printing is on more than one page. The size of the matrix is indicated below the label.
Rows and columns are numbered. On any one page, the maximum number of rows printed is 25, and the maximum number of columns is 10. Hence if \( J \leq 10 \) and \( I \leq 25 \), printing is completed on one page. If \( J > 10 \), the first 10 columns are printed, until all the rows (25 or less to a page) are exhausted. Then the second 10 columns (or less) are printed, until all the rows are exhausted, and so on.

The elements of matrix \( A \) are output in E-FORMAT. (See SUBROUTINE OUTF \((A,I,J,TITLE,TITEL)\) for F-FORMAT output, and SUBROUTINE OUTG \((A,I,J,TITLE,TITEL)\) for G-FORMAT output). Five digits appear to the right of the decimal point (E 12.5).

b. Calling Program:

i) Matrix \( A \) is of size \( I \) rows by \( J \) columns.

ii) Matrix \( A \) and integers \( I \) and \( J \) should be defined. Also, the user's label must be provided as a Hollerith string of characters (maximum 20) through alphanumeric variables or 'values' corresponding to the arguments TITLE and TITEL. See examples of CALL statement.

iii) No formal 'values' are returned by this subprogram.

c. Additional Notes:

This subprogram is recommended for use in preference to SUBROUTINE OUTF \((A,I,J,TITLE,TITEL)\) whenever the magnitudes of the elements of the matrix to be printed are unknown, unpredictable, or exceed the field F12.5. A slight sacrifice of easy readability is implicit.
5.11. SUBROUTINE OUTF(A,I,J,TITLE,TITEL)

a. Function:
All the details are the same as in SUBROUTINE OUTE(A,I,J,TITLE,TITEL) except that the elements are output in F-FORMAT. Five digits appear to the right of the decimal point, and a maximum of six digits (five, if the value is negative) appear to its left. (F12.5)

b. Calling Program:
Same as in SUBROUTINE OUTE(A,I,J,TITLE,TITEL)

c. Limitations:
The 'largest' numbers that can be printed are of the form abcdef.ghijk or -bcdef.ghijk. If a number 'larger' than these is attempted to be printed, an asterisk (*) will appear at the beginning of the corresponding field.

d. Example:
CALL OUTF(A,I,J,8HMATRIX A,1H)

5.12. SUBROUTINE OUTG(A,I,J,TITLE,TITEL)

a. Function:
All the details are the same as in SUBROUTINE OUTE(A,I,J,TITLE,TITEL) except that the elements are output in G-FORMAT.
b. **Calling Program:**

Same as in SUBROUTINE OUTE(A,I,J,TITLE,TITEL).

c. **Additional Notes:**

Five significant digits appear on output, if the absolute value \( x \) of the element being printed is in the range

\[
0.1 \leq x < 10^5 \quad (G \ 12.5)
\]

Otherwise, the output is in E-FORMAT for the element.

d. **Example:**

CALL OUTG(A,I,J,lH ,lH )

5.13. **SUBROUTINE PMULT(A,B,K,L,X)**

a. **Function:**

The square matrix \( B \) is premultiplied by a rectangular (or square) matrix \( A \) and the product matrix is stored in \( A \).

\[ A = AB. \]

b. **Calling Program:**

i) Matrices \( A \) and \( B \), and vector \( X \), have the following dimensions:

\[
\begin{array}{ll}
\text{Matrix} & \text{Size} \\
A & K \text{ rows by } L \text{ columns} \\
& \text{(may be square, } K = L) \\
B & L \text{ rows by } L \text{ columns} \\
X \text{ (Vector)} & L \text{ elements}
\end{array}
\]

ii) Matrices \( A, B \) and integers \( K, L \) should be defined.

iii) The product matrix is returned in matrix \( A \). Vector \( X \) is required in the subprogram for computations only.

c. **Limitations:**

The original matrix \( A \) is destroyed.
d. Examples:
   i) CALL PMULT(A,B,K,L,X)
   ii) CALL PMULT(A,B,K,K,X)
       The example below yields wrong results:
   iii) CALL PMULT(A,A,L,L,X)

5.14. SUBROUTINE POSTM(A,B,K,L,X)

a. Function:
   The square matrix A is postmultiplied by a rectangular
   (or square) matrix B and the product matrix is stored in B.
   \[ B = AB \]

b. Calling Program:
   i) Matrices A and B, and vector X, have the following
      dimensions:
      \[
      \begin{array}{ccc}
      \text{Matrix} & \text{Size} \\
      A & K \text{ rows by } K \text{ columns} \\
      B & K \text{ rows by } L \text{ columns} \\
      & (\text{may be square, } K = L) \\
      X \ (\text{Vector}) & K \text{ elements} \\
      \end{array}
      \]
   ii) Matrices A, B and integers K, L should be defined.
   iii) The product matrix is returned in matrix B. Vector X
       is required in the subprogram for computations only.

c. Limitations:
   The original matrix B is destroyed.

d. Examples:
   i) CALL POSTM(A,B,K,L,X)
   ii) CALL POSTM(A,B,L,L,X)
       The example below yields wrong results:
   iii) CALL POSTM(A,A,K,K,X)
5.15. **SUBROUTINE RDCBC(A,M,N)**

a. **Function:**

Elements of matrix $A$ are defined (column by column) by reading in values from data cards.

b. **Calling Program:**

i) Matrix $A$ is of size $M$ rows by $N$ columns.

ii) Integers $M$ and $N$ should be defined.

The number of data cards required per column of matrix $A$ is $M/8$ if this division leaves the remainder zero, or $M/8$ rounded to the next integer, otherwise.

iii) Matrix $A$ is defined in the subprogram.

c. **Additional Notes:**

i) **FORMAT Control:**

The FORMAT is $(8F10.0)$ and the decimal point should preferably be punched in each data field.

If it is not punched, it will be assumed to be at the end of the field. The non-punch positions in the field are assumed to be filled with zeroes.

For example, if 23.5 is punched beginning in column 21, the value assigned to the corresponding element is the same. If the decimal point is not punched, the value assigned is 2305000000.0.

ii) **Order in Assigning Values:**

Assume matrix $A$ is of size 14 rows by 6 columns. Two data cards are required per column. Hence, the total number of data cards required is 12. The eight values on the first data card will be assigned in order to $A_{1,1}, A_{2,1}, \ldots, A_{8,1}$. The six values on the second data
d. Example:
CALL RDCBC(A,14,6)

5.16. SUBROUTINE RDCOLG(A,M,N)

a. Function:
Elements of matrix A are defined by reading in values from data cards. The elements are assumed to be in a continuous string of columns of matrix A.

b. Calling Program:
i) Matrix A is of size M rows by N columns.

ii) Integers M and N should be defined.
The number of data cards required is \((M\times N)/8\) if this division leaves the remainder zero, or \((M\times N)/8\) rounded to the next integer, otherwise.

iii) Matrix A is defined in the subprogram.

c. Additional Notes:
i) Same as in SUBROUTINE RDCBC(A,M,N).

ii) Order in Assigning Values:
Assume matrix A is of size 14 rows by 6 columns. Eleven data cards are required. The eight values on the first data card will be assigned in order to \(A_{1,1}, A_{2,1}, \ldots, A_{8,1}\). The first six values on the second card to \(A_{9,1}, A_{10,1}, \ldots, A_{14,1}\). The last two values on the second card to \(A_{1,2}, A_{2,2}\). The eight values on the third card to \(A_{3,2}, A_{4,2}, \ldots, A_{10,2}\). And so on.

Of the four READ subprograms in this package, this one requires minimum execution time.
d. **Example:**

CALL RDCOLG(A,14,6)

5.17. **SUBROUTINE RDRBR(A,M,N)**

a. **Function:**

Elements of matrix A are defined (row by row) by reading in values from data cards.

b. **Calling Program:**

i) Matrix A is of size M rows by N columns.

ii) Integers M and N should be defined.

The number of data cards required per row of matrix A is N/8 if this division leaves the remainder zero, or N/8 rounded to the next integer, otherwise.

iii) Matrix A is defined in the subprogram.

c. **Additional Notes:**

i) Same as in SUBROUTINE RDCBC(A,M,N)

ii) **Order in Assigning Values:**

Assume matrix A is of size 14 rows by 6 columns. One data card is required per row. Hence, the total number of data cards required is 14. The six values on card number I (I ranges in value from 1 to 14) are assigned in order to $A_{1,1}, A_{1,2}, \ldots, A_{1,6}$.

d. **Example:**

CALL RDRBR(A,14,6)

5.18. **SUBROUTINE RDROWG(A,M,N)**

a. **Function:**

Elements of matrix A are defined by reading in values from
data cards. The elements are assumed to be in a continuous string of rows of matrix A.

b. Calling Program:
Same as in SUBROUTINE RDCOLG(A,M,N)

c. Additional Notes:
   i) Same as in SUBROUTINE RDCBC(A,M,N)
   ii) Order in Assigning Values:
       Assume matrix A is of size 14 rows by 6 columns. Eleven data cards are required. The first six values on the first data card will be assigned in order to $A_{1,1}$, $A_{1,2}$, ..., $A_{1,6}$. The last two values on the first card to $A_{2,1}$, $A_{2,2}$. The first four values on the second card to $A_{2,3}$, $A_{2,4}$, $A_{2,5}$, $A_{2,6}$. The last four values on the second card to $A_{3,1}$, $A_{3,2}$, $A_{3,3}$, $A_{3,4}$. And so on.

d. Example:
CALL RDROWG(A,14,6)

5.19. SUBROUTINE SCMUL(A,M,N,X)

a. Function:
Elements of matrix A are multiplied by the scalar quantity X.

b. Calling Program:
   i) Matrix A is of size M rows by N columns.
   ii) Matrix A and the scalar multiplier X as well as integers M and N should be defined.
   iii) The modified matrix is returned in A itself.

c. Limitations:
The original matrix is destroyed.
d. Example:
   i) \( \text{REI} = 1.0 / \text{EI} \)
      
      \[
      \text{CALL SCMUL}(A,M,N,\text{REI})
      \]
   ii) \( \text{CALL SCMUL}(A,M,N,1.0/30000.0) \)

5.20. SUBROUTINE SINV(A,DA,N)

a. Function:
   The symmetric matrix A (also positive-definite) is inverted in its own space and its determinant is computed.

b. Calling Program:
   i) Matrix A is of size N rows by N columns.
   ii) Matrix A and integer N should be defined.
   iii) The inverse is returned in matrix A itself. DA stores the value of the determinant of the original matrix A.

c. Limitations:
   The original matrix A must be symmetric as well as positive-definite. The original matrix is destroyed.

d. Additional Notes:
   The inverse of a symmetric matrix is also symmetric. This property has been utilized in this subprogram, and hence the resulting inverse of a symmetric matrix will be symmetric when this subprogram is used.
   It is possible that the inverse of a symmetric matrix obtained by the use of SUBROUTINE SINV(A,N,DET,NEXCH) is not ideally symmetric because of round-off errors in the machine.

e. Example:
   \[
   \text{CALL SINV}(A,DA,N)
   \]
5.21. **SUBROUTINE SOLVE(C,B,A,X,N,M,L,DET,IFLAG)**

a. **Function:**
   A system of linear simultaneous equations \( CX = B \) is solved.

b. **Calling Program:**
   i) Matrices \( C, B, A \) and \( X \) have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient Matrix ( C )</td>
<td>N rows by N columns</td>
</tr>
<tr>
<td>Right Hand Side Matrix ( B )</td>
<td>N rows by L columns</td>
</tr>
<tr>
<td>Augmented Matrix ( A )</td>
<td>N rows by M columns (M = N + L)</td>
</tr>
<tr>
<td>Solution Matrix ( X )</td>
<td>N rows by L columns</td>
</tr>
</tbody>
</table>

   ii) Matrices \( C \) and \( B \) as well as integers \( N, M \) and \( L \) should be defined. Note that the equation \( M = N + L \) must be satisfied. See 'Additional Notes' regarding \( IFLAG \).

   iii) The solution matrix is returned in \( X \). Note that there is a one-to-one correspondence between columns of matrices \( B \) and \( X \). The value of the determinant of the coefficient matrix \( C \) is returned in \( DET \).

c. **Additional Notes:**
   None of the input (to the subprogram) quantities is affected. Matrix \( A \) is required in the subprogram for computations only. The subprogram produces its own output if \( IFLAG \) has been set to zero. Such output has the following features:
   i) Printing begins on a new page.
   ii) The determinant \( DET \) is labelled and printed in G-FORMAT.
   iii) There are assumed to be as many 'problems' as there are number of columns of \( B \) (and \( X \)). For the \( K \)th problem, \( K \)th
column of $X$ is the 'Solution Vector' and the $K$th column of $B$ is the 'Right Hand Side'. The product of matrix $C$ and the $K$th column of matrix $X$ is the 'Generated Right Hand Side'. The column vector ERROR is the vector difference (RHS-Generated RHS).

For each problem $K$, where $K$ ranges from 1 to $L$, the column vectors, 1) Solution Vector, 2) RHS, 3) Generated RHS and 4) ERROR are tabulated.

iv) The output is terminated with a message.

If IFLAG has been set to a non-zero value, no output is produced. (Also, no computations are made for the vectors Generated RHS and ERROR.)

d. Examples:

i) DIMENSION $C(4,4),B(4,1),A(4,5),X(4,1)$

::

     CALL SOLVE($C,B,A,X,4,5,1,DET,0$)

ii) CALL SOLVE($A,C,TEMP,X,N,M,L,DET,1$)

5.22. **SUBROUTINE SQTR(A,N)**

a. Function:

   The transpose of the square matrix $A$ is returned to the calling program in $A$ itself.

b. Calling Program:

   i) Matrix $A$ is of size $N$ rows by $N$ columns.

   ii) Matrix $A$ and integer $N$ should be defined.

   iii) The transposed matrix is returned in $A$ itself.

c. Limitations:

   Matrix $A$ must be square. The original matrix is destroyed.
5.23. SUBROUTINE SUB(A,B,C,M,N)

a. Function:
Subtract matrix B from matrix A and store the result in matrix C. \((C = A - B)\)

b. Calling Program:
   i) Matrices A, B and C are each of size M rows by N columns.
   ii) Matrices A and B as well as integers M and N should be defined.
   iii) Matrix C is defined in the subprogram.

c. Additional Notes:
The resultant matrix may be stored in one of the original matrices. Only in such a case, the specific original matrix will be destroyed.

d. Examples:
   i) CALL SUB(A,B,C,10,15)
   ii) CALL SUB(A,B,B,25,30)

5.24. SUBROUTINE TMULT(A,B,C,L,M,N)

a. Function:
The transpose of matrix A is postmultiplied by matrix B to give matrix C. \((C = A^T B)\).

b. Calling Program:
i) Matrices A, B and C have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>L rows by M columns</td>
</tr>
<tr>
<td>B</td>
<td>L rows by N columns</td>
</tr>
<tr>
<td>C</td>
<td>M rows by N columns</td>
</tr>
</tbody>
</table>
ii) Matrices A and B as well as integers L, M and N should be defined.

iii) The product matrix is returned in matrix C.

c. Limitations:
Matrix C should be distinct from matrices A and B. However, matrices A and B may be identical.

d. Examples:

i) CALL TMULT(A, B, C, L, M, N)

ii) CALL TMULT(A, A, B, L, M, M)

iii) CALL TMULT(A, A, B, N, N, N)

The examples below yield wrong results:

iv) CALL TMULT(A, B, A, L, L, L)

v) CALL TMULT(A, B, B, M, M, M)

5.25. SUBROUTINE TRANS(A, B, M, N)

a. Function:
Matrix A is transposed to give matrix B. \((B = A^T)\)

b. Calling Program:

i) Matrix A is of size M rows by N columns.
Matrix B is of size N rows by M columns.

ii) Matrix A and integers M and N should be defined.

iii) Matrix B is defined in the subprogram.

c. Example:

CALL TRANS(A, B, M, N)
After a few problems have been solved with the use of this package, the user will have gained enough experience to operate within its requirements and limitations, and frequent references to this Guide become unnecessary in day-to-day usage. All the same, matching the argument list of each subprogram and dimensioning arrays suitably are problems which may require some memory aid. A ready reference sheet has, therefore, been provided. The subprograms are listed alphabetically.
## READY REFERENCE SHEET (FLMXPK)

<table>
<thead>
<tr>
<th>NO.</th>
<th>SUBPROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>ADD(A,B,C,M,N)</td>
</tr>
<tr>
<td>2.</td>
<td>DETMT(A,DA,N)</td>
</tr>
<tr>
<td>3.</td>
<td>DIAG(A,DA,N)</td>
</tr>
<tr>
<td>4.</td>
<td>EV(A,S,N)</td>
</tr>
<tr>
<td>5.</td>
<td>GEVP(A,B,S,T,N)</td>
</tr>
<tr>
<td>6.</td>
<td>IEV(A,S,N)</td>
</tr>
<tr>
<td>7.</td>
<td>MINV(A,N,DET,NEXCH)</td>
</tr>
<tr>
<td>8.</td>
<td>MOVE(A,B,M,N)</td>
</tr>
<tr>
<td>9.</td>
<td>MULT(A,B,C,L,M,N)</td>
</tr>
<tr>
<td>10.</td>
<td>OUTE(A,I,J,TITLE,TITEL)</td>
</tr>
<tr>
<td>11.</td>
<td>OUTF(A,I,J,TITLE,TITEL)</td>
</tr>
<tr>
<td>12.</td>
<td>OUTG(A,I,J,TITLE,TITEL)</td>
</tr>
<tr>
<td>13.</td>
<td>PMULT(A,B,K,L,X)</td>
</tr>
<tr>
<td>14.</td>
<td>POSTM(A,B,K,L,X)</td>
</tr>
<tr>
<td>15.</td>
<td>RDCBC(A,M,N)</td>
</tr>
<tr>
<td>16.</td>
<td>RDCOLG(A,M,N)</td>
</tr>
<tr>
<td>17.</td>
<td>RDRBR(A,M,N)</td>
</tr>
<tr>
<td>18.</td>
<td>RDROWG(A,M,N)</td>
</tr>
<tr>
<td>19.</td>
<td>SCMUL(A,M,N,X)</td>
</tr>
<tr>
<td>20.</td>
<td>SINV(A,DA,N)</td>
</tr>
<tr>
<td>21.</td>
<td>SOLVE(C,B,A,X,N,M,L,DET,IFLAG)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td>SQTR(A,N)</td>
</tr>
<tr>
<td>23.</td>
<td>SUB(A,B,C,M,N)</td>
</tr>
<tr>
<td>24.</td>
<td>TMULT(A,B,C,L,M,N)</td>
</tr>
<tr>
<td>25.</td>
<td>TRANS(A,B,M,N)</td>
</tr>
</tbody>
</table>
7. ACKNOWLEDGMENTS

The authors are indebted to Mrs. Michele Kostem for her patient reading of the manuscript and typing the first copy.

Grateful appreciation is extended to Mrs. Anna L. Silfies who typed the final version.

8. REFERENCE

1. Iyengar, S. and Kostem, C. N.

FLMXPK - A MATRIX PACKAGE, Fritz Engineering Laboratory

Report No. 400.2 (under preparation)