NBER WORKING PAPER SERIES

WP 103

ROSEPACK Document No. 1

Semi-portability of FORTRAN Programs

Neil E. Kaden*
Virginia Klema*

Working Paper No. 103

Computer Research Center for Economics and Management Science
National Bureau of Economic Research, Inc.
575 Technology Square
Cambridge, Massachusetts 02139

September 1975

*NBER Computer Research Center. Research supported in part by National Science Foundation Grant #DCR 75-08802 to the National Bureau of Economic Research, Inc.

Table of Contents

Abstract

Transferring Fortran subroutines from one manufacturer's machine to another or from one operating system to another puts certain constrains on the construction of the Fortran statements that are used in the subroutines. The reliable performance of this mathematical software should be unaffected by the host environment in which the software is used or by the compiler from which the code is generated. In short, the algorithm is to be independent of the computing environment in which it is run.

The subroutines of ROSEPACK (Robust Statistics Estimation Package) are Fortran IV source code designed to be semi-portable where semi-portable is defined to mean transportable with minimum change.*

Acknowledgments

The authors wish to thank J. Boyle and W. Cody for sharing their internal document on programming conventions, J. Kirsch for his helpful suggestions on documentation of subroutines, D. Hoaglin and G. Ruderman for their constructive criticism of this document, and Sheila Howard for her careful typing of the manuscript.

*Cody, W.J., "The Construction of Numerical Subroutine Libraries," SIAM Review, Vol. 16, No. 1, pp. 36-46, January (1974).

Semi-portability of Fortran subroutines puts certain constraints on the construction of Fortran statements, the declaration of variables, and the representation of constants that are used in the subroutines. Many of these constraints are needed for Fortran subroutines that are to be imbedded in applications subsystems that are written in another language, say PL/1.

Inevitably, the numerical algebra algorithms themselves are strengthened when their performance is unaffected by the arithmetic of the machine on which they are used and the Fortran compiler by which their code is generated.

The rules for structured programming [1,2,3], and structured documentation (see Part Two, Section X of this document) should be followed insofar as possible. The comments within the program or subroutine should be sufficient to inform the user about input parameters, output parameters, temporary storage parameters, error exits, and the algorithm that the program implements.

This document presents certain suggestions for programming that will tend toward requirements for semi-portability of ANSI Fortran IV (as described in CACM, Vol. 7, No. 10, October '64) subroutines and programs. We also suggest certain conventions for comments and general formatting of the Fortran code. By "formatting" we mean the spacing and indentation that determine the general appearance and readability of the code. Such formatting is suggested to help the reader or the user understand the algorithm, the program, and the flow of control within the program.

Part One

The suggestions for programming are

I. CØMMØN storage should not be used for arrays. This is not an ANSI restriction, but driver programs become simpler to write, and the use in a paged environment is enhanced if one does not use CØMMØN.

II. All array arguments should have adjustable dimensions. These dimensions should be made explicit in the declarations of the formal parameters for each subroutine. For example,

```
REAL   A(NM,N)
```

not

```
REAL   A(NM,1)
```

III. EQUIVALENCE statements should not be used.

IV. Certain Fortran compilers do not distinguish more than six characters of an identifier. Hold identifiers to six characters or fewer.

V. Do not use multiple entry points or non-standard returns.

VI. Be sure that the precision of any Fortran library routine or built-in function is explicit in all statements. For example, DABS, not ABS, for absolute value for long precision computing. Do not use mixed mode arithmetic or assume there is an implied conversion anywhere, not even for constants. For example

```
DØUBLE PRECISION X
X = X + 10.0D0
```

not

```
DØUBLE PRECISION X
X = X + 10.0
```

VII. Constants that are used in iterations or convergence criteria should be functions of the machine's precision, i.e., the smallest floating point number, ε, representable in the machine for which the floating representation of $1+\varepsilon>1$. Certainly, a constant that cannot be converted precisely on the machine should never be used. For example, .1, is representative of such numbers.

VIII. Test cases must be devised so that data can be converted uniformly on all target machine. The word length of the machine determines the truncation of the internal representation of floating point numbers, and conversion routines do not treat floating point numbers uniformly. The integers are treated uniformly with respect to conversion so long as they lie within the precision range of arithmetic of the computing machine. One suggestion for portability for test cases used as input numbers is to read them in as integers and then DFLOAT to get the floating point representations.

IX. Obscure underflows can often produce side effects that give divide checks or overflows. This problem is particularly acute because the range of arithmetic on many machines is not symmetric about zero. For example, the range of arithmetic on the IBM 360/370 machines is about $10^{75} > |x| > 10^{-78}$. That an algorithm will exhibit overflow, underflow, or divide check problems is often not known in advance. However, some linear systems routines have this problem when an inner product is formed or when multiple divides are encountered. Be prepared to isolate such problems.

One solution is to reorder arithmetic expression. Another solution is to resort to extended precision arithmetic for those critical sections of code.

X. The usual rules for separate sections for error handling and input-output that are required for applications subsystems are equally applicable for semi-portable Fortran programs. The error handling from the subroutines should be uniform. The input-output should be confined to main programs or special I-Ø subroutines; that is to say, computation subroutines should be I-Ø free. The goal of the error recovery is to permit computation to continue without resorting to system termination.

XI. We expect the subroutines in ROSEPACK to be compiled with Fortran compilers with the highest level of optimization. We have not used hand optimization in the subroutines

<u>Part Two</u>

The suggestions for formatting are

I. Identifiers

Identifiers, i.e., variable names, should correspond to default declarations in Fortran. However, explicit declarations should be written for each identifier.

Variables from the calling sequence, internal variables, and function names should all be declared separately. For a suggestion on how to accomplish this see X Internal Documentation Section B (PARAMETERS), Section C (LOCAL VARIABLES), and Section D (FUNCTIONS) of the description of the Prologue.

II. Labels

If the order of labels within a subroutine is not linear the convention used should be explicitly described. This ordering of statement labels should be linear and could proceed in multiples of 10 for interior program sections. The next level of program section could proceed as 100, and perhaps the next as 1000.

Do not use unreferenced labels.

Code for error exits should be surrounded by comments and located at the end of the program or subroutine. The labels for error exits should be 2 or 3 digits, the first of which is 9, the last non-zero.

One format statement may be used by more than one print statement in a program. Therefore we suggest that all format statements be labeled with 4 digit numbers the last of which is non-zero and placed after RETURN and before END of the program.

Preferably all input-output should be written in subroutine form. We suggest that a DATA statement be used to fix units of I-Ø and that this DATA statement be made particular to a given installation.

The variables containing this I-Ø unit information should be passed as parameters to all subroutines using these I-Ø units. This device allows a global change of an I-Ø unit without recompiling individual subroutines.

III. Use of blank spaces

There should not be extra blank spaces around dummy variables or constants in DO loops. Blank spaces should delimit = symbols in assignment statements. Blank spaces should be used wherever such use will enhance readability of elements of expressions or statements.

IV. Tab Spacings

Throughout this document we are assuming tab setting in columns 1, 7, 10, 15, 20, 25, etc.

V. Continuation Characters

Second and subsequent lines of all continuation statements should be numbered 1 through 9, then A through Z in column 6. The text of each continuation statement should be indented one tab space from the initial line of the statement.

VI. DØ loops

All DØ loops should be surrounded by comment statements which may be blank. Text comments should follow a blank comment statement. If more than one statement is in the range of a DO loop, the closing statement of the DO loop should be a CONTINUE. This CONTINUE should be unambiguous. Statements between DO and CONTINUE should be indented one additional tab space to correspond to a block structure.

Inner loops should be indented one tab space to the right of their surrounding outer loop.

For examples of indentation of DO loops see the examples in Appendix I.

VII. DATA Statements

Data statements should be used to set installation-dependent constants, such as data-set numbers for I/O, and machine precisions, underflow tolerances, or other machine-dependent constants. See X, Internal Documetnation, Section L, for more details.

If a non-numeric character string must be used in a DATA statement, it should be packed as one character per machine word and always stored in an array.

VIII. Structured programming

The programming and formatting conventions that we describe are similar to structured programming in the following ways:

1. format for readability and understanding
2. indentation for major and minor loops
3. array dimensions are adjustable
4. temporary storage arrays are passed as parameters
5. documentation is structured such that it is contained within the routine.

IX. Printed output

All printed output should be formatted such that it is not greater than 8 1/2 inches in width. Most line printers print 10 characters per inch, and 80 characters per line allows ample margins. This will greatly aid in reproduction of the output.

X. Internal Documentation

All Fortran programs should be well documented by liberal use of comment statements. Proper documentation will enhance readability and appearance of the code, improve understanding of the algorithm used, help ensure proper use of the program, and aid in future modifications. When semi-portability is also considered, proper documentation serves to isolate those portions of the code which are installation-dependent.

In-line documentation of Fortran programs can be considered in two major sections, the Prologue and the Program-flow comments. The latter consists of the comment statements embedded within the code describing how the algorithm is being carried out as the flow of control passes from statement to statement. The former consists of certain non-executable FORTRAN statements found at the beginning of the subprogram which fully describe the proper use of the software, as well as information concerning its development. Any user familiar with the guidelines has the added advantage of knowing where to find specific information concerning the program. The Prologue also identifies and isolates installation-dependent aspects of the program and thus enhances semi-portability. The Prologue is the major documentation for the use of the program subroutine.

Program-flow comments should be delimited by special characters to enhance their readability and appearance. In ROSEPACK the colon (:) is used. Such comments should also follow the rules for statement indentation described elsewhere in this document.

In most cases, the text of the comment should be preceded and followed by a string of 10 special characters (colons). At least one blank space, but not more than three blank spaces, should be put between the special character strings and the text of the comment. If this method is used for a comment extending over several lines, all lines should have a "C" in column 1, and all but the first line should be indented one additional tab (beyond the current level of indentation).

Building on the suggestions of Boyle and Cody, an alternative method of delimiting comments, recommended for important comments or those extending over several lines, is to surround them by a "box" of special characters. The following is an example of what is meant by a "box":

```
C           ::::::::::::::::::::::::
C           ::                    ::
C           :: THIS IS A COMMENT  ::
C           ::                    ::
C           ::::::::::::::::::::::::
```

Blank comment statements (i.e., comment statements containing blanks in columns 2 through 72) may be used wherever their use enhances the readability of the program.

The statement immediately before the END statement should always be a comment statement delimiting the end of the program and containing the name of the program. An example follows:

```
            RETURN
C           :::::::::::   LAST CARD OF (NAME OF SUBROUTINE) :::::::::::
            END
```

The Prologue consists of the declarations of the calling sequence and variable names of the subprogram, a number of sections of text on the subprogram, and any DATA or EQUIVALENCE statements. It contains a number of headings denoting the different logical sections of the Prologue. The headings are comment statements with the character "*" in columns 7 through 11 and the heading name beginning in column 12 and followed by a colon (:). A blank comment statement should not immediately follow a heading. If the section denoted by the heading line is empty, the heading should be followed by a comment statement containing "NONE" in columns 7 through 10 and then a blank comment statement.

The different headings, in the order they should appear, are:

-PARAMETERS
-LOCAL VARIABLES
-FUNCTIONS
-PURPOSE
-PARAMETER DESCRIPTION
-APPLICATION AND USAGE RESTRICTIONS
-ALGORITHM NOTES
-REFERENCES
-HISTORY
-GENERAL
-BODY OF PROGRAM

Three delimiter lines, consisting of a blank comment statement, a comment statement consisting of the special character colon (:) in columns 7 through 72, and then a second blank comment statement, should occur immeidately before the purpose heading and immediately after the GENERAL section. All lines between these delimiters should be comment statements. When columns 73 through 80 of each card contain serialization or identification characters, this gives a box-like appearance to the part of the Prologue containing text.

An example of two programs following these guidelines is in Appendix I.

What follows is a brief description of each section of the Prologue. Note that no blank comment statements should occur until after the FUNCTIONS section.

A. CALLING SEQUENCE

The SUBROUTINE or FUNCTION statement should be the first line of the subprogram. Blanks should be used to enhance readability.

B. PARAMETERS

Declaration statements should be grouped by type, i.e., first INTEGER, then REAL, then DØUBLE PRECISION, or REAL*8, then REAL*16, then CØMPLEX, CØMPLEX*32, then LØGICAL. Within each type grouping, variable names should be listed in the order they occur in the calling sequence. By parameters, we mean all of the variable names appearing in the calling sequence.

C. LOCAL VARIABLES

As with the preceding section, declaration statements are grouped by type, and in the same order. Within each type, variable names should be listed alphabetically.

<u>ALL</u> variables used in the program should be explicitly declared.

D. <u>FUNCTIONS</u>

All functions called by the program should be explicitly declared. Declaration statements are grouped by type.

E. PURPOSE

Briefly describe the purpose of this subprogram. Give references when necessary. More detail can be given in later sections.

F. PARAMETER DESCRIPTION

This section contains 3 subsections. The first describes input parameters, the second describes output parameters, and the third subsection describes parameters used for temporary storage by the subprogram. If the contents of any parameter variable can be changed by the subprogram, it should be considered an output parameter. See the examples in Appendix I for the format, keywords, punctuation and indentation used in this section.

G. APPLICATION AND USAGE RESTRICTIONS

If any other programs in this package can call this subprogram, or are called by it, they should be described here. If this subprogram is part of a group of programs which are called in some specified order, this should also be included. Give references except when a reference is implicit, as with another member of the same package.

Also included in this section are any warnings about special cases or possible errors which can occur if there are errors in the subprogram call. Warnings about misuse of tolerance parameters belong here. The entry in PARAMETER DESCRIPTION should refer the reader to this section where applicable.

H. ALGORITHM NOTES

Anything special about the algorithm used or its implementation should be listed here. Any special conventions regarding statement labeling or commenting should be mentioned. If there is anything special about error handling which has not yet been mentioned, it should be described here.

I. REFERENCES

References from elsewhere in the documentation, as well as any other references pertaining to the subprogram, should be listed.

J. HISTORY

The author of this subprogram, as well as the date and place of origin should be listed. If the subprogram is a translation of a program in another language or is based on another program, a reference should be given. If the program has been modified since it was written, the date and person making the modification should be noted. If this subprogram has been released as part of a subroutine library, the current release date of the library should be given.

K. GENERAL

If this subprogram was developed under research supported by a grant requiring acknowledgment, the required information should occur here. The person to contact concerning comments and problems with the subprogram should have his address in this section.

L. DATA and EQUIVALANCE Statements

Following the second occurrence of delimiting comment statements (a blank comment statement, a comment statement with colons in columns 7 through 72, and a second blank comment statement) is where all DATA and EQUIVALENCE statements should occur. If a DATA statement contains an installation-dependent constant, comment statements explaining its value and mentioning the installation's designation, should precede the DATA statement. Those comment statements should conform to the standards of program-flow comments.

All DATA statements should precede any EQUIVALENCE staements.

M. BODY OF PROGRAM

This heading denotes the end of the Prologue and the beginning of the program body.

References

[1] Dahl, O.H, Dijkstra, E.W., Hoare, C.A.R., Structured Programming, Academic Press, (1972).

[2] Kernighan, B.W., Plauger, P.J., "Programming Style: Examples and Counter-Examples," ACM Computing Surveys, Vol 6, No. 4, pp. 303-319, December, (1974).

[3] Kernighan, B.W., Plauger, P.J., "The Elements of Programming Style," Bell Telephone Laboratories (1974).

Appendix

This appendix contains listings of two subroutines that are samples of candidates for inclusion in ROSEPACK. The reader is reminded that we are relying on Fortran compiler optimization of sub-expressions within loops.

```
      SUBROUTINE MINSOL(NM,N,V,W,IP,B,RKTOL,IERR,RV1)
C
C     *****PARAMETERS:
      INTEGER NM,N,IP,IERR
      REAL*8  V(NM,N),W(N),B(NM,IP),RKTOL,RV1(N)
C     *****LOCAL VARIABLES:
      INTEGER I,J,K
      REAL*8  RKTOL1,X,Z
C     *****FUNCTIONS:
C     NONE
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE DETERMINES A CANDIDATE SOLUTION TO THE LINEAR
C                                                               T
C     SYSTEM AX=B, AFTER THE SINGULAR VALUE DECOMPOSITION A=USV  OF A
C                                                 T
C     REAL M BY N RECTANGULAR MATRIX, FORMING U B RATHER THAN U,HAS
C     ALREADY BEEN PERFORMED.  THIS CANDIDATE SOLUTION IS BASED ON THE
C     RANK TOLERANCE PARAMETER, RKTOL, OR THE DEFAULT, 2.0D0**(-26),
C     WHICH IS THE SQUARE ROOT OF THE MACHINE PRECISION 2.0D0**(-52).
C
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NM MUST BE SET TO THE ROW DIMENSION OF THE TWO DIMENSIONAL
C             ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C             DIMENSION STATEMENT;
C
C        N IS THE NUMBER OF ROWS OF B, AND THE ORDER OF V;
C
C        V CONTAINS THE SQUARE MATRIX V (ORTHOGANAL) OF THE SINGULAR
C             VALUE DECOMPOSITION;
C
C        W CONTAINS THE N (NON-NEGATIVE) SINGULAR VALUES OF A (THE
C             DIAGONAL ELEMENTS OF S). THEY ARE UNORDERED;
C
C        IP IS THE NUMBER OF COLUMNS OF B;
C
C                                              T
C        B CONTAINS THE RECTANGULAR MATRIX U B;
C
C        RKTOL IS THE RANK TOLERANCE WHICH WILL BE USED.  IF RKTOL IS
C             NOT POSITIVE, THEN THE DEFAULT WILL BE USED.
C
C
C     ON OUTPUT:
C
C        V REMAINS UNCHANGED;
C
C        W CONTAINS THE PSEUDOINVERSE OF THE DIAGONAL MATRIX S.
C             ANY SINGULAR VALUES THAT ARE LESS THAN RKTOL TIMES THE
C             LARGEST SINGULAR VALUE ARE SET TO ZERO IN THE PSEUDO-
C             INVERSE;
C
C        B HAS BEEN OVERWRITTEN BY THE SOLUTION X;
C
C        IERR IS SET TO
C             ZERO FOR NORMAL RETURN,
C             -1   IF THE MAXIMUM SINGULAR VALUE IS ZERO (INDICATING
C                  A ZERO A-MATRIX IN THE SINGULAR VALUE
C                  DECOMPOSITION).
C
```

```
C
C     RV1 IS A TEMPORARY STARAGE ARRAY.
C
C     *****APPLICATION AND USAGE RESTRICTIONS:
C     IT IS RECOMMENDED THAT THE SUBROUTINE MINFIT (1) PERFORM THE
C     SINGULAR VALUE DECOMPOSITION.
C     THE IERR PARAMETER SHOULD BE CHECKED BEFORE CALLING MINSOL.
C
C     SETTING THE RANK TOLERANCE SHOULD ONLY BE DONE IF THE USER KNOWS
C     THE SINGULAR VALUES OF THE A-MATRIX WITH RESPECT TO THE CERTAINTY
C     OF THE DATA.
C
C     *****ALGORITHM NOTES:
C     NONE
C
C     *****REFERENCES:
C     (1) ARGONNE NATIONAL LAB., FORTRAN SUBROUTINE MINFIT, ANLF233S.
C     (2) BECKER,R.,KADEN,N., AND KLEMA,V., 'THE SINGULAR VALUE
C        ANALYSIS IN MATRIX COMPUTATION', NBER WORKING PAPER NO. 46,
C        (JULY 1974)
C     (3) GOLUB,G.H., AND REINSCH,C.,'SINGULAR VALUE DECOMPOSITION
C        AND LEAST SQUARES SOLUTIONS',IN J.H. WILKINSON AND C. REINSCH
C        (EDS.) HANDBOOK FOR AUTOMATIC COMPUTATION,VOLUME II:LINEAR
C        ALGEBRA, SPRINGER VERLAG,134-151 (1971); PREPUBLISHED IN
C        NUMER. MATH. 14,403-420 (1970).
C
C     *****HISTORY:
C     MINSOL IS BASED ON CODE WRITTEN BY FRED CIARAMAGLIA (NBER/COMPUTER
C     RESEARCH CENTER; MAY 4, 1973).
C
C     ADAPTED BY NEIL KADEN (NBER/COMPUTER RESEARCH CENTER) JUNE 9,1975.
C
C     DATE LAST MODIFIED: JUNE 11, 1975.
C
C     *****GENERAL:
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO:
C        SUPPORT STAFF MANAGER
C        COMPUTER RESEARCH CENTER FOR ECONOMICS AND MANAGEMENT SCIENCE
C        NATIONAL BUREAU OF ECONOMIC RESEARCH
C        575 TECHNOLOGY SQUARE
C        CAMBRIDGE, MASSACHUSETTS 02139.
C
C     DEVELOPMENT OF THIS PROGRAM SUPPORTED IN PART BY
C     NATIONAL SCIENCE FOUNDATION GRANT GJ-1154X3 AND
C     NATIONAL SCIENCE FOUNDATION GRANT DCR75-08802
C     TO NATIONAL BUREAU OF ECONOMIC RESEARCH, INC.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     :::::::::: RKTOL1, FOR THESE APPLICATIONS, IS A MACHINE DEPENDENT
C        PARAMETER BASED ON THE SQUARE ROOT OF THE RELATIVE PRECISION
C        OF FLOATING POINT ARITHMATIC.
C        MACHEP = 16.0D0**(-13) FOR DOUBLE PRECISION ARITHMETIC
C        ON S360 AND S370. ::::::::::
      DATA RKTOL1/Z3A40000000000000/
C
C     *****BODY OF PROGRAM:
      IERR = 0
      IF (RKTOL .LE. 0.0D0)  RKTOL = RKTOL1
C     :::::::::: FIND MAXIMUM ELEMENT OF W ::::::::::
      Z = 0.0D0
C
      DO 750 J = 1, N
         X = W(J)
         IF (X .LE. Z) GO TO 750
         Z = X
  750 CONTINUE
C
```

```
      IF (Z .EQ. 0) GO TO 999
C     :::::::::: FORM PSEUDO INVERSE OF DIAG(W) ::::::::::
      DO 800 J = 1, N
         X = W(J) / Z
         IF (X .LE. RKTOL) GO TO 790
         W(J) = 1.0D0 / W(J)
         GO TO 800
  790    W(J) = 0.0D0
  800 CONTINUE
C     :::::::::: FORM X (RETURNED IN B) ::::::::::
      DO 900 J = 1, IP
C
         DO 810 I = 1, N
            RV1(I) = W(I) * B(I,J)
  810    CONTINUE
C
         DO 890 I = 1, N
            X = 0.0D0
C
            DO 850 K = 1, N
               X = X + V(I,K) * RV1(K)
  850       CONTINUE
C
            B(I,J) = X
  890    CONTINUE
C
  900 CONTINUE
C
      GO TO 1000
C     :::::::::: ERROR IF MAX SINGULAR VALUE = 0 ::::::::::
  999 IERR = -1
C     :::::::::: RETURN TO CALLING PROGRAM ::::::::::
 1000 RETURN
C     :::::::::: LAST CARD OF MINSOL ::::::::::
      END
```

```
      SUBROUTINE LUDCMP(MN,M,N,A,IHTTC,ICOL,IROW,IERR,ICMAX)
C     *****PARAMETERS:
      INTEGER MN,M,N,IHTTC(N),ICOL(N),IROW(M),IERR,ICMAX(N)
      REAL*8  A(MN,N)
C     *****LOCAL VARIABLES:
      INTEGER I,IMIN,IPIVOT,ITEMP,J,MAXCOL
      REAL*8  AMAX,RATIO,TEMP
C     *****FUNCTIONS:
      INTEGER MINO
      REAL*8  DABS
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE DOES AN LU DECOMPOSITION ON THE REAL M BY N
C        RECTANGULAR MATRIX A, WITH MODIFIED COMPLETE PIVOTING, AND
C        RETURNS BOTH THE STRICT LOWER TRIANGLE OF L AND THE FULL
C        UPPER TRIANGLE OF U.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NM MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL
C             ARRAY PARAMETERS AS DECLARED IN THE CALLIMG PROGRAM
C             DIMENSION STATEMENT;
C
C        M MUST BE SET TO THE NUMBER OF ROWS IN THE MATRIX;
C
C        N MUST BE SET TO THE NUMBER OF COLUMNS IN THE MATRIX;
C
C        A CONTAINS THE MATRIX TO BE FACTORED BY THE LU-DECOMPOSITION
C             WITH COMPLETE PIVOTING;
C
C        IHTTC IS SET SUCH THAT IF IHTTC(J) IS LESS THAN ZERO THEN THE
C             J-TH COLUMN OF THE INPUT MATRIX CANNOT BE PIVOTED INTO
C             THE FIRST N COLUMNS OF ITS LU-DECOMPOSITION. ALL ELEMENTS
C             OF THIS VECTOR SHOULD BE SET TO ZERO TO INSURE NORMAL
C             PIVOTING;
C
C      ON OUTPUT:
C
C        A CONTAINS THE L-MATRIX AND THE U-MATRIX AS FOLLOWS:
C             A( ROW(I),COL(J) ) FOR I LESS THAN OR EQUAL TO J CONTAINS
C             THE (I,J) ELEMENTS OF THE UPPER TRIANGULAR U-MATRIX;
C             A( ROW(I),COL(J) ) FOR I GREATER THAN J CONTAINS THE
C             SUB-DIAGONAL (I,J) ELEMENTS OF THE L-MATRIX. THE DIAGONAL
C             OF THE L-MATRIX CONTAINS ALL ONES;
C
C        IROW REFLECTS THE ROW PIVOTING PERFORMED. IF IROW(J) IS
C             EQUAL TO K THEN THE J-TH ROW WAS PIVOTED INTO THE K-TH
C             ROW-POSITION.  SEE ALSO THE OUTPUT DESCRIPTION OF A;
C
C        ICOL REFLECTS THE COLUMN PIVOTING PERFORMED. SEE ALSO THE
C             OUTPUT DESCRIPTION OF IROW;
C
C        IERR IS SET TO ZERO FOR NORMAL EXITS.
C             IT IS SET TO K, WHERE K IS AN INTEGER DENOTING THE
C             CURRENT ITERATION, IF NO ACCEPTABLE PIVOTS COULD BE FOUND
C             (AN ACCEPTABLE PIVOT IS A NON-ZERO ONE WITH THE
C             CORRESPONDING ELEMENT OF IHTTC NON-NEGATIVE).
C             IF AN ERROR EXIT IS TAKEN, THE DECOMPOSITION HAS ONLY
C             BEEN PERFORMED FOR K-1 ITERATIONS.
C
```

```
C     ICMAX IS USED FOR TEMPORARY STORAGE BY THE SUBROUTINE.
C
C     *****APPLICATION AND USAGE RESTRICTIONS:
C     LUDCMP CAN BE USED IN SOLVING A LINEAR SYSTEM AX=B.
C
C     SPECIAL CARE SHOULD BE EXERCISED IN THE USE OF THE PARAMETER
C     IHTTC.
C
C     *****ALGORITHM NOTES:
C     LUDCMP USES INDIRECTION IN FORMING THE L-MATRIX AND THE U-MATRIX
C     TO AVOID ACTUALLY INTERCHANGING ROWS AND COLUMNS IN MAIN STORAGE.
C
C     *****REFERENCES:
C     NONE
C
C     *****HISTORY:
C     WRITTEN BY NEIL KADEN (NBER/COMPUTER RESEARCH CENTER) JULY 31,
C     1974.
C
C     DATE LAST MODIFIED: JUNE 17, 1975.
C
C     *****GENERAL:
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO:
C        SUPPORT STAFF MANAGER
C        COMPUTER RESEARCH CENTER FOR ECONOMICS AND MANAGEMENT SCIENCE
C        NATIONAL BUREAU OF ECONOMIC RESEARCH
C        575 TECHNOLOGY SQUARE
C        CAMBRIDGE, MASS. 02139.
C
C     DEVELOPMENT OF THIS PROGRAM SUPPORTED IN PART BY
C     NATIONAL SCIENCE FOUNDATION GRANT GJ-1154X3
C     TO NATIONAL BUREAU OF ECONOMIC RESEARCH, INC.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****BODY OF PROGRAM:
      IMIN = MINO(M,N)
      IERR = 0
C
      DO 10 I=1,M
         IROW(I) = I
   10 CONTINUE
C
C
      DO 20 I=1,N
         ICOL(I) = I
   20 CONTINUE
C
C     ::::::::::BEGINNING OF OUTER LOOP::::::::::
      DO 110 IPIVOT=1,IMIN
C
         DO 40 J=IPIVOT,N
              ICMAX(J) = IPIVOT
C
              DO 30 I=IPIVOT,M
                   IF (DABS(A(IROW(I),ICOL(J))) .LE. DABS(A(IROW(
     1                  ICMAX(J)),ICOL(J))))  GO TO 30
                   ICMAX(J) = I
   30         CONTINUE
C
   40    CONTINUE
C
```

```
      MAXCOL = 0
      AMAX = 0.0D0
C     ::::::::::FIND PIVOT::::::::::
      DO 50 J=IPIVOT,N
           IF (IHTTC(ICOL(J)) .NE. 0)  GO TO 50
           TEMP = DABS(A(IROW(ICMAX(J)),ICOL(J)))
           IF (TEMP .LE. AMAX)  GO TO 50
           AMAX = TEMP
           MAXCOL = J
   50 CONTINUE
C
      IF (MAXCOL .EQ. 0)  GO TO 99
      IF (MAXCOL .EQ. IPIVOT) GO TO 60
C     ::::::::::COLUMN EXCHANGE::::::::::
      ITEMP = ICOL(IPIVOT)
      ICOL(IPIVOT) = ICOL(MAXCOL)
      ICOL(MAXCOL) = ITEMP
   60 CONTINUE
      IF (ICMAX(MAXCOL) .EQ. IPIVOT)  GO TO 70
C     ::::::::::ROW EXCHANGE::::::::::
      ITEMP = IROW(IPIVOT)
      IROW(IPIVOT) = IROW(ICMAX(MAXCOL))
      IROW(ICMAX(MAXCOL)) = ITEMP
   70 CONTINUE
C     ::::::::::PIVOTING ACCOMPLISHED::::::::::
      IF (IPIVOT .EQ. M)  GO TO 110
      ITEMP = IPIVOT + 1
C
      DO 90 I=ITEMP,M
           RATIO = A(IROW(I),ICOL(IPIVOT)) / A(IROW(IPIVOT),ICOL(
     1          IPIVOT))
           A(IROW(I),ICOL(IPIVOT)) = RATIO
           IF (IPIVOT .EQ. N)  GO TO 90
C
           DO 80 J=ITEMP,N
                A(IROW(I),ICOL(J)) = A(IROW(I),ICOL(J))-RATIO*A(
     1                    IROW(IPIVOT),ICOL(J))
   80      CONTINUE
C
   90 CONTINUE
C
  110 CONTINUE
C     ::::::::::END OF OUTER LOOP::::::::::
      GO TO 100
C     ::::::::::NO NON-ZERO PIVOT FOUND::::::::::
   99 IERR = IPIVOT
C
  100 RETURN
C     ::::::::::LAST CARD OF LUDCMP::::::::::
      END
```