NASA TECHNICAL MEMORANDUM

NASA TM X-1749

NASA TM X-1749

A CONVERSATIONAL APPROACH TO MATRIX CALCULATIONS — CONCEPTS AND IMPLEMENTATION

by R. Bruce Canright, Jr.

Lewis Research Center

Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • FEBRUARY 1969

NASA TM X-1749

A CONVERSATIONAL APPROACH TO MATRIX CALCULATIONS - CONCEPTS AND IMPLEMENTATION

By R. Bruce Canright, Jr.

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – CFSTI price $3.00

ABSTRACT

The conversational use of computers makes user-oriented languages desirable. A computer program has been developed which enables users to do matrix calculations conversationally, without knowledge of computers or programming languages. The program converses in a brief command language, which provides for input/output, matrix arithmetic, forming of the transpose, inverse, determinant, and eigenvalues, and exponentiation. Details of the program language and logic are given, with a sample user-program conversation. Technical details and FORTRAN program listings are also presented.

A CONVERSATIONAL APPROACH TO MATRIX CALCULATIONS - CONCEPTS AND IMPLEMENTATION

by R. Bruce Canright, Jr.

Lewis Research Center

SUMMARY

Calculations with matrices often require the aid of computers. The ability to do such calculations while conversing with the computer frees users from doing any programming to perform everyday calculations, and allows easy checking of results. For such an application a user-oriented language, not a computer language, is desirable.

A computer program has been written which enables users to do matrix computations without knowledge of computers or programming languages. Potential users can easily learn the command language recognized by the program. The program, called MATAR, interacts with users in a conversational manner. It is now running under a time-sharing system.

MATAR provides for simple input/output of matrices, matrix arithmetic, and several other operations. These include formation of the transpose, inverse, determinant, and eigenvalues and exponentiation.

The limitations of the program are discussed, with some remarks on the logic it uses. A complete and brief user's guide is provided, with a sample user-program conversation carried on through a typewriter-like terminal. Technical details and FORTRAN program listings are also presented.

INTRODUCTION

In the last few years, the increasing capabilities of electronic computers have made possible the numerical solutions of complex problems. These problems could not practically be solved without computers because of the large amounts of arithmetic involved.

But computers must be told exactly what to do in their own languages. Unfortunately, this often creates a barrier between the problem solver and his solution. He is forced either to master a computer language or to work with a computer programmer who translates his problem for the computer. However, there are some problems which should not require this; that is, some problems (e.g., matrix calculations) are just hard enough that they require the help of a computer, but not so hard that the problem solver should have to translate them into computer language.

One approach to breaking down this barrier is to develop more capable desk calculators which remain easy to use. Desk calculators are not suited to matrix manipulations, however, because of their storage limitations.

Another approach is to make computers available for conversational use. By this is meant real-time interactions between computers and users such as programmers, businessmen, or engineers. The conversational approach is practical especially where time-sharing systems are available; under such systems, computers handle more than one user essentially as if each user had the computer to himself.

The conversational approach can be made easy by developing computer programs which are strongly user-oriented. For matrix calculations, the user need only know how to get matrices in, how to operate on them, and how to get back the answers, all by means, for example, of short lines typed at a typewriter-like terminal. This report shows a way of doing matrix calculations on a computer by conversing with the computer in a brief set of commands.

MATRIX CALCULATIONS - AN EXAMPLE PROGRAM

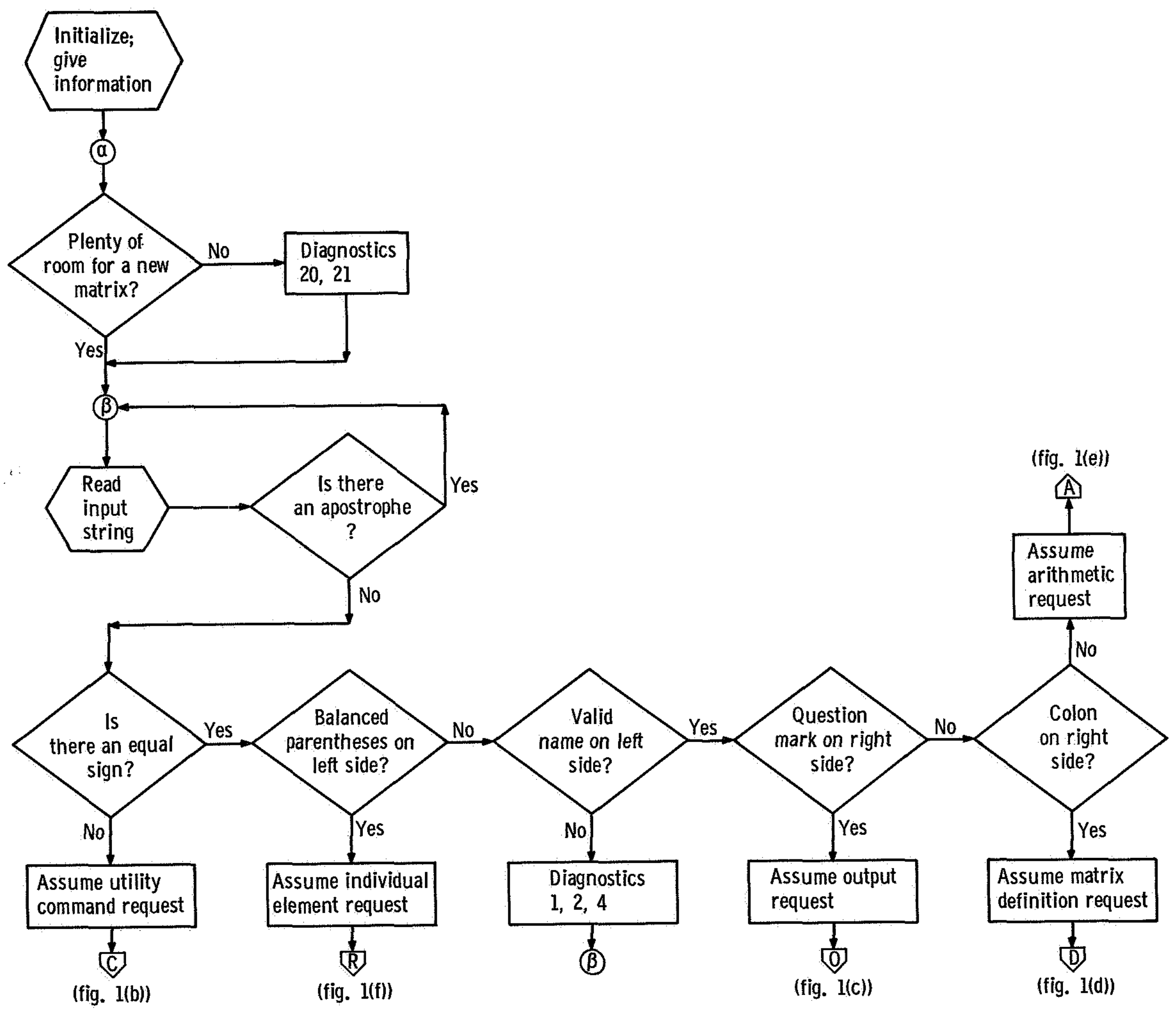
Capabilities of Program

An unusual feature of MATAR is its power to help users learn the proper usages of MATAR. The program can issue specific complaints about input it cannot process. For example, if the user tries to use a matrix not previously defined, the program tells him that the matrix is unknown and therefore that particular calculation must be canceled. The user can promptly see and rectify his errors.

The program will add, subtract and multiply two matrices, and raise a matrix to a positive integer power. In addition, it will form the inverse, determinant, eigenvalues, and transpose of user-defined matrices on command. Mixed scalar/matrix arithmetic is permitted. The inverse, determinant, and eigenvalues are calculated only for square matrices.

The program stores matrices by names the user chooses and allows him to use these names in simple expressions. The user defines a matrix by giving its name and a punctuated sequence of numbers for its elements, or by defining it in terms of previously defined matrices. The user can quickly display any matrix by asking for it by name. Also, individual elements can be called out or altered. This storage of matrices is limited, but the program allows the user to manage and control use of storage by permitting him to delete matrices and to observe what matrices have been stored.

These capabilities make matrix computations quick and simple. Such computations have many uses, for example, in the study of linear systems and stability problems.



(a) First breakdown of user input.

Figure 1. - General flow of program logic.

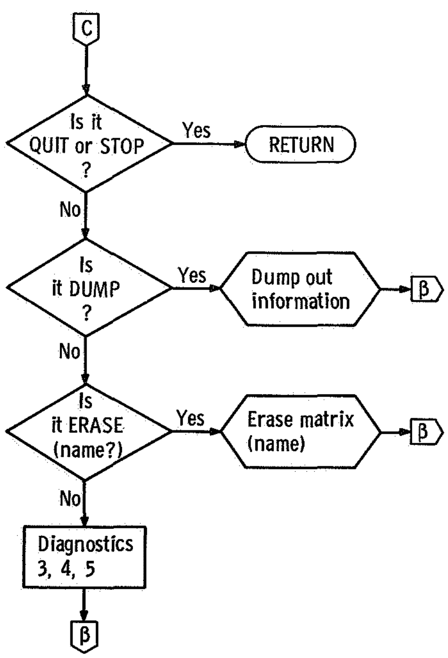
Limitations of Program

The program is designed for "medium-size" (10 rows and/or columns) problems. It is not intended for solving large systems of equations, for extensive computations, or for huge blocks of input or output. For this reason, it is limited in input/output, storage, arithmetic, and reasoning.

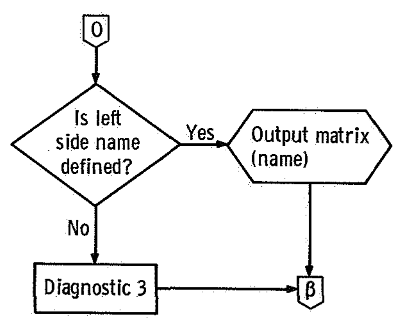
There are two limits on input/output. First, input/output is presently by means of typewriter-like terminals only, because they are better suited to conversational applications than magnetic tape, card readers, or printers. Second, users may type in no more than two lines, or about 240 characters. This restriction is easily removed.

Storage of matrices is limited, both in the size of matrices and in the total number of matrix elements. Individual matrices can have as many as 10 rows and/or 10 columns. Storage is provided for roughly 1400 elements (e.g., fourteen 10×10 matrices or fifty-six 5×5 matrices).

All arithmetic is limited to six significant figures. Matrix inversion and eigenvalues are usually accurate to five figures. The program translates all input matrix elements to real, short-precision numbers; only positive integers are allowed in the exponentiation operation.



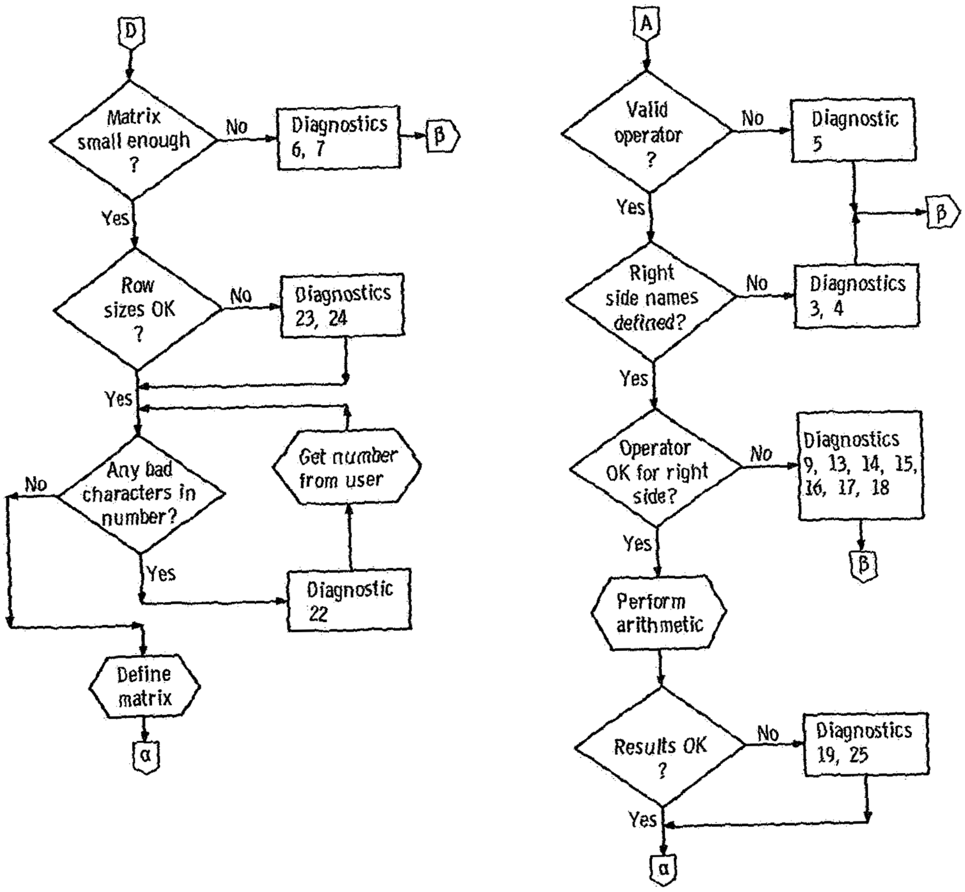
(b) Utility request processing.



(c) Output request processing.

Figure 1. - Continued.

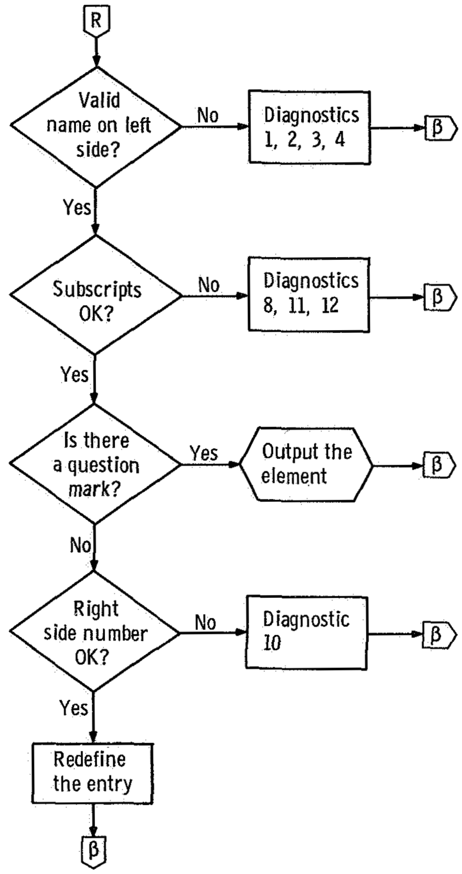
Logic limitations allow rapid processing of correct commands, but can occasionally cause the user some confusion. For example, the program regards any line with a colon on the right side of an equal sign as a definition of a matrix. Figure 1 shows the general flow of the logic used to translate commands typed in by the user. This figure is especially intended to show why the diagnostic messages are issued.



(d) Definition request processing.

(e) Arithmetic request processing.

Figure 1. - Continued.



(f) Individual element request processing.

Figure 1. - Concluded.

USER'S GUIDE TO MATAR

Some Details of Usage

To use the program, users gain access to terminals (or other such devices) and begin by executing the call CALL MATAR. Details for doing this will differ (see appendix). The following example shows the beginning of a user-program conversation. The user commands are (arbitrarily) in small letters; the program responses are in capital letters and are indented slightly. Blanks are ignored by the program, except that where a number is expected, blanks are interpreted as zero. The user can easily include comments because the program ignores lines beginning with an apostrophe.

```
load sample
run sample
 YOU ARE USING A MATRIX ARITHMETIC PROGRAM.
 WANT MORE INFORMATION?
yes
 I RECOGNIZE THE FOLLOWING COMMAND FORMS:
 (1) NAME1 = A11,A12,...,A1M;A21,...,A2M;...;AN1,...,ANM:
    THIS DEFINES MATRIX NAME1, ENTERED ROW BY ROW.
 (2) NAME1 = NAME2 OP , WHERE OP = I% (INVERSE) D% (DETERMINANT)
     T%(TRANSPOSE) P%N(TO POWER N) OR E%(EIGENVALUES)--
      THIS LAST DEFINES NAME1 TO BE A MATRIX WHOSE FIRST COLUMN
      IS THE REAL PARTS, AND WHOSE SECOND COLUMN IS THE IMAG
      PARTS, OF THE EIGENVALUES OF MATRIX NAME2.
 (3) NAME1 = NAME2 OP NAME3 , WHERE OP = + - *
 (4) NAME1 = ?          THIS OUTPUTS MATRIX NAME1
 (5) QUIT, OR STOP      THESE CAUSE A RETURN.
 (6) DUMP        THIS TELLS YOU ALL MATRIX NAMES AND SIZES.
 (7) ERASE(NAME1)  THIS ERASES MATRIX CALLED NAME1.
 (8) NAME1 = NAME2 OP NAME3 , WHERE OP = A%(DIAGONAL ADD),
    S%(DIAGONAL SUBTRACT), OR M%(DIAGONAL MULTIPLY), AND NAME2
    OR NAME3 MUST BE A SCALAR.
 (9) NAME1(N,M) = XX     IF XX IS ? , THIS OUTPUTS THE NAME1 ELEMENT IN ROW N, COLUMN M;
    IF XX IS A NUMBER, IT IS PLACED IN THAT ELEMENT.

 YOU MAY USE TWO LINES FOR INPUT. TO CONTINUE TO A SECOND LINE
 ENTER  |  SOMEWHERE IN THE FIRST LINE


 DO YOU NEED ADDITIONAL COMMANDS?
no
 READY
a=1,0,3.0;0,2,3;0,0,3:
 READY
ainv=a I%
 READY
ainv=?
 ROW  1   1.00000       0.0           -1.000000
 ROW  2  0.0            0.500000      -0.500000
 ROW  3  0.0            0.0            0.333333
 READY
stop

 CHCIW STOP   RETURN
```

After MATAR is executing, it asks if information is desired (see preceding example). The user replies yes. The program types out information on its usage; when this is done, the program types READY. Whenever this READY appears, the program expects a new complete command for processing. The example continues by showing the definition of a matrix, inversion of it, writing the inverse, and stopping.

The same problem is done, but with some user's errors to show typical program responses to imperfect commands in the following example:

```
 READY
a=1,0,%;0,22,3;0,0,3,4:
 BAD CHARACTER %    . ROW 1 , COLUMN  3
 RE-ENTER NUMBER FOR THIS ENTRY. DEFAULT: ZERO.
3.
 WARNING: ROW  3 IS TOO LONG AND HAS BEEN TRUNCATED.
 READY
a=?
 ROW  1   1.00000      0.0            3.00000
 ROW  2  0.0            22.0000       3.00000
 ROW  3  0.0           0.0            3.00000
 READY
a(2,2)=2
 READY
ainv=aa i%
           AA NOT FOUND.
 READY
ainv=ai%
 READY
ainv=!
 DID NOT SEE A  ?  :  OR VALID COMMAND OR OPERATOR. LINE REJECTED.
 READY
ainv=?
 ROW  1   1.00000      0.0           -1.000000
 ROW  2  0.0           0.500000      -0.500000
 ROW  3  0.0           0.0            0.333333
 READY
identity3=a*ainv
 NAME HAS MORE THAN 8 CHARS. BEGINS WITH  IDENTITY
 READY
identity=a*ainv
 READY
identity=?
 ROW  1   1.00000      0.0           0.0
 ROW  2  0.0            1.00000      0.596046E-07
 ROW  3  0.0           0.0           1.000000
 READY
stop
CHCIW STOP   RETURN
```

Command Language

The program recognizes, and converses in, its own language. This language is simple to learn and use. Only this language is recognized, so that knowledge of standard computer programming languages is not necessary.

The user commands are described in the following formats and examples:

(1) $\text{name}_1 = a_{11}, a_{12}, \ldots, a_{1m}; a_{21}, \ldots, a_{2m}; \ldots; a_{n1}, \ldots, a_{nm}:$

This command is used to define an $n{\times}m$ matrix and store it as matrix name_1. The entries are separated by commas; the rows are separated by semicolons; and the matrix must end with a colon. (Note: e-1 is FORTRAN notation for $\times 10^{-1}$.) Name_1 may contain one to eight alphanumeric or special characters, with the exception of reserved characters, which include + - * % ? : = ( ).

Example: a = 1, 3.0, 3.2; 1 e-1, 2, 0:

This defines matrix a to be

$$\begin{pmatrix} 1.0 & 3.0 & 3.2 \\ 0.1 & 2.0 & 0.0 \end{pmatrix}$$

(2) $name_1$ (i, j) = xxx

This command is used to redefine the entry in the i^{th} row, j^{th} column, of matrix $name_1$ to be the number xxx.

Example: a(1, 2) = 0

This redefines matrix a to be

$$\begin{pmatrix} 1.0 & 0.0 & 3.2 \\ 0.1 & 2.0 & 0.0 \end{pmatrix}$$

(3) $name_1$ = ?

This command is used to type out the matrix called $name_1$.

Example: a = ?

This causes the output

ROW 1	1.00000	0.0	3.20000
ROW 2	0.10000	2.00000	0.0

(4) $name_1$ (i, j) = ?

This command is used to output the i^{th} row, j^{th} column element in matrix $name_1$.

Example: a(2, 2) = ?

This command causes the output 2.00000.

(5) $name_1$ = $name_2$ op $name_3$

where op (short for matrix operator) is + (add), - (subtract), or * (multiply). This command is used to perform the arithmetic on the matrices $name_2$ and $name_3$ and defines the result to be matrix $name_1$. If either matrix is 1×1, the arithmetic performed is scalar arithmetic.

Example: b = a + a

This defines the matrix B to be

$$\begin{pmatrix} 2.0 & 0.0 & 6.4 \\ 0.2 & 4.0 & 0.0 \end{pmatrix}$$

(6) $name_1 = name_2$ op

where op is i% (inverse), d% (determinant), t% (transpose), p%N (to power N), or e% (eigenvalues). The eigenvalue command is used to define matrix $name_1$ to be a two-column matrix whose rows contain the real, then the imaginary, parts of the eigenvalues of matrix $name_2$.

Example: b = b t%

This redefines b to be

$$\begin{pmatrix} 2.0 & 0.2 \\ 0.0 & 4.0 \\ 6.4 & 0.0 \end{pmatrix}$$

As this example shows, previously-defined matrices can be redefined.

(7) $name_1 = name_2$ op $name_3$

where op is a% (add), s% (subtract), or m% (multiply). This command is used to perform scalar arithmetic on diagonal entries only. Therefore, either $name_2$ or $name_3$ must be scalar, the other must be square.

Example: If

$$c = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}$$

and scalar = 2, c = c m% scalar redefines c to be

$$\begin{pmatrix} 4 & 1 \\ 0 & 6 \end{pmatrix}$$

(8) dump

This command is used to output all matrix names and dimensions.

Example (after defining only a and b above): DUMP

This would cause the output

YOU HAVE DEFINED 2 MATRICES.

A HAS 2 ROWS, 3 COLUMNS.

B HAS 3 ROWS, 2 COLUMNS.

(9) erase ($name_1$)

This command is used to erase the matrix $name_1$, releasing the space it occupied for use.

Example: erase(a)

This command erases the elements in matrix a.

(10) stop

This command stops execution of MATAR.

PROGRAM MESSAGES

Whenever communication breaks down and MATAR has trouble with a command, it will issue specific complaints. These help the user find errors and learn to use the program. There are three basic types of complaints.

First, no arithmetic is performed (but no previous matrices are lost) if the user's command violates matrix algebra requirements. For example, the program does not try to add two matrices with different numbers of rows. Such errors are arithmetic errors.

Second, in trying to fit the input into one of the recognized command formats, the program sometimes finds forms which violate built-in assumptions and limitations. For example, no name can be longer than eight characters. These violations are called user errors.

These two error types are called fatal, because user commands containing them are not carried out. Finally, there are some messages which are for user's information. They warn him of conditions which are not fatal to the command, but which should be considered. For example, if a user includes nonnumeric characters in a number, he must reenter that number, but the rest of the line is not lost. All three types of program messages are listed in table I.

TABLE I. - MESSAGES ISSUED BY PROGRAM

	User errors
1	NAME HAS MORE THAN 8 CHARACTERS. BEGINS WITH xxxxxxxx.
2	RESERVED CHARACTER USED IN NAME, LINE REJECTED.
3	xxxxxxxx NOT FOUND.
4	BLANK(S) FOUND WHERE NAME, COMMAND, OR REPLY REQUIRED, RE-ENTER.
5	DID NOT SEE A ? : OR VALID COMMAND OR OPERATOR, LINE REJECTED.
6	MORE THAN 10 COLUMNS, LINE REJECTED.
7	MORE THAN 10 ROWS, LINE REJECTED.
8	SUBSCRIPT(S) OUT OF RANGE, xxxxxxxx HAS m ROWS, n COLUMNS.
9	BAD CHARACTER x IN EXPONENT, LINE REJECTED.
10	BAD CHARACTER x IN NUMBER, LINE REJECTED.
11	BAD CHARACTER x IN SUBSCRIPT, LINE REJECTED.
12	NO COMMA SEPARATING SUBSCRIPTS, LINE REJECTED.
	Arithmetic errors
13	MATRICES HAVE DIFFERENT NO. OF ROWS. OPERATION IGNORED.
14	MATRICES HAVE DIFFERENT NO. OF COLUMNS. OPERATION IGNORED.
15	OPERATION NOT DEFINED FOR NONSQUARE MATRIX.
16	FIRST NO. COLUMNS NOT EQUAL SECOND NO. ROWS. MATRIX MULTIPLY REQUEST IGNORED.
17	SINGULAR MATRIX. INVERSE NOT PERFORMED.
18	OPERATOR NOT APPLICABLE FOR MATRICES.
	Warnings
19	MATRIX ILL-CONDITIONED. INVERSE MAY BE INACCURATE.
20	***WARNING*** ROOM FOR ROUGHLY n MORE ENTRIES.
21	*****WARNING***** STORAGE SPACE FOR MATRICES EXCEEDED BY n, RESULTS UNPREDICTABLE.
22	BAD CHARACTER x. ROW n, COLUMN m. RE-ENTER NUMBER FOR THIS ENTRY. DEFAULT: ZERO.
23	WARNING: ROW n IS TOO LONG AND HAS BEEN TRUNCATED.
24	WARNING: ROW n IS TOO SHORT AND HAS BEEN PADDED WITH ZEROS.
25	**WARNING** EIGEN FAILED TO CONVERGE ON AN EIGENVALUE.

CONCLUSIONS

Using computers conversationally, that is, in a real-time interactive way, can be easy for nonprofessionals, as well as fruitful for computer professionals. Matrix calculations have been presented as an example of this conversational use. A brief but powerful set of commands to do matrix calculations has been developed. A FORTRAN program has been written which enables the computer to converse with users (for example, through typewriter-like terminals) and interpret these commands. This program is running successfully under a time-sharing computer system.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, November 27, 1968,
125-23-02-15-22.

APPENDIX - NOTES FOR PROGRAMMERS

Program Details

The program was developed and run under the Time-Sharing System (TSS) of an IBM 360/model 67 (ref. 1). The entire program is roughly 9200 four-byte words (1 byte = 8 bits) in length. Under TSS, each user's virtual storage is divided into pages, at 4096 bytes per page. The number of these pages that the program requires is a strong function of how much the various subprograms have been combined at compile time and by the link-editor.

Source decks for the program are in TSS FORTRAN (ref. 2). The program calls two shift functions standard to IBM 360 - ISLL (NN, IWORD), which shifts the integer short-precision word IWORD NN bits to the left; and ISRL(NN, IWORD), which shifts IWORD NN bits to the right. (See note below.) The program is executed under TSS by compiling the source decks, and then LOADing and RUNning the object decks into the user's virtual memory.

Note: If shift functions above are not readily available, the following FORTRAN coding will serve:

```
C        FORTRAN SHIFTS FOR MATAR
      FUNCTION ISLL(NN,IWRD)
C        MASKR GOOD ONLY FOR 24 BITS TO RIGHT
      INTEGER MASKR/Z00000080/, MASK/Z80000000/
      IWORD = IWRD
      DO 1 I=1,NN
      IWORD = IWORD*2
    1 CONTINUE
      GO TO 3
      ENTRY ISRL(NN,IWRD)
      IWORD = IWRD
C        SET SIGN BIT TO ZERO
      IF(IWORD.LT.0) IWORD = IWORD + MASK
      DO 2 I=1,NN
      IWORD = IWORD/2
    2 CONTINUE
C        RESTORE BIT IF NEEDED
      IF(IWRD.LT.0) IWORD = IWORD + MASKR
    3 ISLL = IWORD
      RETURN
      END
```

Numerical Methods

Matrix arithmetic is done by real short-precision arithmetic with operations on 32-bit words yielding six siginficant figures. The exponentiation command simply causes repeated multiplication.

The inverse, determinant, and eigenvalues are calculated only for square matrices. The inverse and determinant are formed by using Gauss-Jordan elimination (ref. 3). The eigenvalues are calculated by transforming the matrix using Danilevski's method (ref. 4). These calculations are performed by means of standard subprograms developed at this Center.

Listing of FORTRAN Program

The program is entered by running any program which makes the call CALL MATAR. All matrix information and elements are in COMMON/MATE/. A complete listing follows.

```
C
C
C        INITIALIZES, TALKS A LITTLE
      SUBROUTINE MATAR
      DIMENSION INP(10),NM(2)
      COMMON/MATE/ NUM,ILAST,MATRIX(1500)
      DATA IY,IN,INEW/' YES',' NO',1/
      IF(INEW.NE.1) GO TO 3
      INEW = 2
      NUM = 0
      ILAST = 0
      DO 5 I=1,1500
    5 MATRIX (I) = 0
      PRINT 101
      GO TO 2
    1 PRINT 102
    2 READ 201,INP
      CALL NMPACK(INP,1,10,NM,.TRUE.,&1,&1,&2)
      IF(NM(2).EQ.IN) GO TO 3
      IF(NM(2).NE.IY) GO TO 1
      PRINT 103
      PRINT 106
      PRINT 104
      GO TO 7
    6 PRINT 102
    7 READ 201,INP
      CALL NMPACK(INP,1,10,NM,.TRUE.,&6,&6,&7)
      IF(NM(2).EQ.IN) GO TO 3
      IF(NM(2).NE.IY) GO TO 6
      PRINT 105
    3 CALL READER
      RETURN
```

```
 101 FORMAT('  YOU ARE USING A MATRIX ARITHMETIC PROGRAM.',/,
    * '  WANT MORE INFORMATION?')
 102 FORMAT('  TYPE YES OR NO.')
 103 FORMAT('  I RECOGNIZE THE FOLLOWING COMMAND FORMS:',/,
    1 '  (1) NAME1 = A11,A12,...,A1M;A21,...,A2M;...;AN1,...,ANM;',10X,
    2 /,5X,'THIS DEFINES MATRIX NAME1, ENTERED ROW BY ROW.',/,
    3 '  (2) NAME1 = NAME2 OP , WHERE OP = I% (INVERSE) D% (DETERMINANT
    4)',/,5X,' T%(TRANSPOSE) P%N(TO POWER N) OR E%(EIGENVALUES)--',/,
    C5X,'  THIS LAST DEFINES NAME1 TO BE A MATRIX WHOSE FIRST COLUMN',/
    D,5X,'  IS THE REAL PARTS, AND WHOSE SECOND COLUMN IS THE IMAG',/,
    E5X,'  PARTS, OF THE EIGENVALUES OF MATRIX NAME2.',/,
    5 '  (3) NAME1 = NAME2 OP NAME3 , WHERE OP = + - * ',/,
    6 '  (4) NAME1 =  ?          THIS OUTPUTS MATRIX NAME1',/,
    7'  (5) QUIT, OR STOP     THESE CAUSE A RETURN.',/,
    8 '  (6) DUMP          THIS TELLS YOU ALL MATRIX NAMES AND SIZES.',/,
    9 '  (7) ERASE(NAME1)  THIS ERASES MATRIX CALLED NAME1.')
 104 FORMAT('  DO YOU NEED ADDITIONAL COMMANDS?')
 105 FORMAT('  CONTACT BRUCE CANRIGHT, PAX 6263 OR 8224.')
 106 FORMAT('  (8) NAME1 = NAME2 OP NAME3 , WHERE OP = A%(DIAGONAL ADD)
    1,',/,5X,'S%(DIAGONAL SUBTRACT), OR M%(DIAGONAL MULTIPLY), AND NAME
    22',/,5X,'OR NAME3 MUST BE A SCALAR.',/,'  (9) NAME1(N,M) = XX
    3IF XX IS ? , THIS OUTPUTS THE NAME1 ELEMENT IN ROW N, COLUMN M;',/
    3,5X,'IF XX IS A NUMBER, IT IS PLACED IN THAT ELEMENT.',/,/,
    A '  YOU MAY USE TWO LINES FOR INPUT. TO CONTINUE TO A SECOND LINE'
    B ,/,'  ENTER  |  SOMEWHERE IN THE FIRST LINE',/,/ )
 201 FORMAT(10A1)
     END
```

```
C
      SUBROUTINE READER
C*****************************************************************************
C
C        READER READS THE INPUT AND DECIDES WHAT TO CALL
C        THE 1,2 ENTRY IS IN THE FIRST ROW, SECOND COLUMN
C        OUTLINE OF MAJOR FLOW FOLLOWS
C
C          FOR COMMAND TYPE (1)  READER--CHANGE--CNVTR
C          FOR COMMAND TYPE (2),(4)  READER--REPAIR
C          FOR COMMAND TYPE (3)  READER--OUTPUT
C          FOR COMMAND TYPE (5),(6),(7)  READER--ARITHM--ARITHMETICSTUFF
C          FOR COMMAND TYPE (8),(9),(10)  READER--COMAND--DUMPY
C
C*****************************************************************************
C
C        NMPACK  PACKS A*1 LEFT-ADJUSTED INTO 2A4 RIGHT-ADJUSTED
C        NMDEF  DEFINES NEW NAME
C        FINDNM  LOOKS UP OLD MATRIX
C        WIPE  ERASES OLD MATRIX
C
C*****************************************************************************
```

```
      LOGICAL OLD
      DIMENSION NM(2),NM1(2),INPUT(220)
      COMMON/MATE/NUM,ILAST,MATRIX(1500)
      COMMON/PASS/ KLP,KRP,K,KCOL
      DATA II,ID,IT,IP,IE / 'I','D','T','%','P' /
      DATA IM,IA,IS,ICON,IEQ,IQM,IBLK/'*','+','-','|','=','?',' '/
      DATA ICOM,ISEM,ICOL,ILP,IRP,IV/',',';',':','(',')','E'/
      DATA IA2,IS2,IM2,IAP / 'A','S','M', 1H' /
   50 IF(ILAST.LT.1400) GO TO 47
      NMAX = ILAST - 1399
      PRINT 98, NMAX
   98 FORMAT('  *****WARNING***** STORAGE SPACE FOR MATRICES EXCEEDED',/
     *, '   BY ',I3,' , RESULTS UNPREDICTABLE.')
      GO TO 46
   47 IF(ILAST.LT.1300) GO TO 46
      NMAX = 1399 - ILAST
      PRINT 99, NMAX
   99 FORMAT('  ***WARNING*** ROOM FOR ROUGHLY ',I4,' MORE ELEMENTS.')
   46 PRINT 100
  100 FORMAT('  READY')
      DO 45 I=1,220
   45 INPUT(I) = IBLK
      READ 201, (INPUT(I),I=1,110)
      DO 51 I=1,110
      K = I
      IF(INPUT(K).EQ.ICON) GO TO 52
   51 CONTINUE
  201 FORMAT(120A1)
      GO TO 53
C         LINE CONTINUED
   52 READ 201, (INPUT(I),I=111,220)
      INPUT(K) = IBLK
C         LOOK FOR EQUALS SIGN
C         IF APOSTROPHE, IGNORE WHOLE LINE, ASSUME COMMENT
   53 DO 54 I=1,220
      KEQ = I
      IF(INPUT(KEQ).EQ.IAP) GO TO 46
      IF(INPUT(KEQ).EQ.IEQ) GO TO 55
   54 CONTINUE
C         NO EQUALS SIGN   IS IT COMMAND
      CALL COMAND(INPUT,NM,&60,&5,&44,&46)
      GO TO 46
   44 RETURN
C
C         LOOK FOR LHS PARENS
   55 DO 551 I=1,KEQ
      KLP = I
      IF(INPUT(KLP).EQ.ILP) GO TO 552
  551 CONTINUE
      GO TO 560
  552 DO 553 I=KLP,KEQ
      KRP = I
      IF(INPUT(KRP).EQ.IRP) GO TO 554
  553 CONTINUE
      GO TO 49
  554 DO 555 I=KLP,KRP
      KCOM = I
      IF(INPUT(KCOM).EQ.ICOM) GO TO 556
```

```
  555 CONTINUE
      PRINT 101
  101 FORMAT('  NO COMMA SEPARATING SUBSCRIPTS, LINE REJECTED.')
      GO TO 46
  556 CALL NMPACK(INPUT,1,KLP-1,NM,.FALSE.,&56,&49,&46)
      CALL FINDNM(NM,INM,&60)
      CALL REPAIR(INPUT,INM,KLP,KEQ,KRP,NM,&46)
      GO TO 46
C         ASSUME REGULAR MATRIX NAME LHS
C         PACK THE LEFT HAND NAME
  560 INPUT(KEQ) = IBLK
      CALL NMPACK(INPUT,1,KEQ,NM,.FALSE.,&56,&49,&46)
      GO TO 57
   56 PRINT 106, NM
  106 FORMAT('  NAME HAS MORE THAN 8 CHARS. BEGINS WITH  ',2A4)
      GO TO 46
   49 PRINT 105
  105 FORMAT('  RESERVED CHARACTER USED IN NAME, LINE REJECTED.')
      GO TO 46
C         LOOK FOR QUESTION MARK
   57 DO 58 I=KEQ,220
      IF(INPUT(I).EQ.IQM) GO TO 59
   58 CONTINUE
      GO TO 90
C         ASSUME NAME = ?
   59 CALL FINDNM(NM,INM,&60)
C         NAME FOUND
      NROWS = MATRIX(INM+3)
      NCOLUM = MATRIX(INM+4)
      CALL OUTPUT(MATRIX(INM+5),NROWS,NCOLUM)
      GO TO 46
C         NAME NOT FOUND
   60 PRINT 107, NM
  107 FORMAT(   5X,2A4,' NOT FOUND.')
      GO TO 46
   61 PRINT 107,NM1
      GO TO 46
   62 PRINT 106, NM1
      GO TO 46
   65 DO 66 I=KEQ,220
C         LOOK FOR VALID OPERATORS
      K = I
      IF(INPUT(K).EQ.IM) GO TO 71
      IF(INPUT(K).EQ.IA) GO TO 72
      IF(INPUT(K).EQ.IS) GO TO 73
   66 CONTINUE
C         NO * + OR - ENCOUNTERED. LOOK FOR INV OR DET.
C         LOOK FOR I%  D%  T%
      DO 67 I=KEQ,218
      K = I
      IF(INPUT(K) .EQ. IP) GO TO 68
   67 CONTINUE
      GO TO 5
```

```
   68 K = K - 1
      ITEMP = INPUT(K)
      IF(ITEMP .EQ. II) GO TO 74
      IF(ITEMP .EQ. ID) GO TO 75
      IF(ITEMP .EQ. IT) GO TO 69
      IF(ITEMP .EQ. IV) GO TO 70
      IF(ITEMP.EQ.IE) GO TO 81
      IF( ITEMP.EQ.IA2) GO TO 10
      IF( ITEMP.EQ.IS2) GO TO 11
      IF( ITEMP.EQ.IM2) GO TO 12
C        NO KEY THINGS FOUND
    5 PRINT 109
  109 FORMAT('  DID NOT SEE A  ?   :  OR VALID COMMAND OR OPERATOR. LINE
     1REJECTED.')
      GO TO 46
C        OPERATOR CODES * 1 + 2 - 3 INV 4 DET 5
   69 KODE = 6
      GO TO  76
   70 KODE = 8
      GO TO 76
   71 KODE = 1
      GO TO 76
   72 KODE = 2
      GO TO 76
   73 KODE = 3
      GO TO 76
   74 KODE = 4
      GO TO 76
   75 KODE = 5
      GO TO 76
   10 KODE = 9
      GO TO 76
   11 KODE = 10
      GO TO 76
   12 KODE = 11
      GO TO 76
   81 KODE = 7
      KJ = K + 1
      INPUT(KJ) = ILP
      INPUT(220) = IRP
      CALL CNVTR(INPUT,KJ,220,2,ARG,INM2,ICHAR,&76,&83,&84)
      GO TO 76
   83 ICHAR = ICOM
   84 PRINT 121, ICHAR
  121 FORMAT('  BAD CHARACTER ',A4,' IN EXPONENT, LINE REJECTED.')
      GO TO 46
C        CHECK TO SEE IF LEFTHAND NAME IS NEW
   76 OLD = .FALSE.
      CALL FINDNM(NM,INM,&79)
      OLD = .TRUE.
   79 JJ = KEQ + 1
C        RESOLVE RIGHTHAND NAME(S).
      INPUT(K) = IBLK
      CALL NMPACK(INPUT,JJ,K,NM1,.TRUE.,&62,&49,&46)
      CALL FINDNM(NM1,INM1,&61)
      NA = MATRIX(INM1+3)
      MA = MATRIX(INM1+4)
      IF(KODE.GT.3 .AND. KODE.LT.9) GO TO 77
```

```
      JJ = K+1
      IF(KODE.GT.3) JJ = JJ + 1
      CALL NMPACK(INPUT,JJ,220,NM1,.TRUE.,&62,&49,&46)
      CALL FINDNM(NM1,INM2,&61)
      NB = MATRIX(INM2+3)
      MB = MATRIX (INM2+4)
C        PUT AWAY NEW NAME
   77 CALL NMDEF(NM)
C        NAMES RESOLVED, READY TO DO ARITHMETIC
      IN = ILAST - 1
      CALL ARITH(KODE,IN,INM1,INM2,NA,MA,NB,MB,NC,MC,&80)
      MATRIX(ILAST+2) = NC
      MATRIX(ILAST+3) = MC
      MATRIX(ILAST+1) = NC*MC + 5
      ILAST = ILAST + NC*MC + 3
      IF(.NOT.OLD) GO TO 50
C        ERASE OLD NAME IN MIDDLE
      CALL WIPE(INM)
      GO TO 50
C        BAD MATCHUP ROWS AND COLUMNS  GO BACK
   80 MATRIX(ILAST) = IBLK
      MATRIX(ILAST-1) = IBLK
      ILAST = ILAST - 2
      NUM = NUM - 1
      GO TO 46
C        LOOK FOR COLON  IF SO DEFINE MATRIX
   90 DO 91 I=KEQ,190
      KCOL = I
      IF(INPUT(I) .EQ. ICOL) GO TO 92
   91 CONTINUE
C        NO COLON FOUND  ASSUME ARITH
      GO TO 65
   92 INPUT(KEQ) = ILP
      KLP = KEQ
      KRP = KCOL + 1
      INPUT(KRP) = IRP
   95 NCOM = 0
      CALL FINDNM(NM,INM,&96)
C        ALREADY DEFINED
      CALL WIPE(INM)
   96 CALL NMDEF(NM)
C        COUNT THE FIRST COMMAS, FIND A SEMICOLON AT K .
      DO 1 I=KLP,KCOL
      K = I
      IF(INPUT(K) .EQ. ICOM) NCOM = NCOM + 1
      IF(INPUT(K).EQ.ISEM .OR. INPUT(K).EQ.ICOL) GO TO 2
    1 CONTINUE
    2 NCOLUM = NCOM + 1
      NSEMI = 0
      IF(INPUT(K).EQ.ISEM) NSEMI = 1
      IF(NCOLUM .LE. 10) GO TO 21
      PRINT 102
  102 FORMAT('  MORE THAN 10 COLUMNS. LINE REJECTED.')
      GO TO 80
   21 INPUT(K) = IRP
```

```
C          COUNT THE SEMICOLONS, COLON AT J .
      DO 3 I=K,KCOL
      J = I
      IF(INPUT(J) .EQ. ISEM) NSEMI = NSEMI + 1
    3 CONTINUE
    4 NROWS = NSEMI + 1
      IF(NROWS .LE. 10) GO TO 41
      PRINT 103
  103 FORMAT('  MORE THAN 10 ROWS. LINE REJECTED.')
      GO TO 80
   41 MATRIX(ILAST+1) = NROWS*NCOLUM + 5
      MATRIX(ILAST+2) = NROWS
      MATRIX(ILAST+3) = NCOLUM
      ILAST = ILAST + 3
      CALL CHANGE(INPUT,MATRIX(ILAST+1),NROWS,NCOLUM,&97)
      ILAST = ILAST + NROWS*NCOLUM
      GO TO 50
C          GO BACK , CONVERSION FAILED .
   97 ILAST = ILAST - 3
      GO TO 80
      END
```

```
C
      SUBROUTINE NMPACK(INPUT,K1,J1,NM,OLD,*,*,*)
C**********************************************************************
C
C          ISLL AND ISRL ARE ASSEMBLY LANGUAGE ROUTINES IN OUR SYSTEM
C          ISLL(NN,IWORD)   SHIFTS I*4 IWORD NN BITS TO LEFT
C          ISRL(NN,IWORD)   SHIFTS I*4 IWORD NN BITS TO RIGHT
C
C**********************************************************************
C
C          TAKES THE JUNK IN INPUT(K) TO INPUT(J) AND
C          PACKS IT INTO A NAME IF IT LESS THAN 9 NON-BLANK CHARACTERS
C          RETURN 1 MORE THAN 8 CHARS
C          RETURN 2 OPERATOR IN NAME
C          RETURN 3  BLANKS FOUND
      LOGICAL OLD
      DIMENSION NM(2),INPUT(1)
      DATA IBL,IPL,IMI,IAS,IPER,IQM,ICOL,IEQ,ILP,IRP
     *  / ' ','+','-','*','%',' ?',':','=','(',')' /
      J = J1
      K = K1
      NM(1) = IBL
      NM(2) = IBL
      KTR = 0
      IF(OLD) GO TO 3
      DO 1 I=K,J
      IN = INPUT(I)
      IF(IN.EQ.IPL .OR. IN.EQ.IMI .OR. IN.EQ.IAS .OR. IN.EQ.IPER)GO TO 6
      IF(IN.EQ.ILP .OR. IN.EQ.IRP) GO TO 6
      IF(IN.EQ.IQM .OR. IN.EQ.ICOL .OR. IN.EQ.IEQ) GO TO 6
```

```
  1 CONTINUE
  3 DO 4 I=K,J
    ITEMP = INPUT(I)
    IF(ITEMP.EQ.IBL) GO TO 4
    KTR = KTR + 1
    IF(KTR.GT.8) GO TO 5
    NM(1) = ISLL(8,NM(1)) + ISRL(24,NM(2))
    NM(2) = ISLL(8,NM(2)) + ISRL(24,ITEMP)
  4 CONTINUE
    IF(KTR.EQ.0) GO TO 7
    RETURN
  5 RETURN 1
  6 RETURN 2
  7 PRINT 11
 11 FORMAT('  BLANK(S) FOUND WHERE NAME, COMMAND, OR REPLY REQUIRED, R
   *E-ENTER.')
    RETURN 3
    END
```

```
C
C
C         DEFINE NAME AND PUT IT IN BIG ARRAY
      SUBROUTINE NMDEF(NM)
      DIMENSION NM(2),INPUT(1)
      COMMON/MATE/ NUM,ILAST,MATRIX(1500)
      NUM = NUM + 1
      ILAST = ILAST + 2
      MATRIX(ILAST) = NM(2)
      MATRIX(ILAST-1) = NM(1)
      RETURN
      END
```

```
C
C
C         SEARCH FOR NAME
C         RETURN 1 NM NOT FOUND
C         RETURN   NM(1) AT MATRIX(INM)
      SUBROUTINE FINDNM(NM,INM,*)
      DIMENSION NM(2)
      COMMON/MATE/ NUM,ILAST,MATRIX(1500)
      J = 1
      DO 10 I=1,NUM
      IF(NM(1).EQ.MATRIX(J) .AND. NM(2).EQ.MATRIX(J+1)) GO TO 15
      J = J + MATRIX(J+2)
   10 CONTINUE
      RETURN 1
   15 INM = J
      RETURN
      END
```

```
C
C
C          ERASES MATRIX NM, MOVES EVERYTHING BELOW UP
      SUBROUTINE WIPE(II)
      COMMON/MATE/ NUM,ILAST,MATRIX(1500)
      INM = II
      IDEL = MATRIX(INM+2)
      IST = INM + IDEL
      IEND = IST - 1
      DO 2 I=INM,IEND
    2 MATRIX(I) = 0
      DO 1 I=IST,ILAST
      J = I - IDEL
      MATRIX(J) = MATRIX(I)
    1 CONTINUE
      ILAST = ILAST - IDEL
      NUM = NUM - 1
      RETURN
      END
```

```
C
C
C          THIS IS THE ONLY WAY TO GET STUFF OUT
      SUBROUTINE OUTPUT(ARRAY,NR,NC)
      DIMENSION ARRAY(NR,NC)
      NROWS = NR
      NCOLUM = NC
      IF(NROWS .EQ. 1) GO TO 15
      DO 10 I=1,NROWS
   10 PRINT 101, I,(ARRAY(I,J),J=1,NCOLUM)
      RETURN
   15 PRINT 102, (ARRAY(1,J),J=1,NCOLUM)
      RETURN
  101 FORMAT('   ROW',I3,5G14.6,/,5X,5G14.6)
  102 FORMAT(5X,5G14.6,/,5X,5G14.6)
      END
```

```
C
      SUBROUTINE REPAIR(INPUT,INM,KLP,IEQ,KRP,NM,*)
C         PRINT OUT OR REPLACE INDIVIDUAL ENTRIES
C         RETURN 1  BAD SUBSCRIPT OR NUMBER
      DIMENSION INPUT(1),NM(2)
      COMMON/MATE/ NUM,ILAST,MATRIX(1500)
      DATA IRP,ILP,IQM,ICOM  / ')','(','?',',' /
      IPT = INM
      NROW = MATRIX(IPT+3)
      NCOL = MATRIX(IPT+4)
```

```
C          CONVERT THE SUBSCRIPTS
      KKK = KLP
      DO 2 I=1,2
      CALL CNVTR(INPUT,KKK,KRP,2,ARG,IARG,ICHAR,&15,&1,&15)
      ICOL = IARG
      GO TO 3
    1 IROW = IARG
    2 CONTINUE
    3 IF(IROW.LE.0 .OR. ICOL.LE.0 .OR. IROW.GT.NROW .OR. ICOL.GT.
     *  NCOL) GO TO 14
C          SUBSCRIPTS OK
C          GET POINTER TO PROPER ENTRY
      IPT2 = NROW*(ICOL-1) + IROW + IPT+4
C          IS THERE A QUESTION MARK
      DO 5 I=IEQ,190
      IF(INPUT(I) .EQ. IQM) GO TO 9
    5 CONTINUE
C
C          ...NO, ASSUME NUMBER
      INPUT(190) = IRP
      INPUT(IEQ) = ILP
      CALL CNVTR(INPUT,IEQ,190,1,IANS,IARG,ICHAR,&6,&7,&8)
    6 MATRIX(IPT2) = IANS
      GO TO 10
    7 ICHAR = ICOM
    8 PRINT 103, ICHAR
  103 FORMAT('  BAD CHARACTER ',A4,' IN A NUMBER, LINE REJECTED.')
      GO TO 20
C
C          ...YES, WRITE IT OUT
    9 CALL OUTPUT(MATRIX(IPT2),1,1)
   10 RETURN
C          SUBSCRIPTS OUT OF BOUNDS
   14 PRINT 101, NM,NROW,NCOL
  101 FORMAT('  SUBSCRIPT(S) OUT OF RANGE, ',2A4,' HAS ',I2,
     *  ' ROWS, ',I2,' COLUMNS.')
      GO TO 20
C          BAD SUBSCRIPT
   15 PRINT 102, ICHAR
  102 FORMAT('  BAD CHARACTER ',A4,' IN SUBSCRIPT, LINE REJECTED.')
   20 RETURN 1
      END
```

```
C
      SUBROUTINE CHANGE(INPUT,ARRAY,NROWS,NCOLAM,*)
C          CONVERTS THE INPUT STRING TO R*4 NUMBERS, PUTS INTO /MATE/
C
      DIMENSION INPUT(1),INTEMP(90),ARRAY(NROWS,NCOLAM)
      COMMON/PASS/ KLP,KRP,K,J
      DATA ILP,IRP,ISEM,ICOM,ICOL / '(',')',';',',',':' /
      DATA INTEMP(1),INTEMP(90)/'(',')'/
      NCOLUM = NCOLAM
```

```
C         CONVERT THE FIRST ROW
      KKK = KLP
      DO 5  I=1,NCOLUM
      CALL CNVTR(INPUT,KKK,KRP,1,ANS,GARB,ICHAR,&20,&10,&4)
C         ROW DONE
      ARRAY(1,I) = ANS
      GO TO 6
C         BAD CHARACTER , RE-ENTER
    1 ICHAR = ICOM
    4 PRINT 103, ICHAR,I
      PRINT 105
      READ 201, (INTEMP(III),III=2,89)
      L = 1
      CALL CNVTR(INTEMP,L,90,1,ANS,GARB,ICHAR,&20,&1,&4)
C         SKIP ON TO NEXT COLUMN   (THANKS DUE TO EVA)
      DO 2 I2=KKK,KRP
      ITT = INPUT(I2)
      IF(ITT.EQ.ICOM) GO TO 3
      IF(ITT.EQ.ISEM .OR. ITT.EQ.ICOL) GO TO 6
    2 CONTINUE
    3 KKK = I2
C         NUMBER DONE, ROW NOT YET DONE
   10 ARRAY(1,I) = ANS
      IF(I.EQ.NCOLUM) GO TO 6
    5 CONTINUE
   20 PRINT 102
  102 FORMAT('  NO RIGHT PAREN. SOMETHING REALLY WRONG.')
      RETURN 1
  103 FORMAT('  BAD CHARACTER ',A4,'. ROW 1 , COLUMN',I3)
  105 FORMAT('  RE-ENTER NUMBER FOR THIS ENTRY. DEFAULT: ZERO.')
  201 FORMAT(88A1)
C         DO ROW SUB KTR
    6 KTR = 2
      IF(NROWS.EQ.1) RETURN
    7 INPUT(K) = ILP
      NCOM = 0
      DO 8 I=K,J
      K2 = I
      IF(INPUT(K2).EQ.ISEM.OR.INPUT(K2).EQ.ICOL) GO TO 9
      IF(INPUT(K2) .EQ. ICOM) NCOM = NCOM + 1
    8 CONTINUE
C         FOUND NEXT SEMICOLON, MOVE RIGHT TO K2, SAVE LEFT K
    9 NMAX = NCOM + 1
      IF (NCOLUM-NMAX)  23,11,21
   21 PRINT 107, KTR
  107 FORMAT('  WARNING: ROW',I3,' IS TOO SHORT AND HAS BEEN PADDED WITH
     * ZEROS.')
C         PAD IT
      NMAX = NMAX + 1
      DO 22 I=NMAX,NCOLUM
   22 ARRAY(KTR,I) = 0.0
      GO TO 11
   23 PRINT 104, KTR
  104 FORMAT('  WARNING : ROW',I3,' IS TOO LONG AND HAS BEEN TRUNCATED.')
C         CONVERT THE ROW
   11 INPUT(K2) = IRP
      KKK = K
      DO 15 I=1,NCOLUM
      CALL CNVTR(INPUT,KKK,KRP,1,ANS,GARB,ICHAR,&20,&12,&14)
```

```
C         ROW DONE
      ARRAY(KTR,I) = ANS
      GO TO 16
   24 ICHAR = ICOM
   14 PRINT 106, ICHAR,KTR,I
  106 FORMAT('  BAD CHARACTER ',A4,' . ROW',I3,' COLUMN',I3 )
      PRINT 105
      READ 201, (INTEMP(III),III=2,89)
      L = 1
      CALL CNVTR(INTEMP,L,90,1,ANS,GARB,ICHAR,&20,&24,&14)
C         SKIP ON TO NEXT COLUMN
      DO 17 I2=KKK,KRP
      ITT = INPUT(I2)
      IF(ITT.EQ.ICOM) GO TO 18
      IF(ITT.EQ.ISEM .OR. ITT.EQ.ICOL) GO TO 16
   17 CONTINUE
   18 KKK = I2
C         NUMBER DONE, ROW NOT YET DONE
   12 ARRAY(KTR,I) = ANS
   15 CONTINUE
   16 KTR = KTR + 1
      IF(KTR.GT. NROWS) RETURN
      K = K2
      GO TO 7
      END
```

```
C
      SUBROUTINE CNVTR(INPUT,KTR,MAX,TYPE,ARG1,ARG2,CHAR,*,*,*)
C*************************************************************************
C     SUBROUTINE TO CONVERT A TYPE INPUT TO FLOATING POINT NUMBERS
C              INPUT    ARRAY OF CHARACTERS.  ONE PER WORD LEFT ADJ.
C              KTR      INDEX OF FIRST WORD OF ARRAY TO BE CONSIDERED
C              MAX      DIMENSION OF INPUT
C              TYPE     =1  REAL*4,   =2  INTEGER*4
C              ARG1     REAL*4 ANS
C              ARG2     INTEGER*4 ANS
C              CHAR     BAD CHARACTER IF FOUND
C          RETURN       NEXT CHARACTER ) RESULT GOOD
C          RETURN 1     NO , OR ) FOUND RESULT ZERO
C          RETURN 2     NEXT CHARACTER , RESULT GOOD
C          RETURN 3     BAD CHARACTER RESULT ZERO
C*************************************************************************
      INTEGER INPUT(1),TYPE,TEMP,TEMP1,CHAR
      REAL*4 ARG1
      INTEGER*4 ARG2
      INTEGER BLK,RP,PL,DP,E,CM
      LOGICAL SIN
      DATA MASK1,BLK,LP,RP,PL,MI,DP,E,CM
     * / ZF0404040,' ','(',')','+','-','.','E',',' /
      IEXP = 0
      IP = 1
      ISGN = 1
      ARG1 = 0.0
```

```
      ARG2 = 0
      SIN = .FALSE.
      SIGN = 1.0
      NUM = 1
      ISTART = KTR
      DO 20 I=ISTART,MAX
      KTR = I
      TEMP = INPUT(I)
      IF(TEMP.EQ.BLK)  GO TO 20
      GO TO (8,9,9,9,9,9),NUM
    8 IF(TEMP.NE.LP.AND.TEMP.NE.CM)  GO TO 23
      NUM = 2
      GO TO 20
    9 IF(TEMP.EQ.RP.OR.TEMP.EQ.CM)  GO TO 22
      GO TO (10,10,13,15,17,19),NUM
   10 NUM = 3
      IF(TEMP.NE.PL)  GO TO 11
      IF(SIN .AND. TYPE.EQ.2) GO TO 23
      SIN = .TRUE.
      GO TO 20
   11 IF(TEMP.NE.MI)  GO TO 13
      IF(SIN .AND. TYPE.EQ.2) GO TO 23
      SIN = .TRUE.
      SIGN = -1.0
      GO TO 20
   13 IF(TEMP.EQ.E) GO TO 151
      IF(TEMP.NE.DP)  GO TO 14
      IF(TYPE.EQ.2) GO TO 23
      NUM = 4
      GO TO 20
   14 TEMP1 = ISRL(24,TEMP-MASK1)
      IF(TEMP1.LT.0.OR.TEMP1.GT.9)  GO TO 23
      IF(TYPE.EQ.1)  ARG1 = ARG1*10.0+FLOAT(TEMP1)
      IF(TYPE.EQ.2) ARG2 = ARG2*10+TEMP1
      GO TO 20
   15 IF(TEMP.NE.E) GO TO 16
  151 NUM = 5
      IF(TYPE.EQ.2) GO TO 23
      GO TO 20
   16 TEMP1 = ISRL(24,TEMP-MASK1)
      IF(TEMP1.LT.0.OR.TEMP1.GT.9)  GO TO 23
      IF(TYPE.EQ.1)  ARG1 = ARG1+FLOAT(TEMP1)/10.0**IP
      IP = IP+1
      GO TO 20
   17 NUM = 6
      IF(TEMP.NE.PL)  GO TO 18
      GO TO 20
   18 IF(TEMP.NE.MI)  GO TO 19
      ISGN = -1
      GO TO 20
   19 TEMP1 = ISRL(24,TEMP-MASK1)
      IF(TEMP1.LT.0.OR.TEMP1.GT.9)  GO TO 23
      IEXP = IEXP*10+TEMP1
```

```
   20 CONTINUE
      ARG1 = 0.0
      ARG2 = 0
      RETURN 1
   22 IF(TYPE.EQ.1)  ARG1 = SIGN*ARG1*10.0**(ISGN*IEXP)
      IF(TYPE.EQ.2 .AND. SIGN.LT.0.) ARG2 = -ARG2
      IF(TEMP.EQ.CM)  RETURN 2
      RETURN
   23 ARG1 = 0.0
      ARG2 = 0
      CHAR = TEMP
      RETURN 3
      END
```

```
C
      SUBROUTINE COMAND(INPUT,NM,*,*,*,*)
C
C         IF ERASE, CALL WIPE
C         IF DUMP, CALL DUMP
C         RETURN 1 NAME NOT DEFINED
C         RETURN 2 INVALID COMMAND
C         RETURN 3  SAID STOP OR QUIT
C         RETURN 4  ALL BLANKS FOUND
C***********************************************************************
      DIMENSION INPUT(190),NM(2),IERAS(2)
      DATA ILP,IRP,IERAS,IBLK,IQUIT,IDUMP/'(',')','    E','RASE',
     * '    ','QUIT','DUMP' /
      DATA ISTOP/'STOP'/
C         LOOK FOR PARENS
      DO 1 I=1,190
      KLP = I
      IF(INPUT(I).EQ.ILP) GO TO 2
    1 CONTINUE
C         IS IT QUIT
      CALL NMPACK(INPUT,1,190,NM,.TRUE.,&4,&4,&9)
      IF(NM(1).NE.IBLK) GO TO 4
      IF(NM(2).NE.IQUIT .AND. NM(2).NE.ISTOP) GO TO 7
      RETURN 3
C         IS IT DUMP
    7 IF(NM(2) .NE. IDUMP) GO TO 4
      CALL DUMPY
      GO TO 8
    2 DO 3 I=KLP,190
      KRP = I
      IF(INPUT(I).EQ.IRP) GO TO 5
    3 CONTINUE
    4 RETURN 2
C         PACK ERASE
    5 INPUT(KLP) = IBLK
      CALL NMPACK(INPUT,1,KLP,NM,.TRUE.,&4,&4,&9)
      IF(NM(1).NE.IERAS(1) .OR. NM(2).NE.IERAS(2)) GO TO 4
```

```
C          COMMAND OK
      CALL NMPACK(INPUT,KLP+1,KRP-1,NM,.FALSE.,&4,&4,&9)
      CALL FINDNM(NM,INM,&6)
      CALL WIPE(INM)
    8 RETURN
    6 RETURN 1
    9 RETURN 4
      END
```

```
C
      SUBROUTINE DUMPY
C
C          TELLS USER WHAT IS IN BIG ARRAY
      COMMON/MATE/ NUM,ILAST,MATRIX(1500)
      IF(NUM.GT.0) GO TO 1
      PRINT 103
      GO TO 20
    1 IF(NUM.GT.1) GO TO 2
      PRINT 104
      GO TO 3
    2 PRINT 101, NUM
    3 J = 1
      DO 10 I=1,NUM
      NR = MATRIX(J+3)
      NC = MATRIX(J+4)
      IF(NR.EQ.1 .AND. NC.EQ.1) GO TO 5
C          MATRIX PROPER
      PRINT 102, MATRIX(J),MATRIX(J+1),NR,NC
      GO TO 6
C          SCALAR
    5 PRINT 105, MATRIX(J),MATRIX(J+1)
    6 J = J + MATRIX(J+2)
   10 CONTINUE
   20 RETURN
  101 FORMAT('  YOU HAVE DEFINED ',I2,' MATRICES.')
  102 FORMAT(5X,2A4,5X,'HAS',I3,' ROWS',I3,' COLUMNS.')
  103 FORMAT('  NO MATRICES DEFINED.')
  104 FORMAT('  YOU HAVE DEFINED ONE MATRIX.')
  105 FORMAT(5X,2A4,5X,'IS A SCALAR.')
      END
```

```
C
      SUBROUTINE ARITH(KK,INM,INM1,INM2,N1,M1,N2,M2,NC,MC,*)
C*************************************************************************
C
C         ARITH'S JOB IS TO DEFINE THE ANSWER MATRIX ENTRIES AND SIZE
C         THIS ONE USES FP NUMBERS IN THE COMMON
C         KODE TABLE FOLLOWS
C         1--*  2--+  3---   4--INV   5--DET   6--TRANS   7--POWER
C         8--EIGEN  9--DIAGADD  10--DIAGSUB  11--DIAGMULT
C
C*************************************************************************
      COMMON/MATE/ NUM,ILAST,RMATRX(1500)
      KODE = KK
      IND = INM + 5
      IND1 = INM1 + 5
      NA = N1
      MA = M1
      GO TO (13,13,13,4,4,7,4,4,13,13,13) , KODE
   13 IND2 = INM2 + 5
      NB = N2
      MB = M2
      IF(MA.EQ.1 .AND. NA.EQ.1) GO TO 20
      IF(MB.EQ.1 .AND. NB.EQ.1) GO TO 21
      IF(KODE.LE.8) GO TO 12
      PRINT 102
  102 FORMAT('  OPERATOR NOT APPLICABLE FOR MATRICES.')
      GO TO 11
   12 GO TO (1,2,3,4,4,7,4,4) , KODE
    1 CALL MULMAT(RMATRX(IND1),RMATRX(IND2),RMATRX(IND),NA,MA,NB,MB,&11)
      NC = NA
      MC = MB
      GO TO 10
    2 CALL ADDMAT(RMATRX(IND1),RMATRX(IND2),RMATRX(IND),NA,MA,NB,MB,&11)
      NC = NA
      MC = MA
      GO TO 10
    3 CALL SUBMAT(RMATRX(IND1),RMATRX(IND2),RMATRX(IND),NA,MA,NB,MB,&11)
      NC = NA
      MC = MA
      GO TO 10
    4 IF(MA.EQ.NA) GO TO 5
   41 PRINT 101
  101 FORMAT('  OPERATION NOT DEFINED FOR NON-SQUARE MATRIX.')
      GO TO 11
    5 IF(KODE.EQ.5) GO TO 6
      IF(KODE.EQ.7) GO TO 8
      IF(KODE.EQ.8) GO TO 9
   51 CALL MATINV(RMATRX(IND1),RMATRX(IND),MA,&11)
      MC = MA
      NC = MA
      GO TO 10
    6 CALL DETER(RMATRX(IND1),RMATRX(IND),MA,DET,&11)
      RMATRX(IND) = DET
      MC = 1
      NC = 1
      GO TO 10
```

```
    7 MC = NA
      NC = MA
      CALL TRANS(RMATRX(IND1),RMATRX(IND),NA,MA)
      GO TO 10
    8 CALL EXPO(RMATRX(IND1),RMATRX(IND),RMATRX(1400),MA,INM2)
      MC = MA
      NC = MA
      GO TO 10
    9 IND2 = IND + MA
      CALL EIGEN(RMATRX(1400),RMATRX(IND1),RMATRX(IND),RMATRX(IND2),MA)
      MC = 2
      NC = NA
   10 RETURN
   11 RETURN 1
C
C         SCALAR IS INVOLVED
C         IT IS A
   20 ISCALE = 1
      NC = NB
      MC = MB
      GO TO 22
C         IT IS B
   21 ISCALE = 2
      NC = NA
      MC = MA
   22 GO TO (23,24,25,4,4,7,4,4,26,26,26) , KODE
   23 KODE2 = 3
      GO TO 30
   24 KODE2 = 1
      GO TO 30
   25 KODE2 = 2
      GO TO 30
C
   26 GO TO (27,28) , ISCALE
   27 IF(NB.NE.MB) GO TO 41
      GO TO 29
   28 IF(NA.NE.MA) GO TO 41
   29 KODE2 = KODE - 5
   30 GO TO (32,33) , ISCALE
   32 CALL SCALAR(RMATRX(IND1),RMATRX(IND2),RMATRX(IND),NB,MB,KODE2)
      GO TO 10
   33 CALL SCALAR(RMATRX(IND2),RMATRX(IND1),RMATRX(IND),NA,MA,KODE2)
      GO TO 10
      END
```

```
C
C
      SUBROUTINE ADDMAT(A,B,C,NA,MA,NB,MB,*)
C         PERFORMS MATRIX ADDITION
      LOGICAL PLUS
      DIMENSION A(NA,MA),B(NB,MB),C(NA,MA)
      PLUS = .TRUE.
```

```
    1 N1 = NA
      M1 = MA
      N2 = NB
      M2 = MB
      IF(M1.EQ.M2) GO TO 2
      PRINT 101
  101 FORMAT('  MATRICES HAVE DIFFERENT NO. OF COLUMNS. OPERATION IGNORE
     *D.')
      RETURN 1
    2 IF(N1.EQ.N2) GO TO 3
      PRINT 102
  102 FORMAT('  MATRICES HAVE DIFFERENT NO. OF ROWS. OPERATION IGNORED.'
     * )
      RETURN 1
    3 IF(PLUS) GO TO 5
      DO 4 I=1,M1
      DO 4 J=1,N1
    4 C(J,I) = A(J,I) - B(J,I)
      RETURN
    5 CONTINUE
      DO 6 I=1,M1
      DO 6 J=1,N1
    6 C(J,I) = A(J,I) + B(J,I)
      RETURN
      ENTRY SUBMAT(A,B,C,NA,MA,NB,MB,*)
C        PERFORMS MATRIX SUBTRACTION
      PLUS = .FALSE.
      GO TO 1
      END
```

```
C
C
      SUBROUTINE MULMAT(A,B,C,NA,MA,NB,MB,*)
C        PERFORMS MATRIX MULTIPLICATION
C        PERFORMS A*B NOT REPEAT NOT B*A
      DIMENSION A(NA,MA),B(NB,MB),C(NA,MB)
      N1 = NA
      M1 = MA
      N2 = NB
      M2 = MB
      IF(M1.EQ.N2) GO TO 1
      PRINT 101
  101 FORMAT('  FIRST NO. COLUMNS NOT EQUAL SECOND NO. ROWS.',/,
     * '  MATRIX MULTIPLY REQUEST IGNORED.')
      RETURN 1
    1 CONTINUE
      DO 2 I=1,M2
      DO 2 J=1,N1
    2 C(J,I) = 0.
      DO 3 I=1,M1
      DO 3 J=1,M2
      DO 3 K=1,N1
    3 C(K,J) = C(K,J) + A(K,I)*B(I,J)
      RETURN
      END
```

```
C
C
C         FORMS THE TRANSPOSE OF A
      SUBROUTINE TRANS(A,C,NA,MA)
      DIMENSION A(NA,MA),C(MA,NA)
      DO 1  I=1,MA
      DO 1  J=1,NA
    1 C(I,J) = A(J,I)
      RETURN
      END
```

```
C         PERFORMS MIXED ARITHMETIC
C         A IS THE SCALAR
C         KODE=1,2,3   SCALAR ADD,SUBTRACT,MULTIPLY
C         KODE=4,5,6   SCALAR DIAGONAL ADD,SUBTRACT,MULTIPLY
      SUBROUTINE SCALAR(A,B,C,NB,MB,KKK)
      DIMENSION B(NB,MB),C(NB,MB)
      KODE = KKK
      N = NB
      M = MB
      GO TO (5,10,15,20,20,20) , KODE
    5 DO 6 I=1,M
      DO 6 J=1,N
    6 C(J,I) = B(J,I) + A
      GO TO 50
   10 DO 11 I=1,M
      DO 11 J=1,N
   11 C(J,I) = B(J,I) - A
      GO TO 50
   15 DO 16 I=1,M
      DO 16 J=1,N
   16 C(J,I) = B(J,I) * A
      GO TO 50
C         DIAGONAL OPERATIONS
   20 DO 21 I=1,M
      DO 21 J=1,N
   21 C(J,I) = B(J,I)
      KODE = KODE - 3
      GO TO (22,25,30) , KODE
   22 DO 23 I=1,M
   23 C(I,I) = C(I,I) + A
      GO TO 50
   25 DO 26 I=1,M
   26 C(I,I) = C(I,I) - A
      GO TO 50
   30 DO 31 I=1,M
   31 C(I,I) = C(I,I) * A
   50 RETURN
      END
```

```
C
C          TAKES MATRIX A TO POWER IEXP, ANSWER IN B
      SUBROUTINE EXPO(A,B,C,NA,IE)
      DIMENSION A(NA,NA),B(NA,NA),C(NA,NA)
      IPOW = IE
      MA = NA
      DO 1 I=1,MA
      DO 1 J=1,MA
    1 B(J,I) = 0.0
      IF(IPOW.LE.0) GO TO 10
      IF(IPOW.EQ.1) GO TO 8
      DO 2 I=1,MA
      DO 2 J=1,MA
    2 C(J,I) = A(J,I)
C          INITIALIZATION COMPLETE
      IPOW = IPOW - 1
      DO 7 II=1,IPOW
C          PUT A*C IN B
      DO 3 I=1,MA
      DO 3 J=1,MA
      DO 3 K=1,MA
    3 B(K,J) = B(K,J) + A(K,I)*C(I,J)
C          ARE WE DONE
      IF(II.EQ.IPOW) GO TO 20
C          SAVE B IN C
      DO 4 I=1,MA
      DO 4 J=1,MA
    4 C(J,I) = B(J,I)
C          NOW PUT 0 IN B
      DO 5 I=1,MA
      DO 5 J=1,MA
    5 B(J,I) = 0.0
    7 CONTINUE
      GO TO 20
    8 DO 9 I=1,MA
      DO 9 J=1,MA
    9 B(J,I) = A(J,I)
      GO TO 20
C
C
C
   10 DO 11 I=1,MA
   11 B(I,I)=1.0
   20 RETURN
      END
```

```
C
C
C          INVERTS AA, INVERSE IN C
      SUBROUTINE MATINV(AA,C,MX,*)
```

```
C.....MATRIX INVERSION BY GAUSS-JORDAN ELIMINATION
      LOGICAL MATIN
      DOUBLE PRECISION A, B, PROD, Y
      DIMENSION AA(MX,MX),C(MX,MX),X(10,10)
      DIMENSION Y(10,10),Z(2,10,10),K(10)
      EQUIVALENCE (Y,Z)
      EQUIVALENCE (ABCDE,EDCBA)
      DATA DELT,EPS/.0001,.1E-7/
      MATIN=.TRUE.
      GO TO 1
C.....ENTRY FOR OBTAINING DETERMINANT OF INPUT MATRIX
      ENTRY DETER(AA,C,MX,DET,*)
      MATIN=.FALSE.
C.....INITIALIZE ROUTINE AND TEST MX (=ORDER OF MATRIX)
    1 M=MX
      ABCDE = EDCBA
      DO 50 I=1,M
      DO 50 J=1,M
   50 X(J,I) = AA(J,I)
      IF(M.GT.1) GO TO 5
    2 PROD=X(1,1)
      IF(PROD.NE.0.0) GO TO 4
    3 IF(MATIN) GO TO 25
      DET = 0.0
      RETURN
   25 PRINT 101
  101 FORMAT('  SINGULAR MATRIX. INVERSE NOT PERFORMED.')
      RETURN 1
    4 X(1,1)=1.0/PROD
      GO TO 23
    5 PROD=1.0
      MM=M-1
      DO 6 I=1,M
      K(I)=I
      DO 6 J=1,M
    6 Y(I,J)=X(I,J)
C.....BEGIN BY FINDING LARGEST PIVOTAL ELEMENT
      DO 11 I=1,M
      A = 0.0D0
      DO 7  J=I,M
      IF(DABS(Y(J,1)) .LE. A) GO TO 7
      A = DABS(Y(J,1))
      L=J
    7 CONTINUE
      IF(A .EQ. 0.0D0) GO TO 3
C.....REARRANGE ROWS AND ORDER ARRAY
      N=K(L)
      K(L)=K(I)
      K(I)=N
      DO 8  J=1,M
      A=Y(I,J)
      Y(I,J)=Y(L,J)
    8 Y(L,J)=A
C.....REDUCE PIVOTAL ROW
      A=Y(I,1)
      IF(.NOT.MATIN) PROD=PROD*A
```

```
      DO 9 J=1,MM
    9 Y(I,J)=Y(I,J+1)/A
      Y(I,M) = 1.0D0/A
C.....REDUCE REMAINING ROWS
      DO 11 L=1,M
      IF(L.EQ.I) GO TO 11
      A=Y(L,1)
      DO 10 N=1,MM
      Y(L,N)=Y(L,N+1)-A*Y(I,N)
   10 IF(DABS(Y(L,N)) .LT. (EPS*DABS(Y(L,N+1))))  Y(L,N) = 0.0D0
      Y(L,M)=-A*Y(I,M)
   11 CONTINUE
C.....UNSCRAMBLE INVERTED MATRIX
      DO 15 I=1,M
      IF(K(I).EQ.I) GO TO 15
      PROD=-PROD
      DO 12 J=I,M
      IF(K(J).EQ.I) GO TO 13
   12 CONTINUE
      GO TO 3
   13 DO 14 L=1,M
      A=Y(L,I)
      Y(L,I)=Y(L,J)
   14 Y(L,J)=A
      K(J)=K(I)
   15 CONTINUE
      IF(.NOT.MATIN) GO TO 23
C.....OBTAIN ERROR MATRIX
      TEST=0.0
      DO 17 I=1,M
      DO 17 J=1,M
      R=0.0
      DO 16 L=1,M
   16 R=R-Z(1,L,J)*X(I,L)
      IF(I.EQ.J) R = R + 1.0D0
      TEST = AMAX1(TEST,SNGL(DABS(R)))
   17 Z(2,I,J)=R
      DO 19 I=1,M
      DO 19 J=1,M
      A = 0.0D0
      DO 18 L=1,M
   18 A=A+Z(1,I,L)*Z(2,L,J)
   19 Z(1,I,J)=Z(1,I,J)+A
      IF(TEST.LE.DELT) GO TO 21
      PRINT 102
  102 FORMAT('  MATRIX ILL-CONDITIONED. INVERSE MAY BE INACCURATE.')
C.....TRANSFER FINAL INVERSE
   21 DO 22 I=1,M
      DO 22 J=1,M
   22 C (I,J) = Z(1,I,J)
   23 IF(M.EQ.1) C(1,1) = X(1,1)
      DET=PROD
      RETURN
      END
```

```
      SUBROUTINE EIGEN(A,B,EVR,EVI,NN)
      REAL*4 S(10),MAX
      REAL*8 T
      COMPLEX R(10)
      DIMENSION A(NN,NN),B(NN,NN),EVR(NN),EVI(NN)
      DATA EP/5.E-6/
      M = NN
      NR = 1
      DO 1 I=1,M
      DO 1 J=1,M
    1 A(J,I) = B(J,I)
      SUM = 0.
      DO 210 I=1,M
      DO 210 K=1,M
  210 SUM = SUM + ABS(A(K,I))
      EPS = EP*SUM/FLOAT(M*M)
    3 MM1 = M - 1
      NM1 = M
    4 N = NM1
      NM1 = N - 1
    5 IF(N.EQ.1) GO TO 125
      MAX = 0.
      DO 10 I=1,NM1
      IF(MAX.GE.ABS(A(N,I))) GO TO 10
      MAX = ABS(A(N,I))
      IC = I
   10 CONTINUE
      IF(MAX.LE.EPS) GO TO 125
      IF(IC.EQ.NM1) GO TO 40
      DO 20 K=1,N
      D = A(K,NM1)
      A(K,NM1) = A(K,IC)
   20 A(K,IC) = D
      DO 30 I=1,M
      D = A(NM1,I)
      A(NM1,I) = A(IC,I)
   30 A(IC,I) = D
   40 D = 1./A(N,NM1)
      DO 50 I=1,M
      S(I) = A(N,I)
   50 A(N,I) = -A(N,I)*D
      A(N,NM1) = D
      DO 70 K=1,M
      IF(K.EQ.NM1) GO TO 70
      E = A(N,K)
      DO 60 I=1,NM1
   60 A(I,K) = A(I,K)+ A(I,NM1)*E
   70 CONTINUE
      DO 80 I=1,NM1
   80 A(I,NM1) = A(I,NM1)*D
      DO 110 I=1,M
      T = 0.0D0
      DO 100 K=1,NM1
  100 T = T + S(K)*A(K,I)
  110 A(NM1,I) = T
      DO 120 I=NM1,MM1
  120 A(NM1,I) = A(NM1,I) + S(I+1)
      GO TO 4
  125 DO 130 I=N,M
```

```
  130 S(I) = -A(N,I)
      L = M - NM1
      CALL ROOTXX(S(N),R(NR),L)
      IF(N.EQ.1) GO TO 150
      NR = NR + L
      M = NM1
      GO TO 3
  150 DO 160 I=1,NN
      EVR(I) = REAL(R(I))
      EVI(I) = AIMAG(R(I))
  160 CONTINUE
      RETURN
      END
```

```
C        SUBROUTINE TO SOLVE POLYNOMIALS WITH ROOTS ORDER.LE.2
C
      SUBROUTINE ROOTXX(A,R,NN)
      COMPLEX CMPLX,CONJG,CSQRT,A11,A12,A21,A22,A31,A32,S1,S2,X,DX,
     *   TC,F0,F1,F2,R
      EQUIVALENCE (F0,A12),(F1,A22),(F2,A31),(TC,A11)
     *   ,(DX,A11),(S2,A32)
      COMPLEX CZERO / (0.,0.) /
      COMPLEX CONE / (1.,0.) /
      DIMENSION A(1),R(1)
      DATA MAX,EPS / 35,.5E-5 /
      N = NN
      IF(N.EQ.1) GO TO 150
      NP1 = N + 1
      DO 30 K=1,NN
      KK = K
      IF(A(N).NE.0.) GO TO 40
      N = N - 1
   30 R(K) = CZERO
   40 FN = N
      NM1 = N - 1
      FNM1 = NM1
      NP1 = N + 1
      KS = KK
      ST1 = A(N-1)/A(N)
      ST2 = ST1**2 - (A(N-2)*2.)/A(N)
      ASSIGN 110 TO M1
   45 IF(KK.GT.NN) RETURN
      KM1 = KK - 1
      ASSIGN 70 TO M2
C        BEGIN ITERATION LOOP
      DO 120 L=1,MAX
      GO TO M2,(70,50)
```

```
 50 A12 = CONE
    A22 = CONE
    A32 = CONE
    DO 60 I=1,NM1
    A11 = A12
    A12 = A(I) + X*A11
    A21 = A22
    A22 = A12 + X*A21
    A31 = A32
 60 A32 = A22 + X*A31
    F2 = A31+A31
    F0 = A(N) + X*A12
    IF(REAL(F0).EQ.0. .AND. AIMAG(F0).EQ.0.) GO TO 130
    S1 = F1/F0
    S2 = S1*S1 - F2/F0
    GO TO 80
 70 X = CZERO
    S1 = CMPLX(ST1,0.)
    S2 = CMPLX(ST2,0.)
    ASSIGN 50 TO M2
 80 GO TO M1,(110,90)
 90 DO 100 I=KS,KM1
    TC = 1.0/(X-R(I))
    S1 = S1 - TC
100 S2 = S2 - TC*TC
110 TC = FN/S1
    DX = TC/(1.0 + CSQRT(FNM1*(TC*S2/S1-1.0)))
    X = X - DX
    IF(CABS(DX)/CABS(X) .LE. EPS) GO TO 130
120 CONTINUE
    NN = KM1
    PRINT 201
201 FORMAT('  **WARNING**  EIGEN FAILED TO CONVERGE ON AN EIGENVALUE')
130 ASSIGN 90 TO M1
    R(KK) = X
    KK = KK + 1
    IF(AIMAG(X).EQ.0. .OR. KK.GT.NN) GO TO 45
    R(KK) = CONJG(X)
    KK = KK + 1
    GO TO 45
150 R(1) = CMPLX(-A(1),0.)
    RETURN
    END
```

REFERENCES

1. Anon.: IBM System/360 Time Sharing System Concepts and Facilities. IBM form C28-2003-2, Oct. 1967.
2. Anon.: IBM System/360 Time Sharing System/360 IBM FORTRAN IV. IBM form C28-2007-1, Oct. 1967.
3. Hadley, George: Linear Algebra. Addison-Wesley Publ. Co., Inc., 1961, pp. 162-165.
4. Danilevski, A., On the Numerical Solution of the Secular Equation. Mat. Sbornik., vol. 44, no. 2, 1937, p. 164.

NASA-Langley, 1969 — 8 E-4754

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D.C. 20546

OFFICIAL BUSINESS

FIRST CLASS MAIL

POSTAGE AND FEES PAID
NATIONAL AERONAUTICS AND
SPACE ADMINISTRATION

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Notes, and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION DIVISION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546