USER'S MANUAL

FOR

THE MATRIX PACKAGE - FLMXPK

(For Data General MV/10000 Series Computers)
(AOS/VS Operating System and DG FORTRAN 77)

Version 1.0

by
Celal N. Kostem
Kenneth Seiler

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

March 1985

Fritz Engineering Laboratory Report No. 400.28
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>DISCLAIMER</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
</tbody>
</table>

## 1. INTRODUCTION

1.1 A Brief History  
1.2 Technical and Mathematical Aspects  
1.3 General Comments  
1.4 Single vs. Double Precision  
1.5 General Limitations  
1.6 Pertinent Publications  
1.7 Terminology  
1.8 Functional List of the Subprograms  
1.9 Critical Reminders  

## 2. DESCRIPTION OF THE SUBPROGRAMS

2.1 SUBROUTINE ADD(A,B,C,M,N)  
2.2 SUBROUTINE CORR(A,M,N,LABEL)  
2.3 SUBROUTINE DETNT(A,DA,N)  
2.4 SUBROUTINE DIAG(A,DA,N)  
2.5 SUBROUTINE EV(A,S,N)  
2.6 SUBROUTINE GEVP(A,B,S,T,N)  
2.7 SUBROUTINE IEV(A,S,N)  
2.8 SUBROUTINE MINV(A,N,DET,NEXCH)  
2.9 SUBROUTINE MOVE(A,B,M,N)  
2.10 SUBROUTINE MULT(A,B,C,L,M,N)  
2.11 SUBROUTINE OUTE(A,I,J,TITLE,LUO)  
2.12 SUBROUTINE OUTE8(A,I,J,LABEL)  
2.13 SUBROUTINE OUTF(A,I,J,TITLE,LUO)  
2.14 SUBROUTINE OUTF8(A,I,J,LABEL)  
2.15 SUBROUTINE OUTG(A,I,J,TITLE,LUO)  
2.16 SUBROUTINE OUTG8(A,I,J,LABEL)  
2.17 SUBROUTINE PMULT(A,B,K,L,X)  
2.18 SUBROUTINE POSTM(A,B,K,L,X)  
2.19 SUBROUTINE RDCBC(A,M,N,LUI)  
2.20 SUBROUTINE RDCBCL(A,M,N)  
2.21 SUBROUTINE RDCOLC(A,M,N,LUI)  
2.22 SUBROUTINE RDCLGL(A,M,N)  
2.23 SUBROUTINE RDROWC(A,M,N,LUI)  
2.24 SUBROUTINE RDRBRL(A,M,N)  
2.25 SUBROUTINE RDRWG(A,M,N,LUI)  
2.26 SUBROUTINE RDRWGL(A,M,N)  
2.27 SUBROUTINE SCHU(LA,H,N,X)  
2.28 SUBROUTINE SINV(A,DA,N)  
2.29 SUBROUTINE SOLVE(A,B,N,L,DET)  
2.30 SUBROUTINE SORT(A,N)  
2.31 SUBROUTINE SUB(A,B,C,M,N)
2.32 SUBROUTINE TMULT(A,B,C,L,M,N) 51
2.33 SUBROUTINE TRANS(A,B,M,N) 53
2.34 SUBROUTINE XABATC(A,B,C,L,M,X) 55
2.35 SUBROUTINE XABTA(A,B,L,N,X) 57
2.36 SUBROUTINE XABTC(A,B,C,L,M,N) 59
2.37 SUBROUTINE XATBAC(A,B,C,L,M,X) 61
2.38 SUBROUTINE XATBB(A,B,L,N,X) 63

3. READY REFERENCE SHEET 65
DISCLAIMER

No warranties, expressed or implied, concerning the accuracy, completeness, reliability, usability or suitability of the FLMXPK matrix software package for any purposes are made. The authors have no responsibility, financial or otherwise, for any consequences arising from the use of this document or the FLMXPK matrix software package.

The information contained herein is subject to change. Revisions will be issued to advise of such changes and/or additions.

This document may not be duplicated, in whole or in part, without the written consent of Dr. Celal N. Kostem.
ABSTRACT

FLMXPK, Fritz Laboratory Matrix Package, is a collection of 38 subprograms written primarily for undergraduate and graduate level engineering education and research. The subprograms can be used for the majority of the matrix operations encountered at these levels of activities.

The package is designed to be used with minimal mathematical or computer programming background. The main attribute of the package is the ease with which it can be learned and used.

The reported version of the FLMXPK is designed for 32-bit computers using FORTRAN77 compilers. In particular, the software package is tuned for Data General’s MV/10000 computers with AOS/VS operating system.
1. INTRODUCTION

The software package FLMXPK (Fritz Laboratory Matrix Package) is a collection of 38 subroutines. These subroutines will meet most of the needs of the users engaged in computer-based engineering or scientific computations employing matrix operations.

In the development of the package no attempt has been made to have it be an all-inclusive one. For example, the manipulation of "complex matrices," i.e. \( \mathbf{Z}(i,j) = x + iy \), is infrequent, except in specific disciplines. Thus, the subroutines handling complex matrices are not included. Similarly, the extraction of eigenvalues of nonsymmetric matrices is not as frequently encountered as that of symmetric matrices. This software package contains only the eigenvalue extraction routines for symmetric matrices.

The emphasis in the development of the FLMXPK has been to keep it simple and efficient. The primary objective in the development of the package was to make it simple enough to use so the user would concentrate attention and effort on the solution of the problem on-hand, rather than struggling with the various options of a matrix package.

1.1 A Brief History

FCMXPK (FORTRAN Callable Matrix Package) was written by Mr. E. T. Manning, Jr. and Drs. S. N. S. Iyengar and C. N. Kostem in 1968 to be used in CDC 6400 computers. Due to the simplicity of the use of this package, the users preferred FCMXPK over a number of "professionally developed" software packages. In 1970-1971 Drs. Sampath N. S. Iyengar and Celal N. Kostem's work led to FLMXPK, which was based on FCMXPK. FLMXPK contained numerous refinements and additions. Benchmark studies conducted in the 1972-74 period indicated that as compared to many sophisticated matrix packages developed and maintained by software houses, FLMXPK was core-efficient, faster, and more importantly, easy to learn and easy to use.

During the late 1970's and early 1980's the package was modified by Dr. C. N. Kostem to conform with ANSI FORTRAN IV, on CDC 6400 computers with NOS/BE operating system, and later on CYBER 730 with NOS operating system. In the early 1980's Mr. Pournert-Salkasen developed additional subroutines to facilitate interactive input and output via remote terminals. This was the beginning of the switching of the program from "batch" to "interactive" environment.

In 1984 Mr. Kenneth Seiler made substantial modifications to the original program so it could be used on Data General MV/10000.
computers with DG FORTRAN 77 and AOS/VS operating system. This user's manual is for the DG version of the program.

1.2 Technical and Mathematical Aspects

The report "FLMXPK-A MATRIX PACKAGE" (by S. N. S. Iyengar and C. N. Kostem; Fritz Engineering Laboratory Report No. 400.4, Lehigh University, September 1971) contains a detailed technical explanation of the main algorithms used, and the salient programming features. This report does not contain any material on the algorithms. This information can be obtained from the above report by Iyengar and Kostem.

The subroutines do not employ any unusual or uncommon algorithms. Every algorithm employed has been thoroughly tested not only by the developers of the package, but also by the scientific and engineering community as well.

1.3 General Comments

All original subroutines of the 1971 version of FLMXPK were batch oriented. The subroutines that have been added to the original thirty are: CORR, OUTF8, OUTF8, OUTG8, RDCBCL, RDCGL, RDRBRL, RDRWGL. These subroutines are designed to make the package an interactive one. With these additions the batch oriented subroutines have not been withdrawn. The subroutines designed for printing of matrices on line printers are still needed. These routines are OUTE, OUTF, and OUTG.

The subroutines that were designed to read the cards in batch mode are now conveniently used to read matrices from magnetic tapes. Even though it is not recommended, the users store matrices on disk files in a FORMAT-ted fashion, and read these matrices by any one of the "read subroutines," having names starting "RD..." (and not ending with "..L").

A careful review of the subroutines with names starting with the letter "X" indicates that these subroutines correspond to a string of prespecified matrix operations used in many engineering computations. Through the use of the "X..." subroutines it should be possible to perform these sequences of operations through one CALL statement.
1.4 Single vs. Double Precision

In the 1971 version of FLMXPK which runs on Control Data Corporation computers all variables were defined as single precision variables. Extensive tests were conducted using "well-conditioned" as well as "relatively ill-conditioned" matrices. These tests indicated that no additional accuracy was gained through the use of double precision. This has been due to the inherently long word-length of the CDC computers.

However, limited studies conducted with "relatively ill-conditioned" matrices clearly indicated that in 32-bit word length computers it was essential to use double precision, if the matrices in question have some of the characteristics of ill-conditionedness. This has led to the decision that all real variables used in the reported version of FLMXPK are in DOUBLE PRECISION.

It is recognized that in many matrix operations the single precision would suffice due to the characteristics of the matrices operated on. It might have been possible to have two versions of the subroutines, one in single precision and another in double precision. However, it was felt that this could lead to confusion on the part of the user.

1.5 General Limitations

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

In all subprograms, "variable" or "adjustable" dimensions are used for several arrays. The user must, therefore, prescribe exact dimensions for all the arrays to be handled by this package. Overdimensioning, except under special circumstances of usage, may lead to wrong results. Underdimensioning invariably produces wrong results or aborts the execution of the program.

1.6 Pertinent Publications

The mathematical description of the algorithms used in the original version of FLMXPK, as well as some remarks regarding the programming strategy, are presented in:

FLMXPK-A MATRIX PACKAGE
by Sampath N. S. Iyengar and Celal N. Kostem
Fritz Engineering Laboratory Report No. 400.4
Lehigh University, September 1971
(Copyrighted by Sampath N.S.Iyengar and Celal N. Kostem, 1971)

The listing of the FLMXPK software package (Data General MV/10000, DG
FORTRAN 77, AOS/VS Operating System) is presented in:

FLMXPK-A MATRIX PACKAGE
SOURCE CODE LISTING (DG MV/10000 VERSION)
by Celal N. Kostem and Kenneth Seiler
Fritz Engineering Laboratory Report No. 400.30
Lehigh University, March 1985.

1.7 Terminology
In Chapter 2, in the definition of operations, the following notation is used:

[A] indicates that A is a two dimensional array, i.e. matrix.
[X] indicates that X is a one dimensional array, i.e. vector.
[A] T indicates the transpose of [A].

1.8 Functional List of the Subprograms
The matrix operations that can be performed using specific subprograms are as follows:

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>SUBPROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add matrices</td>
<td>ADD, DETMT, MINV, SINV, SOLVE</td>
</tr>
<tr>
<td>Determinant of a matrix</td>
<td>DIAG, EV, IEV, GEVP</td>
</tr>
<tr>
<td>Create a diagonal matrix</td>
<td>MINV, SINV</td>
</tr>
<tr>
<td>Eigenvalues of symmetric matrices</td>
<td>MOVE</td>
</tr>
<tr>
<td>Invert matrix</td>
<td>MULT, PMULT, POSTM, SCMUL, TMULT, XABATC, XABTA, XABTC, XATBAC, XATBB</td>
</tr>
<tr>
<td>Copy matrix</td>
<td>OUTE, OUTF, OUTG</td>
</tr>
<tr>
<td>Multiply matrices</td>
<td>OUTE8, OUTF8, OUTG8</td>
</tr>
<tr>
<td>Print matrix</td>
<td>RDCBC, RDCOLG, RDRBR, RDRROWG</td>
</tr>
<tr>
<td>Display matrix</td>
<td>RDCBCL, RDCLGL, RDRBRL, RDRUGL</td>
</tr>
<tr>
<td>Read matrix (FORMATTED READ)</td>
<td>SOLVE</td>
</tr>
<tr>
<td>Read matrix (LIST DIRECTED READ)</td>
<td>SORT, TRANS</td>
</tr>
<tr>
<td>Simultaneous equations</td>
<td>SUB</td>
</tr>
<tr>
<td>Transpose matrix</td>
<td>CORR</td>
</tr>
<tr>
<td>Subtract matrices</td>
<td></td>
</tr>
<tr>
<td>Correct/edit matrix</td>
<td></td>
</tr>
</tbody>
</table>
1.9 Critical Reminders

* All real variables used in the subprograms are double precision variables. Variables used in the call statements to the subprograms must be defined as double precision in the calling program or subprograms.

* All variables used for titles and/or labels for the display or printing of matrices are character strings of specified length. These variables used in the call statements to the subprograms must be defined as character variables of specified length in the calling program or subprograms.

* Some subprograms "destroy" the matrices they operate on. These actions are indicated in the description of each subroutine in the next chapter. If these arrays will be needed later in the program, duplicates of these matrices must be created early in the program via subprogram "move."

* To prevent unpredictable errors the matrices to be used by the subprograms listed in this report must be correctly dimensioned in the main program.
2. DESCRIPTION OF THE SUBPROGRAMS

From the user's viewpoint, there are, in all, 38 subroutines in this package. They will be described in alphabetical order under the following headings:

(1) Function

(2) Calling Program:

(a) 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 REAL A(10,15). If it is a subprogram, the dimension statement is either of the same form or of the form of REAL A(M,N) where M and N have been defined through operations prior to the use of the subprogram. (The user is obliged, in the latter instance, to include A, M and N in the argument list of the subprogram.)

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.

All real variables used in the subroutines described herein are defined as DOUBLE PRECISION variables. It is essential that the variables used in the CALL statements must be declared as DOUBLE PRECISION in the calling program or subprograms.

The labels or titles used in the printing or display of matrices, e.g. OUTE, OUTE8, can be defined at the CALL statement using apostrophes, such as:

```
CALL OUTE(A,M,N,"TITLE TO BE DISPLAYED",LUO),
```

or can be handled by defining it prior to the CALL as follows:

```
TITLE="TITLE TO BE DISPLAYED"
CALL OUTE(A,M,N,TITLE,LUO)
```

If the latter approach is taken, it is essential that variable TITLE be defined as a CHARACTER string in the calling program or subprogram.

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

(c) Values Returned to the Calling Program
(3) Limitations (if any):
The general limitations mentioned in Section 1.5 will not be repeated.

(3) or (4) Additional Notes (if any)

(3) or (4) or (5) Examples of CALL statements
2.1 SUBROUTINE ADD(A,B,C,M,N)

2.1.1 Function:
Add matrices \([A]\) and \([B]\) and store the sum matrix \([C]\). \(([C] = [A] + [B])\).

2.1.2 Calling Program:
(a) Each matrix is of size \(M\) rows by \(N\) columns.
(b) Matrices \([A]\) and \([B]\), as well as integers \(M\) and \(N\), must be defined.
(c) Matrix \([C]\) is defined in the subprogram.

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

2.1.4 Examples:
CALL ADD(A,B,C,M,N)
CALL ADD(A,B,C,15,20)
CALL ADD(A,B,A,10,15)
2.2 SUBROUTINE CORR(A,M,N,LABEL)

2.2.1 Function:
Matrices can be edited and corrected with the use of SUBROUTINE CORR(A,M,N,LABEL).

2.2.2 Calling Program:
(a) Matrix [A] is of size M rows by N columns.
(b) Integers M and N should be defined.
(c) LABEL is a character variable of up to 40 characters displayed with the matrix.
(d) The matrix with all changes will be returned to the calling program.

2.2.3 Additional Notes:
SUBROUTINE CORR(A,M,N,LABEL) can only be called for interactive use. Called, the subroutine will display the supplied matrix [A] by calling SUBROUTINE OUTG8, then will give the user three options:

(1) Display the output again.
(2) Make corrections.
(3) Do not make corrections, i.e. matrix [A] is correct.

If the user selects to make changes, the user may change either individual elements or entire rows of the matrix. The subroutine
will prompt the user for input, and once the corrections are complete, the edited matrix is returned to the program.

SUBROUTINE CORR is called at the end of input by all four of the interactive "read routines," i.e. RDCBCL, RDCLGL, RDRBRL, RDRWGL, and may also be called by the user to correct or to modify any of the current matrices.

2.2.4 Examples:

CALL CORR(A,N,N,LABEL)

CALL CORR(A,12,8,"OLD H MATRIX")
2.3 SUBROUTINE DETMT(A, DA, N)

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

2.3.2 Calling Program:
(a) Matrix \([A]\) is of size \(N\) rows by \(N\) columns.
(b) Matrix \([A]\) and integer \(N\) should be defined.
(c) DA is defined in the subprogram.

2.3.3 Limitations:
The original matrix \([A]\) is destroyed.

2.3.4 Examples:
CALL DETMT(A, DA, N)

CALL DETMT(ARRAY, DET, 20)
2.4 SUBROUTINE DIAG(A,DA,N)

2.4.1 Function:
A diagonal matrix \([A]\) is generated as follows:
Each diagonal element has the value DA, and each off-diagonal element has the value of zero.

2.4.2 Calling Program:
(a) Matrix \([A]\) is of size \(N\) rows by \(N\) columns.
(b) The value of the diagonal element \(DA\) and integer \(N\) must be defined.
(c) Matrix \([A]\) is defined in the subprogram.

2.4.3 Examples:
CALL DIAG(A,DA,N)
CALL DIAG(A,1.0,15)
CALL DIAG(ARRAY,DE,NSIZE)
2.5  SUBROUTINE EV(A,S,N)

2.5.1 Function:

Eigenvalues and eigenvectors of the symmetric matrix [A] are computed.

2.5.2 Calling Program:

(a) Matrices [A] and [S] are of size N rows by N columns.

(b) Matrix [A] and integer N must be defined.

(c) 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.

2.5.3 Limitations:

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

2.5.4 Additional Notes:

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

2.5.5 Examples:

CALL EV(A,S,N)

CALL EV(ARRAY,EVFC,10)
2.6 SUBROUTINE GEVP(A,B,S,T,N)

2.6.1 Function:
Eigenvalues and eigenvectors of the given matrix [A] where [A] [X] = \lambda
[B] [X] are computed. Both matrices [A] and [B] are symmetric, and
further, matrix [B] is also positive-definite.

2.6.2 Calling Program:
(a) Matrices [A], [B], and [S] are of the same size N rows by N
columns. [T] is a vector of size N elements.
(b) Matrices [A] and [B], as well as integer N, should be defined.
(c) 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.

2.6.3 Limitations:
Matrices [A] and [B] must be symmetric. Matrix [B] must also be
positive-definite. Both 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:

SUBROUTINE EV(A,S,N)
SUBROUTINE MULT(A,B,C,L,M,N)
SUBROUTINE POSTH(A,B,K,L,X)
2.6.4 Examples:

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

CALL GEVP(EVAL,EVEC,S,TEMP,10)
2.7  SUBROUTINE IEV(A,S,N)

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

2.7.2  Calling Program:
Same as SUBROUTINE EV(A,S,N)

2.7.3  Limitations:
Same as SUBROUTINE EV(A,S,N)

2.7.4  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
squares of the elements above the major diagonal (of the original
matrix \([A]\)) is computed first. This is called the initial threshold.
A final threshold value of one-millionth of such a 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.
These successive CALL-s must be to SUBROUTINE IEV(A,S,N). The
following rules apply.

(a) SUBROUTINE EV(A,S,N) must be CALL-ed only once and before
SUBROUTINE IEV(A,S,N) is called.
(b) Subsequently SUBROUTINE IEV(A,S,N) may be CALL-ed the required
number of times to achieve the desired accuracy. If the total number
of "n" CALL-s are made to (both) the subprograms, each off-diagonal
element will be reduced in absolute value to (at least) 10**(-6n)
times the initial threshold.
(c) 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
insignificant changes in the eigenvectors, and practically no changes
in the eigenvalues (apparently because these are already very close
to exact values). An excessive number of improvement cycles may
result in an underflow in the machine.

2.7.5 Example:

CALL EV(A,S,N)
.
.
.
CALL IEV(A,S,N)
.
.
.
CALL IEV(A,S,N)
2.8  **SUBROUTINE MINV(A,N,DET,NEXCH)**

2.8.1  **Function:**

The matrix \([A]\) is inverted in its own space and its determinant is computed.

2.8.2  **Calling Program:**

(a) Matrix \([A]\) is of size \(N\) rows by \(N\) columns. Vector \(\{NEXCH\}\) is of size \(N\) elements.

(b) Matrix \([A]\) and integer \(N\) should be defined.

(c) 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 \(DET\).

2.8.3  **Limitations:**

The original matrix \([A]\) is destroyed in the subprogram. See also "Additional Notes" under **SUBROUTINE SINV(A,DA,N).**

2.8.4  **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.

2.8.5  **Examples:**
CALL MINV(A,N,DET,NEXCH)
CALL MINV(ARRAY,10,DET,TEMP)
2.9 SUBROUTINE MOVE(A,B,M,N)

2.9.1 Function:
Matrix [A] is copied as matrix [B]. ([B] = [A])

2.9.2 Calling Program:
(a) Matrices [A] and [B] are of size M rows by N columns.
(b) Matrix [A] and integers M and N should be defined.
(c) Matrix [B] is defined in the subprogram.

2.9.3 Additional Notes:
When certain subprograms such as SUBROUTINE DETMT(A,DA,N) of this package are CALL-ed, the original matrices are 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.

2.9.4 Examples:
CALL MOVE(A,B,M,N)
CALL MOVE(ARRAY,SAME,10,15)
2.10 SUBROUTINE MULT(A,B,C,L,M,N)

2.10.1 Function:
Matrix [A] is post-multiplied by matrix [B] to yield matrix [C].
([C] = [A] [B])

2.10.2 Calling Program:
(a) 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>

(b) Matrices [A] and [B] and integers L, M and N should be defined.
(c) The product matrix [C] is defined in the subprogram.

2.10.3 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).

2.10.4 Examples:
CALL MULT(A,B,C,L,M,N)
CALL MULT(A,A,C,N,N,N)
The examples below yield wrong results:

CALL MULT(A,B,A,L,M,M)
CALL MULT(A,B,B,L,L,N)
CALL MULT(A,A,A,N,N,N)
2.11 SUBROUTINE OUTE(A,I,J,TITLE,LUO)

2.11.1 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 the printing is done 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 is less than or equal to 10 and I less than or equal to 25, printing is completed on one page. If J is greater than 10, the first 10 columns are printed, until all rows (25 or less to a page) are exhausted. The second 10 columns (or less) are printed, until all rows are exhausted, and so on. The elements of matrix [A] are output in E-Format. Five digits appear to the right of the decimal point (E11.5).

2.11.2 Calling Program:
(a) Matrix [A] is of size I rows by J columns.
(b) Matrix [A] and integers I and J should be defined. Also, the user's label must be provided as a CHARACTER string of up to 70 characters through an alphanumeric variable or a string enclosed in apostrophes corresponding to the argument. See examples of CALL statement.
(c) \textsc{luo} is an integer variable corresponding to the output unit desired. Unit No. 12 is the preconnected \texttt{LISTFILE}, which can be used for this purpose. However, for \textsc{luo} any other unit number can be assigned.

(d) No formal "values" are returned by this subprogram.

2.11.3 Additional Notes:

This subprogram is recommended for use in preference to \texttt{SUBROUTINE OUTF(A,I,J,TITLE,LUO)} whenever the magnitudes of the elements of the matrix to be printed are unknown, unpredictable, or exceed the field F11.5. A slight sacrifice of easy readability is implicit.

2.11.4 Examples:

\begin{verbatim}
CALL OUTE(A,I,J,TITLE,LUO)

VAR = "MATRIX OF REDUNDANTS"
LUO = 12    (Note: A \texttt{LISTFILE} is specified for output.)
CALL OUTE(A,I,J,VAR,LUO)

CALL OUTE(A,I,J,"ORIGINAL MATRIX",12)
\end{verbatim}
2.12 SUBROUTINE OUTE8(A,I,J,LABEL)

2.12.1 Function:
Matrix \([A]\) of size \(I\) rows by \(J\) columns is displayed. This subroutine is similar to OUTE, except that it is formatted for an 80 column field width, suitable for displaying results on a terminal. The matrix is labeled at the top of each screen with a label provided by the user. The word "CONTINUED" in parentheses appears with the label if the display is done on more than one screen. The matrix size is indicated below the label.

Rows and columns are numbered. On any one screen the maximum number of rows displayed is 25, and the maximum number of columns is 5. Hence, if \(J\) is less than or equal to 5 and \(I\) is less than or equal to 25, the display will be completed in one "screen."

The elements of matrix \([A]\) are displayed in E-FORMAT, with five digits to the right of the decimal point (E12.5).

2.12.2 Calling Program:
(a) Matrix \([A]\) is of size \(I\) rows by \(J\) columns.
(b) Matrix \([A]\) and integers \(I\) and \(J\) should be defined. Also, the user’s label must be provided as a character string of up to 40 characters through an alphanumeric variable or a string enclosed in apostrophes (‘) corresponding to the argument LABEL. See examples of OUTE8 -25- OUTE8
CALL STATEMENT.

(c) No formal values are returned by this subprogram.

2.12.3 Additional Notes:
This subprogram is recommended for use in preference to SUBROUTINE OUTF8 whenever the magnitudes of the elements of the matrix to be displayed are unknown, unpredictable or exceed the field F12.5. A slight sacrifice of easy readability is implicit.

2.12.4 Examples:

CALL OUTE8(A,I,J,_LABEL)

HEADER="MATRIX OF REDUNDANTS"

CALL OUTE8(A,I,J,HEADER)

CALL OUTE8(A,I,J,"STIFFNESS MATRIX")
2.13  **SUBROUTINE OUTF(A,I,J,TITLE,LUO)**

2.13.1  **Function:**

All the details are the same as in **SUBROUTINE OUTE(A,I,J,TITLE,LUO)** except that the elements are output in F-FORMAT. Five digits appear to the right of the decimal point, and a maximum of five digits (four, if the value is negative) appear to its left. (Fl1.5)

2.13.2  **Calling Program:**

Same as **SUBROUTINE OUTE(A,I,J,TITLE,LUO)**

2.13.3  **Limitations:**

The "largest" numbers that can be printed are of the form abcde.fghij or -bcde.fghij. If an attempt to print a number "larger" than these is made, asterisks (*) will appear in the corresponding field.

2.13.4  **Examples:**

```fortran
CALL OUTF(A,I,J,TITLE,LUO)
CALL OUTF(A,I,J,"MATRIX A",17)
```
2.14  **SUBROUTINE OUTF8(A,I,J,LABEL)**

2.14.1 **Function:**
All the details are the same as in **SUBROUTINE OUTE8(A,I,J,LABEL)**, except that elements are displayed 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).

2.14.2 **Calling Program:**
Same as **SUBROUTINE OUTE8(A,I,J,LABEL)**.

2.14.3 **Limitations:**
The largest numbers that can be printed are of the form abcdef.ghijk or -bcdef.ghijk. If an attempt to display a number "larger" than these is made, asterisks (*) will appear in the corresponding field.

2.14.4 **Examples:**
CALL OUTF8(A,I,J,LABEL)
CALL OUTF8(A,12,15,"MATRIX FX")
2.15  **SUBROUTINE OUTG(A,I,J,TITLE,LUO)**

2.15.1  **Function:**
All the details are the same as in **SUBROUTINE OUTE(A,I,J,TITLE,LUO)**, except that the elements are output in G-FORMAT.

2.15.2  **Calling Program:**
Same as **SUBROUTINE OUTE(A,I,J,TITLE,LUO)**

2.15.3  **Additional Notes:**
Five significant digits appear on output, if the absolute value "x" of the element being printed is greater or equal to 0.1 but less than 100,000. Otherwise, the output is in E-FORMAT for the element.

2.15.4  **Examples:**

```
CALL OUTG(A,I,J,TITLE,LUO)
CALL OUTG(A,I,J," ",18)
```
2.16  SUBROUTINE OUTG8(A,I,J,LABEL)

2.16.1  Function:
All the details are the same as SUBROUTINE OUTE8(A,I,J,LABEL), except that the elements are output in G-FORMAT.

2.16.2  Calling Program:
Same as SUBROUTINE OUTE8(A,I,J,LABEL).

2.16.3  Additional Notes:
Five significant digits appear on output, if the absolute value "X" of the element being printed is greater or equal to 0.1 but less than 100,000. Otherwise, the output is in E-FORMAT for the element.

2.16.4  Examples:
CALL OUTG8(A,I,J,LABEL)
CALL OUTG8(FM,10,10,"MASS MATRIX")
2.17 SUBROUTINE PMULT(A,B,K,L,X)

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

2.17.2 Calling Program:
(a) Matrices \([A]\) and \([B]\), and vector \([X]\), have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
</table>
| \([A]\) | \(K\) rows by \(L\) columns  
           | (may be square, \(K=L\)) |
| \([B]\) | \(L\) rows by \(L\) columns |
| \([X]\) | \(L\) elements     |

(b) Matrices \([A]\) and \([B]\), and integers \(K\), and \(L\) should be defined.
(c) The product matrix is returned in matrix \([A]\). Vector \([X]\) is required in the subprogram for computations only.

2.17.3 Limitations:
Matrix \([B]\) must be square. The original matrix \([A]\) is destroyed. If matrix \([A]\) is square, it should differ from matrix \([B]\) at least by name.
2.17.4 Examples:

CALL PMULT(A,B,K,L,X)

CALL PMULT(A,B,K,K,X)

The example below yields wrong results:

CALL PMULT(A,A,L,L,X)
2.18 SUBROUTINE POSTM(A,B,K,L,X)

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

2.18.2 Calling Program:
(a) Matrices [A] and [B], and vector [X] have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]</td>
<td>K rows by K columns</td>
</tr>
<tr>
<td>[B]</td>
<td>K rows by L columns</td>
</tr>
<tr>
<td></td>
<td>(may be square, K=L)</td>
</tr>
<tr>
<td>[X]</td>
<td>K elements</td>
</tr>
</tbody>
</table>

(b) Matrices [A] and [B], and integers K, and L should be defined.
(c) The product matrix is returned in matrix [B]. Vector [X] is required in the subprogram for computations only.

2.18.3 Limitations:
Matrix [A] must be square. The original matrix [B] is destroyed. If matrix [B] is square, it should differ from matrix [A] at least by name.
2.18.4 Examples:

CALL POSTM(A, B, K, L, X)
CALL POSTM(A, B, L, L, X)

The example below yields wrong results:

CALL POSTM(A, A, K, K, X)
2.19 SUBROUTINE RDCBC(A,M,N,LUI)

2.19.1 Function:
Elements of matrix \([A]\) are defined (column by column) by reading in values from "data card images," or from a specific file.

2.19.2 Calling Program:
(a) Matrix \([A]\) is of size \(M\) rows by \(N\) columns.
(b) Integers \(M\) and \(N\) should be defined. The number of "cards," or lines of entry, required per column of matrix \([A]\) is \((M+7)/8\) (integer division).
(c) Matrix \([A]\) is defined in the subprogram.
(d) \(LUI\) is the unit number (integer) from which the data is to be read. Unit No. 9 is to be used for the preconnected DATAFILE. However, the user can assign any other connected unit number.

2.19.3 Additional Notes:
(a) 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 not assumed to be filled with zeros, but are blanks.
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 235.0.
(b) **Order of 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 card go to \(A(9,1), A(10,1), \ldots, A(14,1)\). And so on.

2.19.4 **Examples:**

```
CALL RDCBC(A,M,N,LUI)

CALL RDCBC(A,14,6,9)
```
2.20 SUBROUTINE RDCBCL(A,M,N)

2.20.1 Function:
Elements of matrix [A] are defined (column-by-column) by reading in values from the terminal, in LIST DIRECTED FORMAT.

2.20.2 Calling Program:
(a) Matrix [A] is of size M rows and N columns.
(b) Integers M and N should be defined.
(c) Matrix [A] is defined in the subprogram.

2.20.3 Additional Notes:
(a) Format Control: The input to this subroutine is list directed, with the entries separated by a space or a comma (or any other valid delimiter). If the value is to be zero, a zero must be entered.
(b) Order of Assigning Values: The values read from the terminal will be assigned as follows: A(1,1), A(2,1), ..., A(M,1). The program will then prompt for the second column, and the values will be assigned as A(1,2), A(2,2), ..., A(M,2). And so on.
(c) This subroutine calls SUBROUTINE CORR(A,M,N,LABEL) once all values have been assigned.

2.20.4 Examples:
CALL RDCBCL(A,M,N)
CALL RDCBCL(A,14,6)
2.21 SUBROUTINE RDCOLG(A,M,N,LUI)

2.21.1 Function:
Elements of matrix [A] are defined by reading in values from "data card images," or from a specific file. The elements are assumed to be in a continuous string of columns of matrix [A].

2.21.2 Calling Program:
(a) Matrix [A] is of size M rows by N columns.
(b) Integers M and N should be defined.
The number of data card images, or lines of entry, required is (M*N+7)/8 (integer division).
(c) Matrix [A] is defined in the subprogram.
(d) Same as in SUBROUTINE RDCBC, i.e. Section 2.19.2.(d).

2.21.3 Additional Notes:
(a) Same as in SUBROUTINE RDCBC(A,M,N,LUI), i.e. Section 2.19.3.(a).
(b) Order of Assigning Values: Assume matrix [A] is of size 14 rows by 6 columns. Eleven data card images, or lines of entry, are required. The eight values on the first data card will be assigned in order to A(1,1), A(2,1), ..., A(8,1). The first six values on the second card go to A(9,1), A(10,1), ..., A(14,1). The last two values on the second card go to A(1,2), A(2,2), ..., A(10,2). And so on.
2.21.4 Examples:

CALL RDCOLG(A,M,N,LUI)

CALL RDCOLG(A,14,6,9)
2.22 SUBROUTINE RDCLGL(A,M,N)

2.22.1 Function:
Elements of matrix [A] are defined by reading in values from the terminal. The elements are assumed to be in a continuous string of columns of matrix [A].

2.22.2 Calling Program:
(a) Matrix [A] is of size M rows by N rows.
(b) Integers M and N should be defined.
(c) Matrix [A] is defined in the subprogram.

2.22.3 Additional Notes:
(a) Same as in SUBROUTINE RDCBCL(A,M,N), i.e. Section 2.20.3.(a).
(b) Order of Assigning Values: The values read from the terminal will be assigned as follows: A(1,1), A(2,1), A(3,1), ....., A(M,1), A(1,2), A(2,2), A(3,2), ....., up to A(M,N).
(c) Same as in subroutine RDCBCL(A,M,N), i.e. Section 2.20.3.(c).

2.22.4 Examples:
CALL RDCLGL(A,M,N)
CALL RDCLGL(A,15.18)
2.23  **SUBROUTINE RDRBR(A,M,N,LUI)**

2.23.1  **Function:**

Elements of matrix \([A]\) are defined (row by row) by reading in values from "data card images," or from a specific file.

2.23.2  **Calling Program:**

(a) Matrix \([A]\) is of size \(M\) rows by \(N\) columns.

(b) Integers \(M\) and \(N\) should be defined.

The number of "data card images," or lines of entry, required per row of matrix \([A]\) is \((N+7)/8\) (integer division).

(c) Matrix \([A]\) is defined in the subprogram.

(d) Same as SUBROUTINE RDCBC, i.e. Section 2.19.2.(d).

2.23.3  **Additional Notes:**

(a) Same as in SUBROUTINE RDCBC(A,M,N,LUI), i.e. Section 2.19.3.(a).

(b) **Order of Assigning Values:** Assume matrix \([A]\) is of size 14 rows by 6 columns. One "data card image," or line of entry, is required per row. Hence, the total number of data cards required is 14. The six values on card number 1 (I ranges in value from 1 to 14) are assigned in order to \(A(I,1)\), \(A(I,2)\),..., \(A(I,6)\).

2.23.4  **Examples:**

    CALL RDRBR(A,M,N,LUI)
    CALL RDRBR(A,14,6,LUI)
2.24 SUBROUTINE RDBRBR(A,M,N)

2.24.1 Function:

Elements of matrix [A] are defined (row-by-row) by reading values from the terminal.

2.24.2 Calling Program:

(a) Matrix [A] is of size M rows by N columns.
(b) Integers M and N should be defined.
(c) Matrix [A] is defined in the subprogram.

2.24.3 Additional Notes:

(a) Same as in SUBROUTINE RDCBCL(A,N,N), i.e. Section 2.20.3.(a).
(b) Order of Assigning Values: The values to be read from the terminal will be assigned as follows: A(1,1), A(1,2), A(1,3), ..., A(1,N). The program will then prompt for the second row, and the values will be assigned as A(2,1), A(2,2), A(2,3), ..., A(2,N). And so on.
(c) Same as in SUBROUTINE RDCBCL(A,M,N), i.e. Section 2.20.3.(c).

2.24.4 Examples:

CALL RDBRBR(A,M,N)

CALL RDBRBR(A,5,10)
2.25 **SUBROUTINE RDROWG(A,M,N,LUI)**

2.25.1 **Function:**
Elements of matrix [A] are defined by reading in values from "data card images," or from a specific file. The elements are assumed to be in a continuous string of rows of matrix [A].

2.25.2 **Calling Program:**
Same as in SUBROUTINE RDCOLG(A,M,N,LUI), i.e. Section 2.19.2.

2.25.3 **Additional Notes:**
(a) Same as in SUBROUTINE RDCBC(A,M,N,LUI), i.e. Section 2.19.3.(a).
(b) Assume matrix [A] is of size 14 rows by 6 columns. Eleven "data card images," or lines of entry, 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 go to A(2,1), A(2,2). The first two values on the second card go to A(2,3), A(2,4), A(2,5), A(2,6). The last four values on the second card go to A(3,1), A(3,2), A(3,3), A(3,4). And so on.

2.25.4 **Examples:**
CALL RDROWG(A,M,N,LUI)
CALL RDROWG(A,4,6,9)
2.26 SUBROUTINE RDRWGL(A,M,N)

2.26.1 Function:
Elements of matrix [A] are defined by reading values from the terminal. The elements are assumed to be in a continuous string of rows of matrix [A].

2.26.2 Calling Program:
(a) Matrix [A] is of size M rows by N columns.
(b) Integers M and N should be defined.
(c) Matrix [A] is defined in the subprogram.

2.26.3 Additional Notes:
(a) Same as in SUBROUTINE RDCBCL(A,M,N), i.e. Section 2.20.3.(a).
(b) Order of Assigning Values: The values read from the terminal will be assigned as follows: A(1,1), A(1,2), A(1,3), ...., A(1,N), A(2,1), A(2,2), .... up to A(M,N).
(c) Same as in SUBROUTINE RDCBCL(A,M,N), i.e. Section 2.20.3.(c).

2.26.4 Examples:
CALL RDRWGL(A,M,N)

CALL RDRWGL(A,12,12)
2.27 SUBROUTINE SCMUL(A,M,N,X)

2.27.1 Function:

Elements of matrix [A] are multiplied by the scalar quantity "X".

\[[A]=X[A]\]

2.27.2 Calling Program:

(a) Matrix [A] is of size M rows by N columns.

(b) Matrix [A] and the scalar multiplier "X" as well as integers M and N should be defined.

(c) The modified matrix is returned in [A] itself.

2.27.3 Limitations:

The original matrix is destroyed.

2.27.4 Examples:

CALL SCMUL(A,M,N,X)

REI = 1.0 / EI

CALL SCMUL(A,M,N,REI)

CALL SCMUL(A,M,N,1.0/30000.0)
SUBROUTINE SINV(A,DA,N)

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

2.28.2 Calling Program:
(a) Matrix [A] is of size N rows by N columns.
(b) Matrix [A] and integer N should be defined. The subprogram utilizes only the elements on and above the diagonal of the original matrix [A]. Hence, if so desired, only these elements of matrix [A] may be defined.
(c) The inverse is returned in matrix [A] itself. DA stores the value of the determinant of the original matrix [A].

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

2.28.4 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 MINV(A,Det,Inv) is not ideally symmetric.
because of round-off errors in the machine.

2.28.5 Examples:

CALL SINV(A,DA,N)

CALL SINV(K,DETK,20)
2.29  **SUBROUTINE SOLVE(A,B,N,L,DET)**

2.29.1  **Function:**

A system of linear simultaneous equations \([A] [X] = [B]\) is solved.

2.29.2  **Calling Program:**

(a) Matrices \([A]\) and \([B]\) have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient Matrix ([A])</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>
</tbody>
</table>

(b) Matrices \([A]\) and \([B]\) as well as integers \(N\) and \(L\) should be defined.

(c) The solution matrix is returned in \([B]\). The value of the determinant of the coefficient matrix \([A]\) is returned in \(DET\).

2.29.3  **Limitations:**

The original matrices \([A]\) and \([B]\) are destroyed.

2.29.4  **Examples:**

```
CALL SOLVE(A,B,N,L,DET)
CALL SOLVE(COEFF,KHS,N,L,DA)
```
2.30 SUBROUTINE SQRT(A,N)

2.30.1 Function:
The transpose of the square matrix \([A]\) is returned to the calling program in \([A]\) itself. \((A) = (A)^\top\).

2.30.2 Calling Program:
(a) Matrix \([A]\) is of size \(N\) rows by \(N\) columns.
(b) Matrix \([A]\) and integer \(N\) should be defined.
(c) The transposed matrix is returned in \([A]\) itself.

2.30.3 Limitations:
Matrix \([A]\) must be square. The original matrix is destroyed.

2.30.4 Examples:
CALL SQRT(A,N)
CALL SQRT(ASQ,10)
2.31    SUBROUTINE SUB(A,B,C,M,N)

2.31.1   Function:
Subtract matrix [B] from matrix [A] and store the result in matrix
[C]. ([C] = [A] - [B])

2.31.2   Calling Program:
(a) Matrices [A], [B] and [C] are each of size M rows by N columns.
(b) Matrices [A] and [B], as well as integers M and N should be
defined.
(c) Matrix [C] is defined in the subprogram.

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

2.31.4   Examples:
        CALL SUB(A,B,C,M,N)
        CALL SUB(A,B,C,10,15)
        CALL SUB(A,B,B,25,30)
2.32 SUBROUTINE TMULT(A,B,C,L,M,N)

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

2.32.2 Calling Program:
(a) 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>

(b) Matrices [A] and [B], as well as integers L, M and N, should be defined.
(c) The product matrix is returned in matrix [C].

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

2.32.4 Examples:
CALL TMULT(A,B,C,L,M,N)
CALL TMULT(A,A,B,L,M,M)
CALL TMULT(A,A,B,N,N,N)

The examples below yield wrong results:

CALL TMULT(A,B,A,L,L,L)
CALL TMULT(A,B,B,M,M,M)
CALL TMULT(A,A,A,N,N,N)
2.33 **SUBROUTINE TRANS(A,B,M,N)**

2.33.1 **Function:**
Matrix \([A]\) is transposed to give matrix \([B]\). \(([B] = [A]^\top)\).

2.33.2 **Calling Program:**
(a) Matrix \([A]\) is of size \(M\) rows and \(N\) columns. Matrix \([B]\) is of size \(N\) rows by \(M\) columns.
(b) Matrix \([A]\) and integers \(M\) and \(N\) should be defined.
(c) Matrix \([B]\) is defined in the subprogram.

2.33.3 **Limitations:**
Even if matrix \([A]\) is square, it should be distinct from the (square) matrix \([B]\).

2.33.4 **Additional Notes:**
If matrix \([A]\) is square and the original matrix may be destroyed, it is preferable to use **SUBROUTINE SQRT(A,N)**.

2.33.5 **Examples:**
```
CALL TRANS(A,B,M,N)
CALL TRANS(A,B,10,15)
CALL TRANS(A,B,N,N)
```

The example below yields wrong results:

```
TRANS
```
CALL TRANS(A,A,N,N)
SUBROUTINE XABATC(A,B,C,L,M,X)

2.34.1 Function:
The matrix product $[A][B][A]^\top$ is formed in matrix $[C]$. Matrix $[B]$ is symmetric. ($[C] = [A][B][A]^\top$)

2.34.2 Calling Program:
(a) Matrices $[A]$, $[B]$, and $[C]$ and vector $[X]$ 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 M columns</td>
</tr>
<tr>
<td>$[C]$</td>
<td>L rows by L columns</td>
</tr>
<tr>
<td>$[X]$</td>
<td>M elements</td>
</tr>
</tbody>
</table>

(b) Matrices $[A]$ and $[B]$, and integers $L$, and $M$ should be defined.
(c) The product matrix (symmetric) is returned in matrix $[C]$. Vector $[X]$ is required in the subprogram for computations only.

2.34.3 Limitations:

2.34.4 Examples:

XABATC

-55-  XABATC
CALL XABATC(A,B,C,L,M,X)
CALL XABATC(A,B,C,L,L,X)

The examples below yield wrong results:

CALL XABATC(A,B,B,L,L,X)
CALL XABATC(A,B,A,L,L,X)
CALL XABATC(A,A,A,L,L,X)
2.35 SUBROUTINE XABTA(A,B,L,N,X)

2.35.1 Function:
The matrix product \([A][B]\) is formed in matrix \([A]\). Matrix \([B]\) is square. \(([A] = [A][B]^\top)\).

2.35.2 Calling Program:
(a) Matrices \([A]\), \([B]\), and vector \([X]\) have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>([A])</td>
<td>(L) rows by (N) columns</td>
</tr>
<tr>
<td>([B])</td>
<td>(N) rows by (N) columns</td>
</tr>
<tr>
<td>([X])</td>
<td>(N) elements</td>
</tr>
</tbody>
</table>

(b) Matrices \([A]\) and \([B]\), and integers \(L\), and \(N\) should be defined.
(c) The product matrix is returned in matrix \([A]\). Vector \([X]\) is required in the subprogram for computations only.

2.35.3 Limitations:
Matrix \([B]\) must be square. The original matrices \([A]\) and \([B]\) should be distinct from each other even if matrix \([A]\) is square. The original matrix \([A]\) is destroyed.

2.35.4 Examples:
CALL XAR(TA(A,B,L,N,X)
CALL XABTA(A,B,N,N,X)

The examples below yield wrong results:

CALL XABTA(A,A,N,N,X)
2.36 **SUBROUTINE XABTC(A,B,C,L,M,N)**

2.36.1 **Function:**
The matrix product \([A][B]^\top\) is formed in matrix \([C]\). \([C] = [A][B]^\top\)

2.36.2 **Calling Program:**
(a) 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>N rows by M columns</td>
</tr>
<tr>
<td>([C])</td>
<td>L rows by N columns</td>
</tr>
</tbody>
</table>

(b) Matrices \([A]\), and \([B]\) and integers \(L, M\) and \(N\) should be defined.
(c) The product matrix \([C]\) is defined in the subprogram.

2.36.3 **Limitations:**
Matrix \([C]\) should be distinct from matrices \([A]\) and \([B]\).

2.36.4 **Examples:**

```plaintext
CALL XABTC(A,B,C,L,M,N)
CALL XABTC(A,A,C,L,N,L)
CALL XABTC(A,A,C,L,L,L)
```

The examples below yield wrong results:

```
XABTC
   -59-
```
CALL XABTC\(A,B,A,L,M,M\)  
CALL XABTC\(A,B,B,L,L,L\)  
CALL XABTC\(A,A,A,L,L,L\)
2.37 SUBROUTINE XATBAC(A,B,C,L,M,X)

2.37.1 Function:
The matrix product \([A]^T [B] [A]\) is formed in matrix \([C]\). Matrix \([B]\) is symmetric. \(([C] = [A]^T [B] [A])\)

2.37.2 Calling Program:
(a) Matrices \([A]\), \([B]\) and \([C]\) and vector \([X]\) 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 (L) columns</td>
</tr>
<tr>
<td>([C])</td>
<td>(M) rows by (M) columns</td>
</tr>
<tr>
<td>([X])</td>
<td>(L) elements</td>
</tr>
</tbody>
</table>

(b) Matrices \([A]\) and \([B]\), and integers \(L\), and \(M\) should be defined.
(c) The product matrix \([C]\) (symmetric) is defined in the subprogram. Vector \([X]\) is required in the subprogram for computations only.

2.37.3 Limitations:
Matrix \([B]\) must be symmetric. Matrix \([C]\) should be distinct from matrices \([A]\) and \([B]\).

2.37.4 Examples:
CALL XATBAC(A,B,C,L,M,X)
CALL XATBAC(A,B,C,L,L,X)

The examples below yield wrong results:

CALL XATBAC(A,B,B,L,L,X)
CALL XATBAC(A,B,A,L,L,X)
CALL XATBAC(A,A,AL,L,X)
2.38 SUBROUTINE XATBB(A,B,L,N,X)

2.38.1 Function:
The matrix product \([A]' [B]\) is formed in matrix \([B]\). Matrix \([A]\) is square. \(([B] = [A]' [B])\)

2.38.2 Calling Program:
(a) Matrices \([A]\) and \([B]\), and vector \([X]\) have the following dimensions:

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>([A])</td>
<td>(L) rows by (L) columns</td>
</tr>
<tr>
<td>([B])</td>
<td>(L) rows by (N) columns</td>
</tr>
<tr>
<td>([X])</td>
<td>(L) elements</td>
</tr>
</tbody>
</table>

(b) Matrices \([A]\) and \([B]\), and integers \(L\), and \(N\) should be defined.

(c) The product matrix is returned in matrix \([B]\). Vector \([X]\) is required in the subprogram for computations only.

2.38.3 Limitations:
Matrix \([A]\) must be square. The original matrix \([B]\) is destroyed. The original matrices \([A]\) and \([B]\) should be distinct from each other even if matrix \([B]\) is square.

2.38.4 Examples:

XATBB
CALL XATBB(A,B,L,N,X)
CALL XATBB(A,B,N,N,X)

The example below yields wrong results:

CALL XATBB(A,A,N,N,X)
### 3. READY REFERENCE SHEET (FLMXPK)

<table>
<thead>
<tr>
<th>NO.</th>
<th>SUBPROGRAM</th>
<th>BRIEF DESCRIPTION</th>
<th>DIMENSIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADD(A,B,C,M,N)</td>
<td>C = A + B</td>
<td>A(M,N),B(M,N),C(M,N)</td>
</tr>
<tr>
<td>2</td>
<td>CORR(A,M,N,LABEL)</td>
<td>Edit A</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>3</td>
<td>DETMT(A,DA,N)</td>
<td>Determinant</td>
<td>A(N,N)</td>
</tr>
<tr>
<td>4</td>
<td>DIAG(A,DA,N)</td>
<td>Diagonal matrix</td>
<td>A(N,N)</td>
</tr>
<tr>
<td>5</td>
<td>EV(A,S,N)</td>
<td>E-values, E-vectors</td>
<td>A(N,N),S(N,N)</td>
</tr>
<tr>
<td>6</td>
<td>GEVP(A,B,S,T,N)</td>
<td>Solve AX = λBX</td>
<td>A(N,N),B(N,N),S(N,N),T(N)</td>
</tr>
<tr>
<td>7</td>
<td>IEV(A,S,N)</td>
<td>Improve on EV</td>
<td>A(N,N),S(N,N)</td>
</tr>
<tr>
<td>8</td>
<td>MINV(A,N,DET,NEXCH)</td>
<td>Invert A</td>
<td>A(N,N),NEXCH(N)</td>
</tr>
<tr>
<td>9</td>
<td>MOVE(A,B,M,N)</td>
<td>B = A</td>
<td>A(M,N),B(M,N)</td>
</tr>
<tr>
<td>10</td>
<td>MULT(A,B,C,L,M,N)</td>
<td>C = AB</td>
<td>A(L,M),B(M,N),C(L,N)</td>
</tr>
<tr>
<td>11</td>
<td>OUTE(A,I,J,TITLE,LUO)</td>
<td>Print A</td>
<td>A(I,J)</td>
</tr>
<tr>
<td>12</td>
<td>OUTE8(A,I,J,LABEL)</td>
<td>Display A</td>
<td>A(I,J)</td>
</tr>
<tr>
<td>13</td>
<td>OUTF(A,I,J,TITLE,LUO)</td>
<td>Print A</td>
<td>A(I,J)</td>
</tr>
<tr>
<td>14</td>
<td>OUTF8(A,I,J,LABEL)</td>
<td>Display A</td>
<td>A(I,J)</td>
</tr>
<tr>
<td>15</td>
<td>OUTG(A,I,J,TITLE,LUO)</td>
<td>Print A</td>
<td>A(I,J)</td>
</tr>
<tr>
<td>16</td>
<td>OUTG8(A,I,J,LABEL)</td>
<td>Display A</td>
<td>A(I,J)</td>
</tr>
<tr>
<td>17</td>
<td>PMULT(A,B,K,L,X)</td>
<td>A = AB</td>
<td>A(K,L),B(L,L),X(L)</td>
</tr>
<tr>
<td>18</td>
<td>POSTM(A,B,K,L,X)</td>
<td>B = AB</td>
<td>A(K,K),B(K,L),X(K)</td>
</tr>
<tr>
<td>19</td>
<td>RDCBC(A,M,N,LUI)</td>
<td>Read col. by col.</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>20</td>
<td>RDCBCCL(A,M,N)</td>
<td>Read col. by col. (LIST DIRECTED)</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>21</td>
<td>RDCOLG(A,M,N,LUI)</td>
<td>Read columns (grouped)</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>22</td>
<td>RDCOLGL(A,M,N)</td>
<td>Read columns (grouped) (LIST DIRECTED)</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>23</td>
<td>RDRBR(A,M,N,LUI)</td>
<td>Read row by row</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>24</td>
<td>RDRBRRL(A,M,N)</td>
<td>Read row by row (LIST DIRECTED)</td>
<td>A(M,N)</td>
</tr>
</tbody>
</table>
### 3. READY REFERENCE SHEET (FLMXPK) (continued)

<table>
<thead>
<tr>
<th>NO.</th>
<th>SUBPROGRAM</th>
<th>BRIEF DESCRIPTION</th>
<th>DIMENSIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.</td>
<td>RDROWG(A,M,N,LUI)</td>
<td>Read rows (grouped)</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>26.</td>
<td>RDRWGL(A,M,N)</td>
<td>Read rows (grouped)</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>27.</td>
<td>SCMUL(A,M,N,X)</td>
<td>A = xA</td>
<td>A(M,N)</td>
</tr>
<tr>
<td>28.</td>
<td>SINV(A,DA,N)</td>
<td>Sym. inversion</td>
<td>A(N,N)</td>
</tr>
<tr>
<td>29.</td>
<td>SOLVE(A,B,N,L,DET)</td>
<td>Sim. Eqs. (AX = B)</td>
<td>A(N,N),B(N,L)</td>
</tr>
<tr>
<td>30.</td>
<td>SQTR(A,N)</td>
<td>Transpose in place</td>
<td>A(N,N)</td>
</tr>
<tr>
<td>31.</td>
<td>SUB(A,B,C,M,N)</td>
<td>C = A - B</td>
<td>A(M,N),B(M,N),C(M,N)</td>
</tr>
<tr>
<td>32.</td>
<td>TMULT(A,B,C,L,M,N)</td>
<td>C = A'B</td>
<td>A(L,M),B(L,N),C(M,N)</td>
</tr>
<tr>
<td>33.</td>
<td>TRANS(A,B,M,N)</td>
<td>B = A'</td>
<td>A(M,N),B(N,M)</td>
</tr>
<tr>
<td>34.</td>
<td>XABATC(A,B,C,L,M,X)</td>
<td>C = ABA'</td>
<td>A(L,M),B(M,M),C(L,L),X(M)</td>
</tr>
<tr>
<td>35.</td>
<td>XABTA(A,B,L,N,X)</td>
<td>A = AB'</td>
<td>A(L,N),B(N,N),X(N)</td>
</tr>
<tr>
<td>36.</td>
<td>XABTC(A,B,C,L,M,N)</td>
<td>C = AB'</td>
<td>A(L,M),B(N,M),C(L,N)</td>
</tr>
<tr>
<td>37.</td>
<td>XATBAC(A,B,C,L,M,X)</td>
<td>C = A'BA</td>
<td>A(L,M),B(L,L),C(M,M),X(L)</td>
</tr>
<tr>
<td>38.</td>
<td>XATBB(A,B,L,N,X)</td>
<td>B = A'B</td>
<td>A(L,L),B(L,N),X(L)</td>
</tr>
</tbody>
</table>

**NOTE:** \( [A]' = [A]^T \)