NASA TECHNICAL MEMORANDUM

NASA TM X-2417

c. 1

NASA TM X-2417

USERS MANUAL FOR THE VARIABLE DIMENSION AUTOMATIC SYNTHESIS PROGRAM (VASP)

by John S. White and Homer Q. Lee

Ames Research Center
Moffett Field, Calif. 94035

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • OCTOBER 1971

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM X-2417		

4. Title and Subtitle	5. Report Date
USERS MANUAL FOR THE VARIABLE DIMENSION AUTOMATIC SYNTHESIS PROGRAM (VASP)	October 1971
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
John S. White and Homer Q. Lee	A-3882
	10. Work Unit No.
9. Performing Organization Name and Address	125–19–20–02
NASA Ames Research Center Moffett Field, Calif. 94035	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, D. C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

VASP is a Variable dimension Fortran version of the Automatic Synthesis Program, a computer implementation of the Kalman filtering and control theory. It consists of 31 subprograms for analysis, synthesis, and optimization of complicated high-order time-variant problems associated with modern control theory. These subprograms include operations of matrix algebra, computation of the exponential of a matrix and its convolution integral, solution of the matrix Ricatti equation, and computation of dynamical response of a high-order system.

Since VASP is programmed in Fortran, the user has at his disposal not only the VASP subprograms, but all Fortran built-in functions and any other programs written in the Fortran language. All the storage allocation is controlled by the user so the largest system that the program can handle is limited only by the size of the computer, the complexity of the problem, and the ingenuity of the user. No accuracy was lost in converting the original machine language program to Fortran.

The principal part of this report contains a VASP dictionary and some example problems. The dictionary contains a description of each subroutine and instructions on its use. The example problems give the user a better perspective on the use of VASP for solving problems in modern control theory. These example problems include dynamical response, optimal control gain, solution of the sampled data matrix Ricatti equation, matrix decomposition, and pseudo inverse of a matrix. Listings of all subroutines are also included.

The VASP program has been further adapted to run in the conversational mode on the Ames 360/67 computer. The necessary procedures are given in appendix C.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Matrix computation Optimal control Kalman filtering	Unclassified – Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	167	$3.00

*For sale by the National Technical Information Service, Springfield, Virginia 22151

TABLE OF CONTENTS

USERS MANUAL FOR THE VARIABLE DIMENSION AUTOMATIC SYNTHESIS PROGRAM (VASP)

John S. White and Homer Q. Lee

Ames Research Center

SUMMARY

VASP is a *V*ariable dimension Fortran version of the *A*utomatic *S*ynthesis *P*rogram, a computer implementation of the Kalman filtering and control theory. It consists of 31 subprograms for analysis, synthesis, and optimization of complicated high-order time-variant problems associated with modern control theory. These subprograms include operations of matrix algebra, computation of the exponential of a matrix and its convolution integral, solution of the matrix Ricatti equation, and computation of dynamical response of a high-order system.

Since VASP is programmed in Fortran, the user has at his disposal not only the VASP subprograms, but all Fortran built-in functions and any other programs written in the Fortran language. All the storage allocation is controlled by the user so the largest system that the program can handle is limited only by the size of the computer, the complexity of the problem, and the ingenuity of the user. No accuracy was lost in converting the original machine language program to Fortran.

The principal part of this report contains a VASP dictionary and some example problems. The dictionary contains a description of each subroutine and instructions on its use. The example problems give the user a better perspective on the use of VASP for solving problems in modern control theory. These example problems include dynamical response, optimal control gain, solution of the sampled data matrix Ricatti equation, matrix decomposition, and pseudo inverse of a matrix. Listings of all subroutines are also included.

The VASP program has been further adapted to run in the conversational mode on the Ames 360/67 computer. The necessary procedures are given in appendix C.

INTRODUCTION

The VASP, the *V*ariable dimension Fortran version of the *A*utomatic *S*ynthesis *P*rogram, is the new Fortran IV version of the ASP, the Automatic Synthesis Program. It is intended to implement the Kalman filtering and control theory. Basically, it consists of 31 subprograms for solving most modern control problems in linear, time-variant (or time-invariant) control systems. These subprograms include operations of matrix algebra, computation of the exponential of a matrix and its convolution integral, and the solution of the matrix Riccati equation. The user calls these subprograms by means of a FORTRAN main program, and so can easily obtain solutions to most general problems of extremization of a quadratic functional of the state of the linear dynamical system.

Particularly, these problems include the synthesis of the Kalman filter gains and of the optimal feedback gains for minimization of a quadratic performance index.

The VASP is an outgrowth of ASP, which was developed for NASA under contract with the Research Institute for Advanced Studies, a division of the Martin Company. There are two urgent reasons for reprogramming ASP into the present Fortran version. First, ASP was programmed in FAP (Fortran Assembly Program) and could be used only on the IBM 7090-7094. Second, many complicated time-variant analysis, synthesis, and optimization problems tax the capability of the ASP and other programs written in the Fortran language. Fortran IV language makes the program adaptable to a much wider class of computers and expands its versatility.

The VASP is based extensively on a Fortran version of ASP, called FASP (Fortran ASP) by its programmer Mr. Don Kesler of Northrop, Norair.

Two basic questions the user will inevitably ask are:

(1) How accurate is VASP compared with ASP?
(2) What is the highest order of system that VASP can handle?

The answer to these questions depends on the number of significant digits carried by the user's computer and the amount of available storage in the computer. To answer the first question in a more concrete way, the check cases given in the ASP manual were duplicated and the results were compared with those in the manual. The accuracy of VASP was found to be the same as that of ASP. The second question can best be answered by first noting some of the basic differences between FASP and VASP. The pertinent difference between the two is that VASP has variable dimensioning and more efficient storage. To allow a computer to handle the highest order system possible, all matrix storage is assigned by the user's main program. Consequently, as an illustrative example, a 125,000-byte version of the IBM 360/50 can easily determine the solution of the matrix Riccati equation for a 30-order system (perhaps 40, depending on the size of other related matrices). Another basic difference between these two Fortran versions is that VASP diagnostics are more self-explanatory.

To recapitulate, the objectives of VASP are flexibility and versatility so that it can serve the maximum number of users. To achieve these goals FASP was revised extensively so as to have, for example, variable dimensioning, more efficient storage, and more self-explanatory diagnostics.

In this report, no attempt will be made to discuss any details of the theoretical background and the algorithms associated with the appropriate subprograms since they are well documented in the ASP manual, an NASA contractor report (NASA CR–475, 1966). This report does not repeat information from the contractor report, and the user is urged to consult that manual so that he may utilize VASP proficiently.

This program can be obtained from the centralized facility known as COSMIC, located at the Computer Software Management and Information Center, Barrow Hall, University of Georgia, Athens, Georgia, 30601.

FEATURES OF THE PROGRAM

The advantages of VASP over ASP are (1) a more versatile programming language, (2) a more convenient input/output format, (3) some new programs, and (4) variable dimensioning.

Since VASP is programmed in Fortran, it can be used in a very large class of machines. Moreover, as VASP is part of a larger main program, all the Fortran built-in functions are available to the main program. Furthermore, any subroutine available in the Fortran language may be used. In other words, the user has at his disposal the combined capabilities of VASP, Fortran built-in functions, and all other subroutines written in Fortran.

The input/output subroutines have been changed and now consist of READ, RDTITL, and PRNT. In addition, LNCNT has been added to control paging. The VASP allows the user the option of using the existing standard VASP format, or of supplying the output format of his own choice. For a more detailed explanation of how to exercise this latter option, see the dictionary entry under PRNT (p. 10), or Example 2.

Our experience with ASP is that certain groups of statements are often repeated. For the user's convenience, these groups of statements are incorporated as new subroutines in the VASP. They are AUG, UNITY, and SCALE. Detailed explanations of them are available in the VASP dictionary in this report.

To utilize the storage space as efficiently as possible, the subroutines are written with variable dimensioning, with the storage allocation controlled by the user's calling program. Consequently, it is necessary to provide some dummy storage space as a part of the argument of the subroutine. From the user's point of view, the price for efficient storage is inconvenience. All the subroutines are written in double precision for adequate accuracy; that is, *all matrix and scalar variables,* except integers, *are assumed to be in double precision.*

Universal Features

The arguments in the subroutines are arranged in the following order:

Input arguments
Output arguments
Dummy arguments

These are also arranged so that matrices occur before scalars.

An array of length two must be allocated by the user to store the dimensions of the matrix, and this array must be included in the subroutine call statements. For example, the add subroutine is called by

```
CALL ADD(A,NA,B,NB,C,NC)
```

and performs the matrix operation

$$C = A + B$$

Here NA, NB, and NC are arrays of length two which contain the dimensions of matrices A, B, C, respectively. In other words, the numbers of rows and columns of matrices A, B, and C are stored in NA, NB, and NC, respectively. Specifically, the number of rows of A is stored in NA(1) and the number of columns of A, in NA(2).

In general, the dimension array associated with an input matrix contains input information to the subroutine, while that associated with an output matrix contains output information. The dictionary shows the few cases where this rule is violated. In the example above, dimension arrays NA and NB[1] are inputs (since matrices A, B are inputs) and must be loaded before entering this subroutine. On the other hand, NC is an output, since C is an output. That is, the values of NC(1) and NC(2) are computed in the subroutine and are available to the calling program upon return.

When a dummy array is required, it must be appropriately dimensioned in the calling program. The required size is given in the appropriate dictionary entry.

Most of the routines check the array dimensions for compatibility and reasonableness, and check for adequate storage available in the DUMMY array. The "reasonableness" test is to see that all matrix dimensions are greater than zero, and that the product of the matrix dimensions is less than the constant MAXRC. In the program MAXRC has been set at 6400. It is recommended, however, that the user reset MAXRC to equal the size of his matrices, and thus prevent accidental overflowing of the matrix dimensions. If the matrices are incompatible or unreasonable, or if a mathematical error is found, a self-explanatory error message is printed, and no further computations are made in that subroutine. However, computation does go on to the subsequent steps, which are likely to be wrong also. After 10 such errors, the program is terminated.

The VASP program uses double-precision arrays, so that the user's main program must define each matrix to be double precision, and to contain a sufficient number of cells to accommodate the matrix. The dimension statement may classify the array as one- or two-dimensional, or even more.

For example, to use the matrix A, which is a 5 X 5 matrix, any of the following dimension statements will be adequate:

```
DOUBLE PRECISION A(25)
DOUBLE PRECISION A(5,5)
DOUBLE PRECISION A(3,10)
DOUBLE PRECISION A(100)
```

The important factor is the total number of cells reserved, and the user may reserve more cells than the matrix requires, or, conversely, he may put a smaller matrix than originally planned in a specific array. The VASP program stores data in an array as a string of columns, just as Fortran does.

---

[1] The convention used here, and throughout this report, is that the name of a dimension array is obtained by prefixing the letter N to the matrix name.

However, it stores the matrix A according to the dimension given in NA, whereas Fortran stores A according to the dimensions in the Fortran dimension statement.[2]

Consider the following example. The Fortran statements are:

```
DOUBLE PRECISION A(5,5), B(5,5), C(5,5)
DIMENSION NA(2), NB(2), NC(2)
CALL READ(3,A,NA,B,NB,C,NC, . . . )
```

The first card in the data deck specifies NA = 5,5, followed by cards with 25 data words for A; then NB = 4,4, followed by 16 data words of B; finally, NC = 6,6, followed by 36 data words of C. Since the storage of data in VASP is controlled by the VASP dimensioning, the 25 words for A will exactly fill the reserved storage and the 16 words for B will fill the first 16 cells of the storage reserved for B. The 36 words for C will completely fill the reserved storage for C *and overflow into something else.* The user can prevent this overflow by setting the test constant MAXRC to 25. The error test in the READ subroutine will note that the product of NC(1) and NC(2) is greater than MAXRC, and will return an error message. This selection of MAXRC will limit all other VASP arrays to 25, so it is frequently desirable to dimension all arrays the same.

Occasionally the user may wish to refer to a single element of a matrix. Since FORTRAN statements use the FORTRAN dimension statement, a reference to B(4,4) in the previous example will select the 19 element in the B storage. However, VASP, using the VASP dimension, has stored B(4,4) in the 16 element of B, which is not the same. To actually select a specific element, say B(i,j), one must refer to B((j–1)*NB(2)+i,1). In the above example, references to A(i,j) will be correct, since the FORTRAN and VASP dimensions are the same.

System-Dependent Features

Two subroutines in the VASP package are system dependent. The first is BLKDTA. Data statements in this subroutine control the printing. They require a printer with at least 115 characters per line, and place 45 lines on each page. These requirements may be changed as needed. The second is ASPERR, which calls a system subroutine for error tracing. The description of ASPERR indicates any necessary changes to match the system.

The VASP programs frequently generate very small numbers. The computer operating system may detect these small numbers as underflows, and print error messages. If so, the user should arrange to turn off the underflow error messages.

THE VASP DICTIONARY

A detailed description of all the subroutines is given in this dictionary. Each entry is organized into subheadings that describe the subroutine and explain how to use it. Other

[2] The storage in VASP is also compatible with the storage of "general" matrices in the IBM scientific subroutine package.

subheadings, such as motivation and remarks, are sometimes added to offer the user a better understanding of the theoretical background of the subroutine.[3]

The dictionary proper lists only those routines that the user is expected to call directly. Several additional subroutines, used internally, are also a part of VASP. The user may, however, wish to call these routines himself, since they are quite flexible. These additional routines are described in appendix A, and a complete listing of all programs is given in appendix B.

Several procedures have been written to facilitate the use of VASP on Ames time-share system. Their usage and listings are given in appendix C.

Table 1 lists all subroutines with their calling sequence, and the TSS procedures, for easy reference, while table 2 lists the approximate storage used by each of the VASP subroutines when compiled on the NASA Ames 360/67, OS system. Table 2 also lists the external references for each subroutine.

---

[3]Some of the subroutines are almost direct copies from the Northrop FASP. The detailed description of the theory is either obvious, or is given in the ASP manual (NASA CR–475). Other routines were written by one of the authors. These were quite simple, and needed little description. Subroutines ANDRA, BDNRM, DECOM, and PSEU were written by John Andrews, Information Systems Company. Since no description of these subroutines is available elsewhere, a detailed description of their theory and usage is included. Because there were various programmers, the nomenclature internal to the various subroutines is not completely consistent.

TABLE 1.– SUBROUTINE CALL STATEMENTS IN VASP

1.	CALL READ(I,A,NA,B,NB,C,NC,D,ND,E,NE)
2.	CALL RDTITL
3.	CALL PRNT(A,NA,NAM,IP)
4.	CALL LNCNT(N)
5.	CALL ADD(A,NA,B,NB,C,NC)
6.	CALL SUBT(A,NA,B,NB,C,NC)
7.	CALL MULT(A,NA,B,NB,C,NC)
8.	CALL SCALE(A,NA,B,NB,S)
9.	CALL TRANP(A,NA,B,NB)
10.	CALL INV(A,NA,DET,DUM)
11.	CALL NORM(A,NA,ANØRM)
12.	CALL UNITY(A,NA)
13.	CALL TRCE(A,NA,TR)
14.	CALL EQUATE(A,NA,B,NB)
15.	CALL JUXTC(A,NA,B,NB,C,NC)
16.	CALL JUXTR(A,NA,B,NB,C,NC)
17.	CALL EAT(A,NA,TT,B,NB,C,NC,DUMMY,KDUM)
18.	CALL ETPHI(A,NA,TT,B,NB,DUMMY,KDUM)
19.	CALL AUG(F,NF,G,NG,RI,NRI,H,NH,Q,NQ,C,NC,Z,NZ,II)
20.	CALL RICAT(PHI,NPHI,C,NC,NCONT,K,NK,PT,NPT,DUM,KDUM)
21.	CALL SAMPL(PHI,NPHI,H,NH,Q,NQ,R,NR,P,NP,K,NK,NCONT,DUM,KDUM)
22.	CALL TRNSI(F,NF,G,NG,J,NJ,R,NR,K,NK,H,NH,X,NX,T,DUMMY,KDUM)
23.	CALL PSEUDO(A,NA,B,NB,DUM,KDUM)
24.	CALL DECGEN(R,NR,G,NG,H,NH,DUM,KDUM)
24a.	CALL DECSYM(R,NR,G,NG,H,NH,DUM,KDUM)

Programs 25 through 31 are called internally and need not be used by the programmer. They are described in appendix A.

25.	CALL READ1(A,NA,NZ,NAM)
26.	CALL ASPERR
27.	BLKDATA (nonexecutable)
28.	CALL PSEU(A,B,C,EE,DEP,IP,D)
28a.	CALL PSEUP(A,B,C,EE,DEP,IP,D)
29.	FUNCTIØN BDNRM(NR,CT,EE,D,KRV)
29a.	CALL TTRM(NR,CT,EE)
30.	CALL ANDRA(B,T,DPR,JP)
31.	CALL DECOM(A,B,C,E,JL,DCM,KP,D)

The remainder of the items are procedures to facilitate the use of VASP on the Ames TSS.

32.	VASP$$ [inputdsname] [,outputdsname]
33.	CHNGIN [inputdsname]
34.	CHNGOUT [outputdsname]
35.	CMPL
36.	CLRVASP
37.	CONVASP [matrixsize] [,$A=name] [,$B=name] [,$C=name] [,$W=name] [,$X=name] [,$Y=name] [,$Z=name]
38.	RECMPT
39.	REWRT [n]

TABLE 2.– APPROXIMATE STORAGE REQUIREMENTS AND EXTERNAL REFERENCES

Subroutines	Storage decimal bytes	External references
1. READ	1000	READ1, PRNT*
2. RDTITL	400	LNCNT
3. PRNT	1400	*
4. LNCNT	500	None
5. ADD	800	*
6. SUBT	800	*
7. MULT	1100	*
8. SCALE	700	*
9. TRANP	800	*
10. INV	2500	*
11. NORM	1000	*
12. UNITY	700	SCALE*
13. TRACE	700	*
14. EQUATE	700	*
15. JUXTC	1000	*
16. JUXTR	1100	*
17. EAT	3200	ADD, MULT, SCALE, NORM, UNITY, EQUATE*
18. ETPHI	2300	ADD, MULT, SCALE, NORM, UNITY, EQUATE*
19. AUG	3300	MULT, TRANP, EQUATE*
20. RICAT	5100	ADD, MULT, INV, EQUATE, PRNT*
21. SAMPL	3700	ADD, SUBT, MULT, TRANP, PSEUDO, PSEU, BDNRM, ANDRA, PRNT*
22. TRNSI	5000	ADD, EAT, SUBT, MULT, PRNT, EQUATE*
23. PSEUDO	900	PSEU, BDNRM, ANDRA*
24. DECGEN	2600	MULT, TRANP, INV, NORM, EQUATE, DECOM, PSEU, BDNRM, ANDRA*
25. READ1	800	PRNT*
26. ASPERR	400	None
27. BLKDATA	None	None
28. PSEU	5900	MULT, NORM, BDNRM, ANDRA*
29. BDNRM	1500	MULT, NORM
30. ANDRA	2000	LNCNT
31. DECOM	1500	MULT, NORM, PSEU, BDNRM, ANDRA*
COMMON/MAX/ COMMON/LINES/ COMMON/FORM/	200	
TOTAL	53,600	

*LNCNT and ASPERR are additional external references.

1. READ

DESCRIPTION

This subroutine reads 1 to 5 matrices from cards, along with the names and dimensions, and prints the same information. For each matrix the routine first reads a header card containing a four-character title, followed by two integers giving the row and column size of the matrix, using format (A 4, 4x, 2I4). Then the matrix data are read using READ1, each row of the matrix starting on a new card, using format (8F10.2). If the card data is in exponential form, it must use a D exponent. The matrix title and the matrix are then printed using subroutine PRNT.

If the header card contains no row and column size (i.e., n = 0), then the matrix in storage is unchanged, no data cards are read for that matrix, and the previously stored matrix is printed.

USAGE

```
CALL  READ(I,A,NA,B,NB,C,NC,D,ND,E,NE)
```

Input Arguments

Control constant: I

where I is an integer from 1 to 5 and indicates the number of matrices to be read. If I is less than 5, the extra matrices in the call list may be dummy variables, or repeated references to the same matrix; for example,

```
CALL  READ(1,A,NA,A,NA,A,NA,A,NA,A,NA)
```

Output Arguments

Matrices:	The first I of the matrices A,B,C,D,E
Dimension arrays:	The first I of the arrays NA,NB,NC,ND,NE

2. RDTITL

DESCRIPTION

This subroutine reads a single card in hollerith format, and loads it into the array TITLES. It then calls LNCNT(100). This latter program in turn skips the printer to the next page, prints the hollerith information in the array TITLES, and positions the output to print next on line 3.

USAGE

```
CALL  RDTITL
```

It has no arguments.

3. PRNT

DESCRIPTION

This subroutine prints a single matrix, with or without a title line, and either on the same page or a new page. The matrix is printed using format (1P7D16.7) for the first line, and (3x,1P7D16.7) for all subsequent lines. The "3x" indents continuation lines for easier reading.

REMARKS

The standard format is stored in arrays FMT1 (for the first line) and FMT2 (for subsequent lines) both of which are stored in labeled COMMON as follows:

COMMON/FORM/NEPR, FMT1(6), FMT2(6)

where NEPR is the number of columns of data to be printed (7, in the standard case). The standard format is loaded into COMMON in the BLKDATA program. If other formats are desired, they can be obtained either by changing the BLKDATA program, or by having the users main program change the contents of COMMON.

CAUTION

In writing a data statement for the formats, put FMT1 and FMT2 in separate statements, as in the BLKDATA program. If they are loaded in one statement, they will probably load incorrectly, because of the dimensionality of FMT1 and FMT2. Also NEPR must be consistent with the numbers in FMT1 and FMT2.

USAGE

CALL PRNT(A,NA,NAM,IP)

Input Arguments

Matrix:	A
Dimension array:	NA
Matrix name:	NAM If NAM = 0, a blank name will be printed. NAM should contain four hollerith characters. It can be written in the calling sequence as 4HAbbb. If written 1HA, the last three characters of the printed name will be garbage.
Control constant:	IP = 1 heading, same page 2 heading, new page 3 no heading, same page 4 no heading, new page

Output Arguments

None

4. LNCNT

DESCRIPTION

This subroutine keeps track of the number of lines printed, and automatically pages the output as required. It is completely internal, and the user need not refer to it unless he has WRITE statements of his own. In that case, the user may (should) put the statement CALL LNCNT(N) before each WRITE statement, where N is the number of lines to be printed.

Page length is controlled by the variable NLP set in the BLOCK DATA program to 45. This is an installation-dependent variable, and may be changed as necessary.

The subroutine provides one line of print at the top of each page. This line contains 92 characters, of which the first 72 are available for the programmers use and may be loaded by use of RDTITL. The remainder contain "VASP PROGRAM." The 92 characters are contained in the array TITLES, which is, in turn, contained in the COMMON area LINES. If $N > NLP$, the printer will automatically skip to the top of the next page, and print the title line.

USAGE

CALL LNCNT(N)

Input Arguments

Constant N = number of lines to be printed

Output Arguments

None

5. ADD

DESCRIPTION

This subroutine computes the matrix sum

$$C = A + B$$

USAGE

CALL ADD(A,NA,B,NB,C,NC)

Input Arguments

Matrices:	A,B
Dimension arrays:	NA,NB

Output Arguments

Matrices:	C
Dimension array:	NC

REMARK

Matrices A and C may share the same storage space or matrices B and C may share the same storage space.

6. SUBT

DESCRIPTION

This subroutine computes the matrix difference

$$C = A - B$$

USAGE

CALL SUBT(A,NA,B,NB,C,NC)

Input Arguments

Matrices:	A,B
Dimension arrays:	NA,NB

Output Arguments

Matrices:	C
Dimension array:	NC

REMARK

Matrices A and C may share the same storage space or matrices B and C may share the same storage space.

7. MULT

DESCRIPTION

This subroutine computes the matrix product

$$C = A \quad B$$

USAGE

CALL MULT(A,NA,B,NB,C,NC)

Input Arguments

Matrices:	A,B
Dimension arrays:	NA,NB

Output Arguments

Matrix:	C
Dimension array:	NC

8. SCALE

DESCRIPTION

This subroutine multiplies every element of matrix A by S and stores the resulting value in B, that is,

$$B_{ij} = S \quad A_{ij}$$

where S is a scalar.

USAGE

CALL SCALE(A,NA,B,NB,S)

Input Arguments

Matrix:	A
Dimension array:	NA
Scalar:	S

Note: If S is a constant, it must be written as a double precision constant (i.e., 2.D0, 0.D0, etc.).

Output Arguments

Matrix:	B
Dimension array:	NB

Note: A and B can be the same matrix.

9. TRANP

DESCRIPTION

This subroutine rearranges the elements of matrix A so that

$$B = A'$$

or

$$B_{ij} = A_{ji}$$

USAGE

CALL TRANP(A,NA,B,NB)

Input Arguments

Matrix:	A
Dimension array:	NA

Output Arguments

Matrix:	B
Dimension array:	NB

10. INV

DESCRIPTION

This subroutine computes the matrix inverse of A and stores this inverse in A, that is,

$$A = A^{-1}$$

Note that after the inversion is performed, the values stored in the original matrix A are destroyed and replaced by the corresponding elements of its inverse.

USAGE

CALL INV(A,NA,DET,DUM)

Input Arguments

Matrix:	A
Dimension array:	NA

Output Arguments

Matrix:	A, the inverse of the original A
Scalar:	DET, the determinant of A

Dummy Argument

Matrix:	DUM, work vector of length 2*NA(1)

This subroutine is a slightly modified copy of the inverse routine given in the IBM scientific subroutine package.

11. NORM

DESCRIPTION

This subroutine computes the norm of the matrix A as follows:

$$\|A\| = \min\left(\max_j \sum_i A_{ij}, \max_i \sum_j A_{ij}\right)$$

USAGE

CALL NORM(A,NA,ANORM)

Input Arguments

Matrix:	A
Dimension array:	NA

Output Arguments

Scalar:	ANORM

12. UNITY

DESCRIPTION

This subroutine computes the unit matrix

$$A = I$$

USAGE

CALL UNITY(A,NA)

Input Argument

Dimension array: NA

Output Argument

Matrix: A

13. TRCE

DESCRIPTION

This subroutine computes the trace of the matrix A

$$TR = \sum_{i=1}^{n} a_{ii}$$

USAGE

```
CALL TRCE(A,NA,TR)
```

Input Arguments

Matrix: A
Dimension array: NA

Output Argument

Scalar: TR

14. EQUATE

DESCRIPTION

This subroutine copies the values stored in matrix A into matrix B as follows:

$$B = A$$

USAGE

```
CALL EQUATE(A,NA,B,NB)
```

Input Arguments

Matrix: A
Dimension array: NA

Output Arguments

Matrix:	B
Dimension array:	NB

15. JUXTC

DESCRIPTION

This subroutine takes the $m \times n$ matrix A, the $m \times p$ matrix B, and forms the $m \times (n+p)$ matrix

$$C = [A \quad B]$$

USAGE

```
CALL  JUXTC(A,NA,B,NB,C,NC)
```

Input Arguments

Matrices:	A,B
Dimension arrays:	NA,NB

Output Arguments

Matrix:	C
Dimension array:	NC

16. JUXTR

DESCRIPTION

This subroutine takes the $m \times n$ matrix A, the $p \times n$ matrix B, and forms the $(m+p) \times n$ matrix

$$C = \begin{bmatrix} A \\ B \end{bmatrix}$$

USAGE

```
CALL  JUXTR(A,NA,B,NB,C,NC)
```

Input Arguments

Matrices:	A,B
Dimension arrays:	NA,NB

Output Arguments

Matrix:	C
Dimension array:	NC

17. EAT

DESCRIPTION

This subroutine computes

$$B = e^{At} \qquad (*)$$

and

$$C = \int_o^t e^{A\tau}\, d\tau \qquad (*)$$

For a linear time-invariant system, the system equation is

$$\dot{x} = Ax + Gu$$

Then,

$$x(t) = e^{At}x_o + \left(\int_o^t e^{A\tau}\, d\tau\right) Gu$$

or

$$x(t) = Bx_o + CGu$$

See ASP manual, page 92, for reference.

USAGE

```
CALL  EAT(A,NA,TT,B,NB,C,NC,DUMMY,KDUM)
```

Input Arguments

Matrix:	A
Dimension array:	NA
Scalar:	TT

where TT is the value of t used in equations *

Output Arguments

Matrices:	B,C
Dimension arrays:	NB,NC

Dummy Arguments

Matrix:	DUMMY
Constant:	KDUM

Note: KDUM contains the size of the DUMMY matrix, which must be at least 2*NA(1)*NA(2).

18. ETPHI

DESCRIPTION

This subroutine computes the matrix exponential

$$B = e^{At}$$

See ASP manual, page 92, and also EAT, page 18 of this manual for reference.

USAGE

CALL ETPHI(A,NA,TT,B,NB,DUMMY,KDUM)

Input Arguments

Matrix:	A
Dimension array:	NA
Scalar:	TT

where TT is the final value of time.

Output Arguments

Matrix:	B
Dimension array:	NB

Dummy Arguments

Matrix:	DUMMY
Constant:	KDUM

Note: KDUM contains the size of the DUMMY matrix, which must be at least 2*NA(1)*NA(2).

19. AUG

DESCRIPTION

This subroutine computes

$$C = R^{-1}G'$$

and

$$Z = \begin{bmatrix} -F & +GR^{-1}G' \\ +H'QH & +F' \end{bmatrix}$$

The matrices C and e^{Zt} are then used in RICAT to calculate the covariance and weighting matrices.

These matrices arise from a linear system of the form

$$\dot{x} = Fx + Gu$$

with output equation

$$y = Hx$$

and cost function

$$J = \int (x'H'QHx + u'Ru)dt$$

See ASP manual, page 212, for reference.

In the special case where

$$y = x$$

then,

$$Z = \begin{bmatrix} -F & +GR^{-1}G' \\ Q & +F' \end{bmatrix}$$

and the cost function is

$$J = \int (x'Qx + u'Ru)dt$$

A control index II is used to distinguish the two cases.

REMARKS

The inputs to this program are the matrices F, G, RI, H, Q.

(a) F must be square.
(b) Q, R must be symmetric.
(c) R must be invertible.

The Fortran symbol for R^{-1} is RI.

USAGE

CALL AUG(F,NF,G,NG,RI,NRI,H,NH,Q,NQ,C,NC,Z,NZ,II)

Input Arguments

Matrices:	F,G,RI,H,Q	
Dimension arrays:	NF,NG,NRI,NH,NQ	
Control constant:	II	
	II ≠ 1	General case
	II = 1	Special case, H is not used in AUG

Output Arguments

Matrices:	C,Z
Dimension arrays:	NC,NZ

20. RICAT

DESCRIPTION

This subroutine computes P(t) and K(t) by the following equations:

$$P(t+\tau) = [\theta_{21} + \theta_{22}P(t)]\,[\theta_{11} + \theta_{12}P(t)]^{-1}$$

$$K(t) = CP(t)$$

See ASP manual, page 9, for reference.

MOTIVATION

A standard control problem will be used to illustrate how this matrix Riccati equation arises. Given the system equation,

$$\dot{x} = Fx + Gu$$

the output equation,

$$y = Hx$$

and the performance index,

$$J = \int_{o}^{T} (x'H'QHx + u'Ru)dt + x'(T)H'S(T)Hx(T)$$

where Q,R,S are symmetric matrices and R is invertible. We wish to find a control law which minimizes the performance index J. Introducing the auxiliary variable $\lambda(t)$ into the system of equation, we have the following Euler-Lagrange equations,[4]

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} F & -GR^{-1}G' \\ -H'QH & -F' \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = -Z \begin{bmatrix} x \\ \lambda \end{bmatrix}$$

which have for a solution

$$\begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix} = e^{Zt} \begin{bmatrix} x_o \\ \lambda_o \end{bmatrix} = \theta \begin{bmatrix} x_o \\ \lambda_o \end{bmatrix}$$

The optimal control law is

$$u(t) = R^{-1}G'\lambda(t)$$

Letting $P(t)$ be a linear transformation from the state variable $x(t)$ to the auxiliary variable $\lambda(t)$, that is,

$$\lambda(t) = P(t)x(t)$$

we obtain from the Euler-Lagrange equation the following Riccati equation,

$$-\dot{P} = F'P + PF - PGR^{-1}G'P + H'QH$$

where the initial condition for this differential equation is

$$P(t) = H'S(T)H$$

The optimal control, in terms of the state variable $x(t)$, is

$$u(t) = -K(t)\,x(t) = -R^{-1}G'P(t)x(t)$$

---

[4] AUG computes Z rather than $-Z$, so that the exponentiation for θ uses positive time increments.

and the optimal feedback gain K(t) is

$$K(t) = -R^{-1}G'P(t)$$

Letting

$$C = R^{-1}G'$$

then,

$$K(t) = CP(t)$$

REMARKS

1. This subroutine will be terminated when

$$\left[\sum_{i=1}^{n} \left|P_{ii}(t+\tau) - P_{ii}(t)\right|\right] \Big/ \left[\sum_{i=1}^{n} \left|P_{ii}(t+\tau)\right|\right] \leqslant \epsilon \text{ where } \epsilon = 10^{-5}$$

or NCØNT(2) steps have been taken.

2. Matrices P(t) and K(t) will be printed out every NCONT(1) steps, as controlled by NCONT(3).

3. Matrices $\theta_{11}, \theta_{12}, \theta_{21}, \theta_{22}$ are submatrices of θ. Their dimensions are n X n where n is the order of the system (i.e., the dimension of the F matrix). They are partitioned from the θ matrix as follows:

$$\theta = \left[\begin{array}{c|c} \theta_{11} & \theta_{12} \\ \hline \theta_{21} & \theta_{22} \end{array}\right]$$

The Fortran symbol for θ is PHI.

USAGE

```
CALL  RICAT(PHI,NPHI,C,NC,NCONT,K,NK,PT,NPT,DUM,KDUM)
```

Input Arguments

Matrices:	PHI, C, PT	
Dimension arrays:	NPHI,NC,NPT	
Control array:	NCONT(1)	Number of steps per print
	NCONT(2)	The maximum number of steps
	NCONT(3)	Printout control 1 → no P, no K 2 → P only 3 → K only 4 → P and K

Output Arguments

Matrices:	K,PT
Dimension array:	NK

Dummy Arguments

Matrix:	DUM
Constant:	KDUM

Note: KDUM contains the size of the DUMMY matrix which must be at least NPHI(1)**2.

Note: PT is used for both input and output arguments. The initial value of P must be placed in PT before calling the subroutine. The value of P is updated every iteration in the subroutine until the final P is reached. This final P is one of the outputs of the subroutine.

21. SAMPL

DESCRIPTION

Subroutine SAMPL calculates the covariance and weighting matrices associated with the discrete case of either the control problem or the filter problem.

Consider the following filter problem.

> Given the system $x_{i+1} = \phi x_i + u$ where u = gaussian random sequence with variance = Q, and observations $y_i = Hx_i + v$ where v = gaussian random variable with variance = R.

The optimum estimate of the state is (see p. 234 in the ASP manual)

$$\hat{x}_{i+1} = \phi \hat{x}_i + K_i(y_i - H\hat{x}_i)$$

where

$$\left.\begin{aligned} K_i &= \phi P_i H^T (HP_i H^T + R)^{\#} \\ P_i^+ &= P_i - P_i H^T (HP_i H^T + R)^{\#} HP_i \end{aligned}\right\} \quad \# = \text{pseudo inverse}$$

$$P_{i+1} = \phi P_i^+ \phi^T + Q$$

Here P_i is the solution of the matrix Ricatti equation, which is obtained by SAMPL. The subroutine has for inputs ϕ, H, Q, R, P_i, and for output, P_{i+n} and K_{i+n-1} where P_{i+n} is written over P_i.

REMARKS

1. The routine will take n steps at a single call where n is an input parameter. Further, if P becomes constant, then the routine will stop and exit before completing the n steps. The actual test is as follows:

$$\text{If } \alpha = \left[\sum_k |P_{kk}(i+1) - P_{kk}(i)|\right] \Big/ \left[\sum_k |P_{kk}(i+1)|\right] \leqslant 10^{-5} \text{ then exit.}$$

2. The routine will print the value of P_i and/or K_{i-1} every j steps, and also when either exit occurs. NCONT(3) controls which arrays are printed.

USAGE

CALL SAMPL(PHI,NPHI,H,NH,Q,NQ,R,NR,P,NP,K,NK,NCONT,DUM,KDUM)

Input Arguments

Matrices:	PHI,H,Q,R,P
Dimension arrays:	NPHI,NH,NQ,NR,NP
Control arrays:	NCONT
	NCONT(1) = j = number of steps per print
	NCONT(2) = n = maximum number of steps
	NCONT(3) = print control 1 no print 2 print P only 3 print K only 4 print both P and K

Output Arguments

Matrices:	P,K
Dimension arrays:	NP,NK

Dummy Arguments

Matrix:	DUM
Constant:	KDUM

Note: KDUM contains the size of the DUM matrix, which must be at least 6*NPHI(1)*NPHI(2).

22. TRNSI

This subroutine computes

$$x(t) = e^{Ft}x(t_o) + \left(\int_{t_o}^{t} e^{F\tau}\, d\tau\right) Gu$$

where

$$u(t) = JR - Kx(t_o + it_2)$$

and u is held constant for any interval specified by

$$it_2 \leqslant t - t_o < (i+1)t_2 \quad i = 0, 1, 2, \ldots$$

The system output y(t) is given by

$$y(t) = Hx(t)$$

The state vector x and system outputs y are printed every t_1 intervals. Also t_2 must be a positive integral multiple of t_1. The program terminates at $t \geqslant t_f$.

See ASP manual, pages 120-121, for reference.

USAGE

CALL TRNSI(F,NF,G,NG,J,NJ,R,NR,K,NK,H,NH,X,NX,T,DUMMY,KDUM)

Input Arguments

Matrices:	F,G,J,R,K,H,X,T
Dimension arrays:	NF,NG,NJ,NR,NK,NH,NX

Note: Dimension of T is 4 where

T1	t_1
T2	t_2
T3	t_f
T4	t_o

Dummy Argument

Matrix:	DUMMY
Constant:	KDUM

Note: KDUM contains the size of the dummy matrix, which must be at least 4*NF(1)*NF(2).

23. PSEUDO

DESCRIPTION

This subroutine computes the Moore-Penrose generalized inverse of the input matrix. It sets up a standard set of options for use by PSEU, which does the actual inversion. For details of the method, see PSEU, p.70.

USAGE

CALL PSEUDO(A,NA,B,NB,DUM,KDUM)

Input Arguments

Matrix:	A
Dimension array:	NA

Output Arguments

Matrix:	$B = A^{\#}$
Dimension array:	NB

Dummy Arguments

Matrix:	DUM
Constant:	KDUM

Note: KDUM contains the size of the dummy matrix, which must be at least 3*NA(1)*NA(2).

24. DECGEN
24a. DECSYM

DESCRIPTION

This subroutine decomposes a real matrix R with dimensions $m \times n$ and rank $r \leqslant \min(m,n)$ into two matrices H and G such that R = HG. Further, both H and G are of maximal rank, with dimensions $m \times r$ and $r \times n$, respectively. It uses subroutine DECOM to provide matrices from which H and G can be computed. The writeup of DECOM, p. 85, describes the method in detail. Subroutine DECOM requires for input a matrix A which is positive semidefinite symmetric. Subroutine DECGEN computes this matrix by letting $A = RR^T$ or R^TR, whichever is smaller, and uses the former if R is square. If the user knows that R is already positive semidefinite symmetric, this step may be omitted by a call to DECSYM, in which case A = R.

USAGE

CALL DECGEN(R,NR,G,NG,H,NH,DUM,KDUM)

if R is general, or

CALL DECSYM(R,NR,G,NG,H,NH,DUM,KDUM)

if R is positive semidefinite symmetric.

Input Arguments

Matrix:	R
Dimension array:	NR

Output Arguments

Matrices:	H,G
Dimension arrays:	NH,NG

Dummy Arguments

Matrix:	DUM
Constant:	KDUM

Note: KDUM contains the size of the DUM array, which must be at least $7*\min(NR(1)^2, NR(2)^2)$.

EXAMPLE USES OF VASP PROGRAM

The examples given demonstrate directly the use of the principal subroutines EAT, ETPHI, AUG, RICAT, SAMPL, DECGEN, and PSEUDO. In addition, they exercise all of the subroutines except TRCE. They can be used to indicate whether the programs are working properly. They do not, however, provide an exhaustive test of the VASP program.

The first example discusses the user's main program in great detail to explain some of the system features. The remainder of the examples simply state the problem, and present the main program listing, the data listings, and the results.

Example 1 – Transient Response

A set of equations for a linear plant can be written as:

$$\dot{x}(t) = Fx(t) + Gu(t), \quad x(0) = x_o$$

$$y(t) = Hx(t)$$

where x, u, and y are, respectively, the state, control, and observation vectors. The system, distribution and observation matrices are F, G, and H, respectively. It is known that

$$x(t) = e^{Ft}x_o + \int_o^t e^{F(t-\tau)}G(\tau)u(\tau)d\tau$$

is the solution for x(t). If G and u are constant, then

$$x(t) = e^{Ft}x_o + \int_o^t e^{F(t-\tau)}\, d\tau\ Gu$$

By letting $s = t - \tau$ the integral becomes

$$\int_t^o e^{Fs} d(-s) = \int_o^t e^{Fs}\, ds$$

Thus, the solution to the system equation can be written

$$x(t) = Bx_o + CGu$$

$$y(t) = Hx(t)$$

where

$$B = e^{Ft}$$

and

$$C = \int_o^t e^{Fs}\, ds$$

It is desired to generate the transient response of such a system in response to a given initial condition x_o and fix control u. In particular, given

$$F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad x_o = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

find x(t) for $0 \leqslant t < 2.0$. Also print x(t) and y(t) every 0.01 second.

The user's main program to solve this problem is shown in figure 1(a), the corresponding data deck is shown in figure 1(b), where each line represents one card, and the beginning of the results is presented in figure 1(c).

```
      DIMENSION F(3,3),NF(2),G(3,3),NG(2),H(2,3),NH(2),B(3,3),
     XNB(2),C(3,3),NC(2),XO(3),NXO(2),XT(3),NXT(2),V1(3),NV1(2),
     XV2(3),NV2(2),A1(3,3),NA1(2),U(3),NU(2),YT(3),NYT(2),W(18)
      DOUBLE PRECISION F,G,H,B,C,XO,U,XT,YT,TT,DELTAT,TFINAL,V1,V2,A1,W
      COMMON /MAX/ MAXRC
      COMMON  /LINES/NLP,LIN,TITLE(23)
      MAXRC=9
      CALL RDTITL

      READ (5,100) TT,DELTAT,TFINAL
  100 FORMAT (8F10.2)
      CALL READ (5,F,NF,G,NG,H,NH,U,NU,XO,NXO)
  101 FORMAT(1HO 59X, 'TIME RESPONSE',/' TIME',22X,'STATE',54X,
     1 'OUTPUT',/' TT',14X,'XT(1)',11X,'XT(2)',11X,'XT(3)',21X,'YT(1)',
     1 11X,'YT(2)',11X,'YT(3)')
  102 FORMAT(1HO F10.2,6X,3E16.7,10X,3E16.7)
      CALL LNCNT (4)
      WRITE(6,101)
   10 CALL EAT (F,NF,TT,B,NB,C,NC,W,18)
      CALL MULT(B,NB,XO,NXO,V1,NV1)
      CALL MULT(C,NC,G,NG,A1,NA1)
      CALL MULT(A1,NA1,U,NU,V2,NV2)
      CALL ADD(V1,NV1,V2,NV2,XT,NXT)
      CALL MULT(H,NH,XT,NXT,YT,NYT)
      CALL LNCNT (2)
      WRITE(6,102) TT,(XT(I),I=1,3),(YT(I),I=1,2)
      TT=TT+DELTAT
      IF(TT.GT.TFINAL)  STOP
      GO TO 10
      END
```

(a) User's main program.

Figure 1.– Example 1.

```
TEST PROGRAM 1   GENERATES TRANSIENT USING EAT
.01          .01        2.00
F             3   3
1.0          0.0        0.0
0.0          2.0        0.0
0.0          0.0        3.0
G             3   3
1.0          0.0        0.0
0.0          1.0        0.0
0.0          0.0        1.0
H             2   3
1.0          1.0        1.0
0.0          1.0        0.0
U             3   1
1.0
0.0
0.0
X0            3   1
1.0
2.0
3.0
```

(b) Data deck.

Figure 1.– Continued.

TEST PROGRAM 1 GENERATES TRANSIENT USING EAT VASP PROGRAM

F MATRIX 3 ROWS 3 COLUMNS

1.0000000D 00	0.0	0.0
0.0	2.0000000D 00	0.0
0.0	0.0	3.0000000D 00

G MATRIX 3 ROWS 3 COLUMNS

1.0000000D 00	0.0	0.0
0.0	1.0000000D 00	0.0
0.0	0.0	1.0000000D 00

H MATRIX 2 ROWS 3 COLUMNS

1.0000000D 00	1.0000000D 00	1.0000000D 00
0.0	1.0000000D 00	0.0

U MATRIX 3 ROWS 1 COLUMNS

1.0000000D 00
0.0
0.0

XO MATRIX 3 ROWS 1 COLUMNS

1.0000000D 00
2.0000000D 00
3.0000000D 00

TIME RESPONSE

TIME	STATE			OUTPUT		
TT	XT(1)	XT(2)	XT(3)	YT(1)	YT(2)	YT(3)
0.01	0.1020100D 01	0.2040403D 01	0.3091364D 01	0.6151867D 01	0.2040403D 01	
0.02	0.1040403D 01	0.2081622D 01	0.3185510D 01	0.6307534D 01	0.2081622D 01	
0.03	0.1060909D 01	0.2123673D 01	0.3282523D 01	0.6467105D 01	0.2123673D 01	
0.04	0.1081622D 01	0.2166574D 01	0.3382491D 01	0.6630686D 01	0.2166574D 01	
0.05	0.1102542D 01	0.2210342D 01	0.3485503D 01	0.6798387D 01	0.2210342D 01	
0.06	0.1123673D 01	0.2254994D 01	0.3591652D 01	0.6970319D 01	0.2254994D 01	
0.07	0.1145016D 01	0.2300548D 01	0.3701034D 01	0.7146598D 01	0.2300548D 01	

(c) Output

Figure 1.– Concluded.

The user's main program– This program will be discussed. statement by statement, using the line numbers on figure 1(a) as a reference.

Lines 1 and 2. These two statements allocate the necessary storage for the variables to be used and define them as double precision. Also, the dimension arrays NF, NG, etc.,are allocated storage. The dimensionality of F, G, etc., could have been included in the double precision statement instead of the dimension statement, and they could have been dimensioned as F(9) instead of F(3,3). The W array has been set up for dummy storage, and is dimensioned 18, as required by the EAT subroutine.

Lines 3 and 4. Common variables to be needed later are made available to the program. Although the variables listed in line 4 are not needed in this program, they are shown for reference.

Line 5. Since the basic matrices are (3,3), MAXRC is set to 9, to prevent overfilling the matrices. Note this will not protect from overfilling the arrays XO, XT, etc., since they are expected to be 3 X 1 vectors, and are dimensioned 3.

Line 6. This statement reads the first card of the data deck (see fig. 1(b)), places its contents in the TITLE array, and prints the first line of the output (see fig. 1(c)).

Lines 8 and 9. The initial time, the time increment, and final time are read from the second data card.

Line 10. The arrays F, G, H, U, and XO are read from the remainder of the data deck, and are printed (fig. 1(c)). Note that the dimensions used by the program are those given on the header card for each matrix. If these were specified as (2,2) this same main program would solve a second-order problem, rather than the third-order problem.

If the initial conditions were already stored in the XO array and you did not wish to disturb them, then the header card for the XO array would contain only the matrix title, no dimensions, and the associated data cards would be omitted. The matrix XO would still be printed.

Line 11. Line 11 contains the information to head the main output.

Line 12. Line 12 is the data format. For this program the transient output was printed using the programmers write statement rather than PRNT. The use of PRNT for this purpose is shown in the third example, p. 40.

Line 13. Line 13 tells the line counter that the program will print 4 lines.

Line 14. Line 14 does the actual printing.

Lines 15 through 25. Lines 15 through 25 form a loop which increments TT (line 23) and stops when TT is large enough (line 24).

Line 15. Line 15 computes the B and C matrices for time TT. When C is computed, the limits of the integral are 0 and the present TT. Note that W is specified for dummy storage and the "18" tells EAT the size of W.

Line 16. Line 16 computes BXO and stores the result in V1. Array V1 is set up for the programmers working storage. Since W is also available at this point in the program, it could have been used instead of V1 if desired.

Line 17. Line 17 computes CG and stores the result in A1, another working storage array.

Line 18. Line 18 computes (CG)U and stores the result in V2, still another working storage array. Note that MULT obtains the product CG from A1.

Line 19. Line 19 adds V1 and V2 to obtain XT. Since the ADD subroutine allows the matrices to be repeated in the call, the array V1 could have been eliminated, then line 16 would have stored its results in XT. Line 19, then, would have added XT and V2 to obtain the complete XT.

Line 20. Line 20 multiplies H times X to obtain Y.

Line 21. Line 21 tells the counter we are going to print 2 lines. If this will not fit on the present page, LNCNT will advance to the next page, print the title as on the first line of the first page of output, and increment the line counter to allow for the paging and the two lines about to be printed.

Line 22. Line 22 prints the variables XT and YT, skipping a line between each print line, as required by the 1HO in FORMAT 102. Note that YT(3) is not printed.

Example 2 – Transient Response Using TRNSI

This example uses the same equations as Example 1, except that u is piecewise constant, that is,

$$u(t) = JR - Kx(t_o + it_2) \qquad it_2 \leq t - t_o < (i+1)t_2$$

where i is a non-negative integer and J, R, K are constant matrices. The first term, JR, represents a forcing function and the second, Kx, is a feedback term. (See ASP manual, p. 121, for detailed explanation.)

It is desired to generate the transient response of such a system in response to a given initial condition x_o and a time varying control u. In particular, given

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \qquad G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$J = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad R = [0] \qquad K = \begin{bmatrix} 1 & 0 & 0.5 & 0 & 2 \\ 0 & 3 & 0 & 1 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \qquad x_o = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$t_1 = 0.5 \text{ sec}$$
$$t_2 = 2 \text{ sec}$$
$$t_f = 3.5 \text{ sec}$$
$$t_o = 0 \text{ sec}$$

The system is monitored at intervals t_1, while the control u(t) is changed only at sampling intervals t_2 (t_2 must be a positive integral multiple of t_1). Specifically, the control u(t) is updated by the equation:

$$u(t) = JR - Kx(t_o + it_2) \qquad it_2 \leqslant t - t_o < (i+1)t_2$$

The x,y vectors are computed at time intervals t_1, and these vectors together with the time t, and the control u (for the subsequent time interval) are printed out each time. The problem terminates when the final time t_f is reached. The matrix T has elements t_1, t_2, t_f, t_o in that order.

The user's main program to solve this problem is shown in figure 2(a), the corresponding data deck is shown in figure 2(b), and the results are presented in figure 2(c).

```
      DIMENSION F(5,5),NF(2),G(5,2),NG(2),J(2,1),NJ(2),R(1,1),NR(2),K(2,
     15),NK(2),H(7,5),NH(2),X(5),NX(2),T(4),NT(2),DUMMY(100)
      DOUBLE PRECISION F,G,J,R,K,H,X,T,DUMMY
      COMMON/FORM/NEPR,FMT1(6),FMT2(6)
      CALL RDTITL
      CALL READ(5,F,NF,G,NG,J,NJ,R,NR,K,NK)
      CALL READ(3,H,NH,X,NX,T,NT)
      CALL TRNSI(F,NF,G,NG,J,NJ,R,NR,K,NK,H,NH,X,NX,T,DUMMY,100)
      STOP
      END
```

(a) User's main program.

Figure 2.– Example 2.

```
TEST  PROGRAM     GENERATES TRANSIENT USING TRNSI
F          5    5
0.        0.         0.         0.         0.
0.        .5         0.         0.         0.
0.        0.         1.         1.         0.
0.        0.         0.         1.         0.
0.        0.         0.         0.         2.
G          5    2
1.        0.
1.        1.
0.        0.
1.        0.
0.        1.
J          2    1
0.
0.
R          1    1
0.
K          2    5
1.        0.         .5         0.         2.
0.        3.         0.         1.         0.
H          7    5
1.        0.         0.         0.         0.
0.        1.         0.         0.         0.
0.        0.         1.         0.         0.
0.        0.         0.         1.         0.
0.        0.         0.         0.         1.
1.        0.         0.         0.         0.
0.        1.         0.         1.         0.
X          5    1
1.
1.
1.
0.
1.
T          4    1
.5
2.
3.5
0.
```

(b) Data deck.

Figure 2.– Continued.

F MATRIX 5 ROWS 5 COLUMNS

0.0	0.0	0.0	0.0	0.0
0.0	5.0000000D-01	0.0	0.0	0.0
0.0	0.0	1.0000000D 00	1.0000000D 00	0.0
0.0	0.0	0.0	1.0000000D 00	0.0
0.0	0.0	0.0	0.0	2.0000000D 00

G MATRIX 5 ROWS 2 COLUMNS

1.0000000D 00	0.0
1.0000000D 00	1.0000000D 00
0.0	0.0
1.0000000D 00	0.0
0.0	1.0000000D 00

J MATRIX 2 ROWS 1 COLUMNS

0.0
0.0

R MATRIX 1 ROWS 1 COLUMNS

0.0

K MATRIX 2 ROWS 5 COLUMNS

1.0000000D 00	0.0	5.0000000D-01	0.0	2.0000000D 00
0.0	3.0000000D 00	0.0	1.0000000D 00	0.0

H MATRIX 7 ROWS 5 COLUMNS

1.0000000D 00	0.0	0.0	0.0	0.0
0.0	1.0000000D 00	0.0	0.0	0.0
0.0	0.0	1.0000000D 00	0.0	0.0
0.0	0.0	0.0	1.0000000D 00	0.0
0.0	0.0	0.0	0.0	1.0000000D 00
1.0000000D 00	0.0	0.0	0.0	0.0
0.0	1.0000000D 00	0.0	1.0000000D 00	0.0

(c) Output

Figure 2.– Continued.

```
TEST  PROGRAM      GENERATES TRANSIENT USING TRNSI                          VASP PROGRAM
   X       MATRIX          5 ROWS         1 COLUMNS
 1.0000000D 00
 1.0000000D 00
 1.0000000D 00
 0.0
 1.0000000D 00

   T       MATRIX          4 ROWS         1 COLUMNS
 5.0000000D-01
 2.0000000D 00
 3.5000000D 00
 0.0

      F    MATRIX          5 ROWS         5 COLUMNS
 0.0              0.0              0.0              0.0              0.0
 0.0              5.0000000D-01    0.0              0.0              0.0
 0.0              0.0              1.0000000D 00    1.0000000D 00    0.0
 0.0              0.0              0.0              1.0000000D 00    0.0
 0.0              0.0              0.0              0.0              2.0000000D 00

    EAT    MATRIX          5 ROWS         5 COLUMNS
 1.0000000D 00    0.0              0.0              0.0              0.0
 0.0              1.2840254D 00    0.0              0.0              0.0
 0.0              0.0              1.6487213D 00    8.2436069D-01    0.0
 0.0              0.0              0.0              1.6487213D 00    0.0
 0.0              0.0              0.0              0.0              2.7182819D 00

    INT    MATRIX          5 ROWS         5 COLUMNS
 5.0000002D-01    0.0              0.0              0.0              0.0
 0.0              5.6805086D-01    0.0              0.0              0.0
 0.0              0.0              6.4872131D-01    1.7563938D-01    0.0
 0.0              0.0              0.0              6.4872131D-01    0.0
 0.0              0.0              0.0              0.0              8.5914097D-01
```

(c) Output – Continued.

Figure 2.– Continued.

```
TEST  PROGRAM     GENERATES TRANSIENT USING TRNSI                              VASP PROGRAM

TRANSIENT RESPONSE, * INDICATES CONTROL CHANGES
    TIME     FIRST  5  ELEMENTS CONTAIN X, NEXT  7  ELEMENTS CONTAIN Y = HX, LAST  2  ELEMENTS CONTAIN U =JR -KX

*    0.0    1.0000000D 00  1.0000000D 00  1.0000000D 00  0.0             1.0000000D 00  1.0000000D 00  1.0000000D 00
            1.0000000D 00  0.0            1.0000000D 00  1.0000000D 00  1.0000000D 00 -3.5000000D 00 -3.0000000D 00

     0.50 -7.5000008D-01 -2.4083052D 00  1.0339835D 00 -2.2705246D 00  1.4085903D-01 -7.5000008D-01 -2.4083052D 00
            1.0339835D 00 -2.2705246D 00  1.4085903D-01 -7.5000008D-01 -4.6788297D 00 -3.5000000D 00 -3.0000000D 00

     1.00 -2.5000002D 00 -6.7846557D 00 -7.8171846D-01 -6.0139868D 00 -2.1945284D 00 -2.5000002D 00 -6.7846557D 00
          -7.8171846D-01 -6.0139868D 00 -2.1945284D 00 -2.5000002D 00 -1.2798642D 01 -3.5000000D 00 -3.0000000D 00

     1.50 -4.2500002D 00 -1.2404001D 01 -6.8612680D 00 -1.2185913D 01 -8.5427698D 00 -4.2500002D 00 -1.2404001D 01
          -6.8612680D 00 -1.2185913D 01 -8.5427698D 00 -4.2500002D 00 -2.4589914D 01 -3.5000000D 00 -3.0000000D 00

*    2.00 -6.0000003D 00 -1.9619383D 01 -2.1972644D 01 -2.2361699D 01 -2.5799080D 01 -6.0000003D 00 -1.9619383D 01
          -2.1972644D 01 -2.2361699D 01 -2.5799080D 01 -6.0000003D 00 -4.1981082D 01  6.8584482D 01  8.1219849D 01

     2.50  2.8292242D 01  5.9904692D 01 -4.2614736D 01  7.6240057D 00 -3.4987285D-01  2.8292242D 01  5.9904692D 01
          -4.2614736D 01  7.6240057D 00 -3.4987285D-01  2.8292242D 01  6.7528697D 01  6.8584482D 01  8.1219849D 01

     3.00  6.2584484D 01  1.6201563D 02 -5.1928756D 01  5.7062075D 01  6.8828247D 01  6.2584484D 01  1.6201563D 02
          -5.1928756D 01  5.7062075D 01  6.8828247D 01  6.2584484D 01  2.1907770D 02  6.8584482D 01  8.1219849D 01

     3.50  9.6876727D 01  2.9312866D 02 -2.6530179D 01  1.3857167D 02  2.5687388D 02  9.6876727D 01  2.9312866D 02
          -2.6530179D 01  1.3857167D 02  2.5687388D 02  9.6876727D 01  4.3170034D 02  6.8584482D 01  8.1219849D 01
```

(c) Output – Concluded.

Figure 2.– Concluded.

The user's main program— A brief explanation of the statements using line numbers on figure 2(a) as reference follows:

Lines 1, 2, and 3. Lines 1, 2, and 3 allocate storage, same as lines 1 and 2 of example 1.

Line 4. Common variables to be needed later are made available to the program.

Line 5. This statement reads the first card of the data deck (see fig. 2(b)), places its contents in TITLE array and prints the first line of the output (see fig. 2(c)).

Lines 6 and 7. The matrices F, G, J, R, K, H, X, and T are read in from data deck (see fig. 2(b)) and are printed.

Line 8. Line 8 calls the TRNSI subprogram, performs the computation, and prints outputs as explained in the example.

Example 3 – An Optimum Control Problem

Given a system

$$\dot{x} = Fx + Gu \qquad y = Hx \qquad x(o) = x_o$$

where x, u, and y are, respectively, the state, control, and observation vectors. The system, distribution, and observation matrices are F, G, and H, respectively.

We wish to define an optimal control u(t), where $u(t) = -Kx(t)$, so as to minimize the performance index

$$J = \int (x'H'QHx + U'RU)dt$$

The solution to this problem is

$$K = R^{-1}G'P$$

where P is the solution of the matrix Ricatti equation.

The VASP program finds P by means of the subroutines AUG, ETPHI, and RICAT, as follows.

First, subroutine AUG is used to generate the matrices

$$Z = \begin{bmatrix} -F & GR^{-1}G' \\ H'QH & F' \end{bmatrix} \qquad \text{and} \qquad C = R^{-1}G'$$

(Note: This is the negative of the Z given on page 212 of the ASP manual.) Subroutine ETPHI is then used to compute the special transition matrix

$$\theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} = e^{Z\tau}$$

Finally, the P matrix is computed by subroutine RICAT for a time increment of τ, by repeated application of the formula

$$P(t+\tau) = [\theta_{21} + \theta_{22}P(t)]\,[\theta_{11} + \theta_{12}P(t)]^{-1}$$

The computation is repeated for several steps, until $P(t+\tau) \approx P(t)$, which is the desired solution. Subroutine RICAT will also stop after a specified number of steps, if P has not converged to a solution. Finally, having P and K, we can compute the transient response of the system with optimum feedback from any desired initial condition. The differential equation becomes

$$\dot{x} = Fx - GKx = (F - GK)x = F^*x$$

and the solution is

$$x(t) = e^{F^*t}x$$

The time history of the control is

$$u(t) = -Kx(t)$$

An alternate solution, used in this example, is to first calculate the transition matrix

$$A2 = e^{F^*\tau_1}$$

where τ_1 is the time increment at which the solution is desired, then compute

$$x(t + \tau_1) = A2\ x(t), \qquad x(0) = x_o$$

The listing of a main program to solve this problem is given in figure 3(a), the data for a particular case is given in figure 3(b), and the first part of the results is given in figure 3(c). In this problem, H = I so the special case of AUG is used. As a result, H is not used in AUG, and need not have been used as an input.

```
0001          DIMENSION NF(2),NG(2),NFSTAR(2),NC(2),NCK(2),NQ(2),NR(2),NRI(2),
             1 NZ(2),NPHI(2),NPO(2),NPT(2),NXO(2),NXT(2),NA1(2),NA2(2),NH(2),
             2 NDELTC(2),NXX(2),NCONT(3),LAB(2),L(6)
0002          DOUBLE PRECISION Q(3,3),R(1,1),RI(1,1),F(3,3),G(3,1),FSTAR(3,3),
             1 C(1,3),CK(1,3),Z(6,6),PHI(6,6),PO(3,3),PT(3,3),XO(3),XT(3),XX(3),
             2 DELTC(1),H(3,3),A1(3,3),A2(3,3),DUMMY(72),TT,DELT1,DELT2,TFINAL,
             2 DET
0003          COMMON  /MAX/MAXRC
0004
0005          CALL RDTITL
0006          MAXRC=36
0007          KDUM=72
0008          READ(5,100) TT,DELT1,DELT2,TFINAL
0009          CALL READ  (5,F,NF,G,NG,XO,NXO,H,NH,Q,NQ)
0010          CALL READ  (2,R,NR,PO,NPO,R,NR,R,NR,R,NR)
0011      100 FORMAT(8F10.2)
0012      101 FORMAT(1H  59X, 'TIME RESPONSE'/5X,' TIME',22X,'STATE',43X,
             1 'OUTPUT'/5X' TT',14X,'XT(1)',11X,'XT(2)',11X,'XT(3)',21X,'DELTC')
0013      102 FORMAT(1H  F10.2,6X,3E16.7,10X,2E16.7,10X,E16.7)
0014          NCONT(1)=1
0015          NCONT(2)=IDINT(TFINAL/DELT1)+1
0016          NCONT(3)=4
0017          CALL EQUATE(XO,NXO,XT,NXT)
0018          CALL EQUATE(PO,NPO,PT,NPT)
0019          CALL EQUATE(R,NR,RI,NRI)
0020          CALL INV(RI,NRI    ,DET,L)
0021          CALL AUG(F,NF,G,NG,RI,NRI,H,NH,Q,NQ,C,NC,Z,NZ,1)
0022          CALL ETPHI(Z,NZ,DELT1,PHI,NPHI,DUMMY,KDUM)
0023          CALL RICAT(PHI,NPHI,C,NC,NCONT,CK,NCK,PT,NPT,DUMMY,KDUM)
0024          CALL MULT(G,NG,CK,NCK,A1,NA1)
0025          CALL SCALE(A1,NA1,A2,NA2,-1.D0)
0026          CALL ADD(F,NF,A2,NA2,FSTAR,NFSTAR)
0027          CALL PRNT  (FSTAR,NFSTAR,'FSTR',1)
0028          CALL ETPHI(FSTAR,NFSTAR,DELT2,A2,NA2,DUMMY,KDUM)
```

(a) User's main program.

Figure 3.– Example 3.

```
      CALL LNCNT(2)
      WRITE  (6,106)
  106 FORMAT  ('0 TRANSITION MATRIX')
      CALL PRNT  (A2,NA2,'  A2',1)
      CALL LNCNT(100)
      CALL LNCNT(3)
      WRITE(6,101)
      CALL MULT(CK,NCK,XT,NXT,DELTC,NDELTC)
      DELTC(1)=-1.0*DELTC(1)
      CALL LNCNT(1)
      WRITE(6,102) TT,XT,DELTC
  200 CALL EQUATE(XT,NXT,XX,NXX)
      CALL MULT(A2,NA2,XX,NXX,XT,NXT)
      CALL MULT(CK,NCK,XT,NXT,DELTC,NDELTC)
      DELTC(1)=-1.*DELTC(1)
      TT=TT+DELT2
      WRITE(6,102) TT,XT,DELTC
      IF(TT.GE.TFINAL)  STOP
      GO TO 200
      END
```

(a) User's main program – Concluded.

Figure 3.– Continued.

```
TEST PROGRAM 2A   GENERATES TRANSIENT USING  X(I+1)=EXP(F*T)*X(I)
0.0          1.0          0.01         3.5
F             3   3
-0.2767      1.0          -0.0372
-17.0872     -0.1785      -12.1983
0.0          0.0          -6.67
G             3   1
0.0
0.0
6.67
X0            3   1
1.0
0.0
0.0
H             3   3
1.0          0.0          0.0
0.0          1.0          0.0
0.0          0.0          1.0
Q             3   3
0.2          0.0          0.0
0.0          0.2          0.0
0.0          0.0          0.0
R             1   1
1.0
P0            3   3
0.0          0.0          0.0
0.0          0.0          0.0
0.0          0.0          0.0
```

(b) Data deck.

Figure 3.– Continued.

```
TEST PROGRAM 2A.  GENERATES TRANSIENT USING  X(I+1)=EXP(F*T)*X(I)          VASP PROGRAM

    F      MATRIX        3 ROWS        3 COLUMNS
 -2.7670000D-01   1.0000000D 00   -3.7200000D-02
 -1.7087200D 01  -1.7850000D-01   -1.2198300D 01
  0.0             0.0             -6.6700000D 00

    G      MATRIX        3 ROWS        1 COLUMNS
  0.0
  0.0
  6.6700000D 00

    XO     MATRIX        3 ROWS        1 COLUMNS
  1.0000000D 00
  0.0
  0.0

    H      MATRIX        3 ROWS        3 COLUMNS
  1.0000000D 00   0.0              0.0
  0.0             1.0000000D 00    0.0
  0.0             0.0              1.0000000D 00

    Q      MATRIX        3 ROWS        3 COLUMNS
  2.0000000D-01   0.0              0.0
  0.0             2.0000000D-01    0.0
  0.0             0.0              0.0

    R      MATRIX        1 ROWS        1 COLUMNS
  1.0000000D 00

    PO     MATRIX        3 ROWS        3 COLUMNS
  0.0             0.0              0.0
  0.0             0.0              0.0
  0.0             0.0              0.0
            1    ITERATIONS
```

(c) Output.

Figure 3.– Continued.

```
TEST PROGRAM 2A   GENERATES TRANSIENT USING  X(I+1)=EXP(F*T)*X(I)          VASP PROGRAM

     K(T)  MATRIX         1 ROWS        3 COLUMNS
  7.3603732D-01  -3.6359943D-01   5.1645438D-01

     P(T)  MATRIX         3 ROWS        3 COLUMNS
  6.6799427D-01  -2.1615347D-02   1.1035042D-01
 -2.1615347D-02   5.5929130D-02  -5.4512659D-02
  1.1035042D-01  -5.4512659D-02   7.7429442D-02
              2    ITERATIONS

     K(T)  MATRIX         1 ROWS        3 COLUMNS
  7.5463314D-01  -3.6917988D-01   5.3032918D-01

     P(T)  MATRIX         3 ROWS        3 COLUMNS
  6.7636913D-01  -2.1772154D-02   1.1313840D-01
 -2.1772154D-02   5.6466312D-02  -5.5349307D-02
  1.1313840D-01  -5.5349307D-02   7.9509622D-02
              3    ITERATIONS

     K(T)  MATRIX         1 ROWS        3 COLUMNS
  7.5462283D-01  -3.6920394D-01   5.3036120D-01

     P(T)  MATRIX         3 ROWS        3 COLUMNS
  6.7640398D-01  -2.1764191D-02   1.1313686D-01
 -2.1764191D-02   5.6470368D-02  -5.5352914D-02
  1.1313686D-01  -5.5352914D-02   7.9514423D-02
              4    ITERATIONS

     K(T)  MATRIX         1 ROWS        3 COLUMNS
  7.5462275D-01  -3.6920418D-01   5.3036153D-01
```

(c) Output – Continued.

Figure 3.– Continued.

P(T) MATRIX	3 ROWS	3 COLUMNS
6.7640404D-01	-2.1764171D-02	1.1313684D-01
-2.1764171D-02	5.6470397D-02	-5.5352950D-02
1.1313684D-01	-5.5352950D-02	7.9514472D-02

FSTR MATRIX	3 ROWS	3 COLUMNS
-2.7670000D-01	1.0000000D 00	-3.7200000D-02
-1.7087200D 01	-1.7850000D-01	-1.2198300D 01
-5.0333338D 00	2.4625919D 00	-1.0207511D 01

TRANSITION MATRIX

A2 MATRIX	3 ROWS	3 COLUMNS
9.9640412D-01	9.9651891D-03	-9.4141918D-04
-1.6739033D-01	9.9592474D-01	-1.1573683D-01
-4.9775055D-02	2.3128283D-02	9.0157826D-01

(c) Output – Continued.

Figure 3.– Continued.

TEST PROGRAM 2A GENERATES TRANSIENT USING X(I+1)=EXP(F*T)*X(I) VASP PROGRAM

TIME RESPONSE

TIME TT	STATE XT(1)	XT(2)	XT(3)	OUTPUT DELTC
0.0	0.1000000D 01	0.0	0.0	-0.7546228D 00
0.01	0.9964041D 00	-0.1673903D 00	-0.4977506D-01	-0.7873117D 00
0.02	0.9911999D 00	-0.3277358D 00	-0.9834363D-01	-0.8168258D 00
0.03	0.9844623D 00	-0.4809355D 00	-0.1455815D 00	-0.8432503D 00
0.04	0.9762668D 00	-0.6269159D 00	-0.1913780D 00	-0.8666736D 00
0.05	0.9666891D 00	-0.7656292D 00	-0.2356354D 00	-0.8871871D 00
0.06	0.9558051D 00	-0.8970518D 00	-0.2782685D 00	-0.9048847D 00
0.07	0.9436909D 00	-0.1021183D 01	-0.3192033D 00	-0.9198623D 00
0.08	0.9304217D 00	-0.1138042D 01	-0.3583772D 00	-0.9322178D 00
0.09	0.9160726D 00	-0.1247671D 01	-0.3957379D 00	-0.9420502D 00
0.10	0.9007178D 00	-0.1350126D 01	-0.4312427D 00	-0.9494598D 00
0.11	0.8844306D 00	-0.1445485D 01	-0.4648584D 00	-0.9545475D 00
0.12	0.8672834D 00	-0.1533838D 01	-0.4965604D 00	-0.9574147D 00
0.13	0.8493472D 00	-0.1615292D 01	-0.5263322D 00	-0.9581629D 00
0.14	0.8306919D 00	-0.1689966D 01	-0.5541649D 00	-0.9568936D 00
0.15	0.8113857D 00	-0.1757991D 01	-0.5800568D 00	-0.9537079D 00
0.16	0.7914954D 00	-0.1819511D 01	-0.6040126D 00	-0.9487064D 00
0.17	0.7710861D 00	-0.1874678D 01	-0.6260435D 00	-0.9419887D 00
0.18	0.7502212D 00	-0.1923654D 01	-0.6461662D 00	-0.9336536D 00
0.19	0.7289623D 00	-0.1966610D 01	-0.6644025D 00	-0.9237984D 00
0.20	0.7073689D 00	-0.2003720D 01	-0.6807793D 00	-0.9125195D 00
0.21	0.6854987D 00	-0.2035170D 01	-0.6953277D 00	-0.8999112D 00
0.22	0.6634074D 00	-0.2061147D 01	-0.7080831D 00	-0.8860665D 00
0.23	0.6411488D 00	-0.2081844D 01	-0.7190843D 00	-0.8710764D 00
0.24	0.6187743D 00	-0.2097458D 01	-0.7283734D 00	-0.8550301D 00
0.25	0.5963334D 00	-0.2108187D 01	-0.7359958D 00	-0.8380144D 00
0.26	0.5738734D 00	-0.2114234D 01	-0.7419991D 00	-0.8201144D 00
0.27	0.5514396D 00	-0.2115803D 01	-0.7464334D 00	-0.8014125D 00
0.28	0.5290751D 00	-0.2113096D 01	-0.7493510D 00	-0.7819890D 00
0.29	0.5068206D 00	-0.2106319D 01	-0.7508056D 00	-0.7619218D 00
0.30	0.4847151D 00	-0.2095676D 01	-0.7508525D 00	-0.7412862D 00
0.31	0.4627952D 00	-0.2081371D 01	-0.7495484D 00	-0.7201551D 00
0.32	0.4410954D 00	-0.2063606D 01	-0.7469508D 00	-0.6985987D 00
0.33	0.4196482D 00	-0.2042582D 01	-0.7431178D 00	-0.6766848D 00
0.34	0.3984841D 00	-0.2018497D 01	-0.7381083D 00	-0.6544784D 00
0.35	0.3776314D 00	-0.1991547D 01	-0.7319813D 00	-0.6320420D 00
0.36	0.3571164D 00	-0.1961926D 01	-0.7247961D 00	-0.6094353D 00
0.37	0.3369636D 00	-0.1929822D 01	-0.7166119D 00	-0.5867156D 00
0.38	0.3171955D 00	-0.1895424D 01	-0.7074875D 00	-0.5639372D 00
0.39	0.2978327D 00	-0.1858913D 01	-0.6974817D 00	-0.5411522D 00

(c) Output – Concluded.

Figure 3.– Concluded.

The user's main program– Some of the details of the main program are discussed briefly. The various matrices are first dimensioned and stated to be double precision. The problem will be solved using basically 3 X 3 matrices, but Z and PHI are 6 X 6 matrices so MAXRC is set to 36 (line 6). A double size dummy array is required in ETPHI, so DUMMY is dimensioned at 72, and KDUM is set to 72 (line 7).

In line 8 the timing information is read in. TT is the initial time, DELT1 is the time increment used in the computation of P, DELT2 is the time increment, τ_1, desired in the printout of the transient and TFINAL is the final time for the transient.

Lines 9 and 10 read data cards to fill a total of 7 matrices.

Lines 14, 15, and 16 set up the appropriate constants for RICAT, specifying a print every step (line 14), the maximum number of steps to be taken by RICAT (line 15), and that both P and K should be printed (line 16). Lines 17 and 18 store the initial values of x_0 and P_0 in the running matrices, and lines 19 through 23 do the necessary computations to obtain P and K (called CK in program). Then F* and the transition matrix A2 (lines 28 through 32) are computed and printed. The transition matrix is labeled on the output (lines 29 through 31). Lines 33 through 39 page the output, print a heading for the transient response, and print the first point. Lines 40 through 47 then increment the solution and the time, and print x(t) and u(t) (called XT and DELTC in the program).

Example 4 – Sampled Data Ricatti Solution

This example is provided to show the general use of the subroutine SAMPL. The theory of the example is given in the ASP manual, page 222, and very briefly in the dictionary description of SAMPL, page 24, in this manual. A listing of the main program is shown in figure 4(a). The data deck is shown in figure 4(b), and the output in figure 4(c).

The main program is reasonably self-explanatory. The statement NCONT(2) = 4(line 13) indicates that SAMPL is to compute P for four successive time intervals and then stop. Both P and K (line 14) are to be printed at every step (line 12).

As mentioned in the dictionary, K is the weighting matrix corresponding to the beginning of the interval, and P is the covariance matrix corresponding to the end of the interval. This is apparent in the output. For example, the first entry to SAMPL prints step number 0 and the K matrix, followed by step number 1 and the P matrix. On exit from SAMPL, P and K contain the data corresponding to P_i and K_{i-1}, which is the last interval. If printing is requested, the exit value of P and K will always be printed, and will be the last set of data.

```
      C     CHECK PROCEDURE FOR SAMPLE  SEE PAGES 234 AND 244 OF ASP MANUAL
            DIMENSION  NPHI(2),NQ(2),NR(2),NP(2),NK(2),NCONT(3),NH(2)
            DOUBLE PRECISION  PHI(3,3),H(3,2),Q(3,3),R(2,2),P(3,3),K(3,2),
           1                  DUM(54)
            COMMON  /MAX/MAXRC
            MAXRC=9
            NDUM=54

            CALL RDTITL
         10 CALL READ   (4,PHI,NPHI,H,NH,Q,NQ,R,NR,R,NR)
            NP(1)=NPHI(1)
            NP(2)=NPHI(2)
            CALL UNITY  (P,NP)
            NCONT(1)=1
            NCONT(2)=4
            NCONT(3)=4
            CALL SAMPL  (PHI,NPHI,H,NH,Q,NQ,R,NR,P,NP,K,NK,NCONT,DUM,NDUM)
            CALL EXIT
            GO TO 10
            END
```

(a) Main program.

```
TEST PROGRAM FOR SAMPL   CASE 1   FROM ASP MANUAL P234 AND P244
PHI           3    3
0            1.0
0            0           0
0            0           2.0
H             2    3
0.0          2.0         0.0
0.0          0.0         1.0
Q             3    3
3.0          1.0         0.0
1.0          1.0         0.0
0            0           1.0
R             2    2
1.0          1.0
1.0          2.0
```

(b) Data.

Figure 4.– Example 4.

```
TEST PROGRAM FOR SAMPL   CASE 1   FROM ASP MANUAL P234 AND P244            VASP PROGRAM


     PHI    MATRIX          3 ROWS        3 COLUMNS
  0.0              1.0000000D 00   0.0
  0.0              0.0             0.0
  0.0              0.0             2.0000000D 00

     H      MATRIX          2 ROWS        3 COLUMNS
  0.0              2.0000000D 00   0.0
  0.0              0.0             1.0000000D 00

     Q      MATRIX          3 ROWS        3 COLUMNS
  3.0000000D 00    1.0000000D 00   0.0
  1.0000000D 00    1.0000000D 00   0.0
  0.0              0.0             1.0000000D 00

     R      MATRIX          2 ROWS        2 COLUMNS
  1.0000000D 00    1.0000000D 00
  1.0000000D 00    2.0000000D 00

STEP NUMBER=   0 IN SAMPL

     K(T)   MATRIX          3 ROWS        2 COLUMNS
  4.2857143D-01   -1.4285714D-01
  0.0              0.0
 -1.4285714D-01    7.1428571D-01

STEP NUMBER=   1 IN SAMPL

     P(T)   MATRIX          3 ROWS        3 COLUMNS
  3.1428571D 00    1.0000000D 00   2.8571429D-01
  1.0000000D 00    1.0000000D 00   0.0
  2.8571429D-01    0.0             3.5714286D 00
```

(c) Output.

Figure 4.– Continued.

```
STEP NUMBER=    1 IN SAMPL
       K(T)  MATRIX          3 ROWS          2 COLUMNS
  4.1489362D-01  -7.4468085D-02
  0.0             0.0
 -2.6595745D-01   1.3297872D 00

STEP NUMBER=    2 IN SAMPL
       P(T)  MATRIX          3 ROWS          3 COLUMNS

  3.1702128D 00   1.0000000D 00   5.3191489D-01
  1.0000000D 00   1.0000000D 00   0.0
  5.3191489D-01   0.0             5.7872340D 00

STEP NUMBER=    2 IN SAMPL

       K(T)  MATRIX          3 ROWS          2 COLUMNS
  4.1054403D-01  -5.2720135D-02
  0.0             0.0
 -3.0510376D-01   1.5255188D 00

STEP NUMBER=    3 IN SAMPL

       P(T)  MATRIX          3 ROWS          3 COLUMNS
  3.1789119D 00   1.0000000D 00   6.1020752D-01
  1.0000000D 00   1.0000000D 00   0.0
  6.1020752D-01   0.0             6.4918676D 00

STEP NUMBER=    3 IN SAMPL

       K(T)  MATRIX          3 ROWS          2 COLUMNS
  4.0964801D-01  -4.8240037D-02
  0.0             0.0
 -3.1316793D-01   1.5658397D 00

STEP NUMBER=    4 IN SAMPL

       P(T)  MATRIX          3 ROWS          3 COLUMNS
  3.1807040D 00   1.0000000D 00   6.2633587D-01
  1.0000000D 00   1.0000000D 00   0.0
  6.2633587D-01   0.0             6.6370228D 00
```

(c) Output – Concluded.

Figure 4.– Concluded.

Example 5 – Matrix Decomposition

This example is a test program to check the operation of DECGEN. It first generates a matrix R to be decomposed, then proceeds with the decomposition, and checks the result, printing all of the associated matrices. The general procedure is to input a diagonal matrix ZL and transform it into the matrix R to be decomposed. Figure 5(a) is a listing of the main program; figure 5(b) is a listing of the subroutine ORTH; figure 5(c) is the data deck; and figures 5(d) through 5(f) are the output.

In the main program, all matrices are dimensioned 100, although the actual matrix size used is 2 X 2 and 4 X 4. Accordingly, MAXRC is set to 100. The dummy matrix is dimensioned 700, since DECGEN requires that much. The input matrices are read at line 8.

Subroutine ORTH, called at line 9, produces a n X n orthogonal matrix, using the original T matrix, and places the results back in T. The procedure is as follows.

First, generate an elementary rotation matrix E_{ij}. This is a unity matrix, with elements e_{ii} and e_{jj} replaced by $\cos t_{ij}$ and elements $e_{ij} = -e_{ji} = \sin t_{ij}$.

Then,

$$T = \Pi E_{ij}$$

Lines 10 through 17 set up indices for referring to the seven dummy matrices. The input matrix, ZL, is then transformed by the matrix T, so that

$$ZL_1 = T{*}ZL{*}T'$$

Note that ORTH leaves T′ in DUM3. Also, if the T at input was the null matrix, the rotation will be the identity matrix, so that R = ZL. Lines 19 through 27 then juxtapose either the matrix EXR or the matrix EXC, using JUXTR or JUXTC, depending on the compatibility of the dimensions. If both sets of dimensions are incompatible, no juxtaposition is done. In any case, the result of this operation is placed in R. The decomposition routine is called next. *If the original ZL matrix had zero in element (2,1) and no juxtaposition was done, then R is assumed symmetric, and the DECSYM entry is used.* If ZL was not symmetric, the program will produce errors. Otherwise, the DECGEN entry is used (lines 29 to 31). Finally, the resulting matrices H and G are tested using

$$R_1 = HG$$

$$RE = R - R_1$$

and all resulting matrices are printed.

In figure 5(c), blank lines represent blank cards. In the data cards for case 4 the header card for EXR has no dimension information and no associated data cards. This indicates that the matrix EXR is to be left unchanged, and that no data cards are to be read for EXR. In case 7, EXR is again left unchanged. A blank data card follows the EXC header card.

The output (figs. 5(d) through 5(f)) contains the results of decomposing three different matrices. Figure 5(d), case 1, is a 2 X 2 rank 1 matrix; figure 5(e), case 4, is a 2 X 3 rank 2 matrix; and

```
C       MAIN PROGRAM TO CHECK DECOM ET AL
        DIMENSION   NZL(2),NT(2),NEXR(2),NR(2),NG(2),NH(2),ND(2),NR1(2),
       1   NRE(2),NEXC(2)
        DOUBLE PRECISION ZL(100),T(100),EXR(100),R(100),G(100),H(100),
       1   DUM(700),R1(100),RE(100),EXC(100)
        COMMON  /MAX/MAXRC
        MAXRC=100

        KDUM=700
     20 CALL RDTITL
        CALL READ  (4,ZL,NZL,T,NT,EXR,NEXR,EXC,NEXC,T,NT)
        CALL ORTH  (T,NT,DUM,KDUM)
        M=NT(1)
        M2=M*M+1
        MS=M*M
        M3=M2+MS
        M4=M3+MS
        M5=M4+MS
        M6=M5+MS
        M7=M6+MS
        DUM(M7) =ZL(2)
        CALL MULT  (T,NT,ZL,NZL,DUM(M3),ND)
        CALL MULT  (DUM(M3),ND,DUM(M2),ND,ZL,NZL)
        IF (NZL(1).NE.NEXC(1))GO TO 30
        CALL JUXTC (ZL,NZL,EXC,NEXC,R,NR)
        GO TO 50
     30 IF (NZL(2) .NE.NEXR(2))GO TO 40
        CALL JUXTR (ZL,NZL,EXR,NEXR,R,NR)
        GO TO 50
     40 CALL EQUATE(ZL,NZL,R,NR)
        IF  (DUM(M7).NE.0.D0)   GO TO 50
        CALL DECSYM (R,NR,G,NG,H,NH,DUM,KDUM)
        GO TO 70
```

(a) Main program.

Figure 5.– Example 5.

```
   50 CALL DECGEN(R,NR,G,NG,H,NH,DUM,KDUM)
   70 CALL MULT  (H,NH,G,NG,R1,NR1)
      CALL SUBT  (R,NR,R1,NR1,RE,NRE)
      CALL PRNT (R,NR,'R   ',1)
      CALL PRNT (R1,NR1,'R1  ',1)
      CALL PRNT (RE,NRE,'RERR',1)
      CALL PRNT (H ,NH ,'H   ',1)
      CALL PRNT (G ,NG ,'G   ',1)
      ND(1)=1
      ND(2)=1
      CALL PRNT (DUM(M6),ND,'RANK',1)
      GO TO 20
      END
```

(a) Main program – Concluded.

Figure 2.– Continued.

```
      SUBROUTINE  ORTH  (T,NT,DUM,KDUM)
      DOUBLE PRECISION T(1),DUM(1),CTH,STH
      DIMENSION NT(2)
      LDM2=2*NT(1)**2+1
   10 LDM1=NT(1)**2+1
      N=NT(1)
      CALL UNITY (DUM(LDM1),NT)
      NM=NT(1)-1
      DO 20 J=1,NM
      JP=J+1
      DO 20 I=JP,N
      CALL UNITY (DUM,NT)
      II=N*(I-1)+I
      JJ=N*(J-1)+J
      IJ=N*(I-1)+J
      JI=N*(J-1)+I
      CTH=DCOS(T(IJ))
      STH=DSIN(T(IJ))
      DUM(II)=CTH
      DUM(JJ)=CTH
      DUM(IJ)=STH
      DUM(JI)=-STH
      CALL MULT (DUM(LDM1),NT,DUM,NT,DUM(LDM2),NT)
      CALL EQUATE (DUM(LDM2),NT,DUM(LDM1),NT)
   20 CONTINUE
      CALL TRANP (DUM(LDM1),NT,T,NT)
      CALL MULT (T,NT,DUM(LDM1),NT,DUM(LDM2),NT)
      CALL PRNT (T,NT,4HT   ,1)
      CALL PRNT (DUM(LDM2),NT,4HT*T',1)
      RETURN
      END
```

(b) Subroutine ORTH.

Figure 5.– Continued.

```
   TEST PROGRAM FOR DECGEN AND DECOM   CASE 1         2X2 RANK1
 ZL            2    2
 1.0          1.0
 2.0          2.0
 T             2    2


 EXR           1    1

 EXC           1    1

   TEST PROGRAM FOR DECGEN AND DECOM   CASE 4      2X3 RANK2
 ZL            2    2
 1.
              2.0
 T             2    2
              .7


 EXR
 EXC           2    1
 2.
 3.
   TEST PROGRAM FOR DECGEN AND DECOM   CASE 7  ILL-COND 4X4 RANK3
 ZL            4    4
 1.
              2.


                                         1.D-6
 T             4    4
              .2        .3        .4
                        .5        .6
                                  .7


 EXR
 EXC           1    1
```

(c) Data.

Figure 5.– Continued.

```
TEST PROGRAM FOR DECGEN AND DECOM   CASE 1        2X2 RANK1                 VASP PROGRAM

   ZL     MATRIX        2 ROWS        2 COLUMNS
1.0000000D 00    1.0000000D 00
2.0000000D 00    2.0000000D 00

   T      MATRIX        2 ROWS        2 COLUMNS
0.0              0.0
0.0              0.0

   EXR    MATRIX        1 ROWS        1 COLUMNS
0.0

   EXC    MATRIX        1 ROWS        1 COLUMNS
0.0

   T      MATRIX        2 ROWS        2 COLUMNS
1.0000000D 00    0.0
0.0              1.0000000D 00

   T*T'   MATRIX        2 ROWS        2 COLUMNS
1.0000000D 00    0.0
0.0              1.0000000D 00

   R      MATRIX        2 ROWS        2 COLUMNS
1.0000000D 00    1.0000000D 00
2.0000000D 00    2.0000000D 00

   R1     MATRIX        2 ROWS        2 COLUMNS
1.0000000D 00    1.0000000D 00
2.0000000D 00    2.0000000D 00

   RERR   MATRIX        2 ROWS        2 COLUMNS
4.1633363D-16    4.1633363D-16
4.4408921D-16    4.4408921D-16
```

(d) Case 1.

Figure 5.– Continued.

```
H       MATRIX          2 ROWS          1 COLUMNS
 1.4142136D 00
 2.8284271D 00
------------------------------------------------------------
G       MATRIX          1 ROWS          2 COLUMNS
 7.0710678D-01   7.0710678D-01
------------------------------------------------------------
RANK    MATRIX          1 ROWS          1 COLUMNS
------------------------------------------------------------
 1.0000000D 00
```

(d) Case 1 – Concluded.

Figure 5. – Continued.

```
TEST PROGRAM FOR DECGEN AND DECOM   CASE 4      2X  RANK2                    VASP PROGRAM

    ZL     MATRIX        2 ROWS       2 COLUMNS
 1.0000000D 00     0.0
 0.0              2.0000000D 00

    T      MATRIX        2 ROWS       2 COLUMNS
 7.9956985D-01    6.0057311D-01
-6.0057311D-01    7.9956985D-01

    EXR    MATRIX        1 ROWS       1 COLUMNS
 0.0

    EXC    MATRIX        2 ROWS       1 COLUMNS
 2.0000000D 00
 3.0000000D 00

    T      MATRIX        2 ROWS       2 COLUMNS
 8.2501188D-01   -5.6511539D-01
 5.6511539D-01    8.2501188D-01

    T*T'   MATRIX        2 ROWS       2 COLUMNS
 1.0000000D 00    0.0
 0.0              1.0000000D 00

    R      MATRIX        2 ROWS       3 COLUMNS
 1.3193554D 00   -4.6622691D-01   2.0000000D 00
-4.6622691D-01    1.6806446D 00   3.0000000D 00

    R1     MATRIX        2 ROWS       3 COLUMNS
 1.3193554D 00   -4.6622691D-01   2.0000000D 00
-4.6622691D-01    1.6806446D 00   3.0000000D 00
```

(e) Case 4.

Figure 5.– Continued.

```
   RERR   MATRIX          2 ROWS          3 COLUMNS
 2.2204460D-16  -2.4980018D-16   4.4408921D-16
-9.7144515D-17   2.2204460D-16   4.4408921D-16

   H      MATRIX          2 ROWS          2 COLUMNS
 2.0493573D 00   1.3259717D 00
 0.0             3.4701490D 00

   G      MATRIX          2 ROWS          3 COLUMNS
 7.3071906D-01  -5.4085965D-01   4.1655791D-01
-1.3435357D-01   4.8431483D-01   8.6451620D-01

   RANK   MATRIX          1 ROWS          1 COLUMNS
 2.0000000D 00
```

(e) Case 4 – Concluded.

Figure 5.– Continued.

TEST PROGRAM FOR DECGEN AND DECOM CASE 7 ILL-COND 4X4 RANK3 VASP PROGRAM

ZL MATRIX 4 ROWS 4 COLUMNS

1.0000000D 00	0.0	0.0	0.0
0.0	2.0000000D 00	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0000000D-06

T MATRIX 4 ROWS 4 COLUMNS

8.7748949D-01	1.5497917D-01	2.4871521D-01	3.7965037D-01
-3.9329146D-01	7.6892918D-01	2.0667223D-01	4.5973507D-01
-2.6426304D-01	-5.4846577D-01	6.9767211D-01	3.7762941D-01
-7.4155739D-02	-2.8968030D-01	-6.3928160D-01	7.0845275D-01

EXR MATRIX 1 ROWS 1 COLUMNS

0.0

EXC MATRIX 1 ROWS 1 COLUMNS

0.0

T MATRIX 4 ROWS 4 COLUMNS

8.8942548D-01	-1.3895672D-01	2.2863098D-01	-3.7059576D-01
-6.6804271D-02	8.9817561D-01	-1.3776143D-01	-4.1211595D-01
1.1688667D-01	1.2733255D-02	9.4444935D-01	-3.0690520D-01
4.3680316D-01	4.1690464D-01	1.9174226D-01	7.7371082D-01

T*T' MATRIX 4 ROWS 4 COLUMNS

1.0000000D 00	2.7755576D-17	-1.3877788D-17	-5.5511151D-17
2.7755576D-17	1.0000000D 00	4.1633363D-17	-4.1633363D-17
-1.3877788D-17	4.1633363D-17	1.0000000D 00	0.0
-5.5511151D-17	-4.1633363D-17	0.0	1.0000000D 00

(f) Case 7.

Figure 5.– Continued.

R MATRIX 4 ROWS 4 COLUMNS

8.2969577D-01	-3.0903234D-01	1.0042336D-01	2.7264018D-01
-3.0903234D-01	1.6179018D 00	1.5064996D-02	7.1972651D-01
1.0042336D-01	1.5064996D-02	1.3986860D-02	6.1673338D-02
2.7264018D-01	7.1972651D-01	6.1673338D-02	5.3841655D-01

R1 MATRIX 4 ROWS 4 COLUMNS

8.2969577D-01	-3.0903234D-01	1.0042336D-01	2.7264018D-01
-3.0903234D-01	1.6179018D 00	1.5064996D-02	7.1972651D-01
1.0042336D-01	1.5064996D-02	1.3986801D-02	6.1673628D-02
2.7264018D-01	7.1972651D-01	6.1673628D-02	5.3841512D-01

RERR MATRIX 4 ROWS 4 COLUMNS

6.9388939D-17	-8.3266727D-17	4.1633363D-17	1.3877788D-17
-5.5511151D-17	2.2204460D-16	-1.7347235D-18	8.3266727D-17
4.1633363D-17	0.0	5.8972036D-08	-2.9047379D-07
1.3877788D-17	1.5265567D-16	-2.9047379D-07	1.4307633D-06

H MATRIX 4 ROWS 2 COLUMNS

8.7787704D-01	-2.4295612D-01
0.0	1.2719677D 00
1.1767126D-01	1.1843851D-02
4.6716539D-01	5.6583710D-01

G MATRIX 2 ROWS 4 COLUMNS

8.7787704D-01	0.0	1.1767126D-01	4.6716539D-01
-2.4295612D-01	1.2719677D 00	1.1843851D-02	5.6583710D-01

RANK MATRIX 1 ROWS 1 COLUMNS

2.0000000D 00

(f) Case 7 – Concluded.

Figure 5. – Concluded.

finally, figure 5(f), case 7, is a 4 X 4 matrix of rank 3, with one very small eigenvalue equal to 10^{-6}. The error matrices of the first two decompositions are extremely small, but that from the third one has errors of the order of 10^{-6}. These are caused by the built-in pivot rejection device, which rejects all pivots smaller than 2×10^{-5} times the largest of the diagonal elements (see DECOM, p. 85 and PSEU, p. 70). This last matrix, case 7, was also tried with an eigenvalue of 10^{-3}, and the errors were then on the order of 10^{-16}.

Example 6 – Use of the Pseudoinverse Routine

This program is designed to check the operation of PSEUDO. The procedure is as follows:

First the input matrix A is read; then $B = A^{\#}$ is computed. The accuracy of the pseudoinverse is then checked by the first two Moore-Penrose axioms

$$BAB - B = A_{\epsilon}$$

$$ABA - A = B_{\epsilon}$$

All the various matrices are printed.

Figure 6(a) is the program listing and figure 6(b), the output. Three cases are presented; the first two are the examples presented in the ASP manual; and the third one contains several zeros. The first matrix printed for each case is the input matrix and each has a different label. The other titles are abbreviations chosen to fit the allotted four character space as follows:

APSE → $A^{\#}$
AASA → $AA^{\#}A$
AERR → A_{ϵ} or B_{ϵ}
ASAA → $A^{\#}AA^{\#}$

It can be noted that the size of the numbers in the AERR matrices is 10^{-16}, which is very good.

Ames Research Center
National Aeronautics and Space Administration
Moffett Field, Calif., 94035, July 12, 1971

```
      DIMENSION A(50),B(50),W(350),NA(2),NB(2),A1(50),A2(50),NA1(2),
     1 NA2(2),LAB(2)
      DOUBLE PRECISION A,B,W,A1,A2
      COMMON  /MAX/MAXRC

      MAXRC=50
    5 CALL RDTITL
      CALL READ   (1,A,NA,A,NA,A,NA,A,NA,A,NA)
      NW=350
      CALL PSEUDO(A,NA,B,NB,W,NW)
      CALL PRNT  (B,NB,'APSE',1)
      CALL MULT(A,NA,B,NB,A1,NA1)
      CALL MULT(A1,NA1,A,NA,A2,NA2)
      CALL SCALE(A,NA,A1,NA1,-1.D0)
      CALL ADD(A1,NA1,A2,NA2,A1,NA1)
      CALL PRNT  (A2,NA2,'AASA',1)
      CALL PRNT  (A1,NA1,'AERR',1)
      CALL MULT(B,NB,A,NA,A1,NA1)
      CALL MULT(A1,NA1,B,NB,A2,NA2)
      CALL SCALE(B,NB,A1,NA1,-1.D0)
      CALL ADD(A1,NA1,A2,NA2,A1,NA1)
      CALL PRNT  (A2,NA2,'ASAA',1)
      CALL PRNT  (A1,NA1,'AERR',1)
      GO TO 5
      END
```

(a) Main program to check PSEUDO.

Figure 6.– Example 6.

PSEUDO TEST PROGRAM CASE 1 FROM ASP MANUAL PAGE 146 VASP PROGRAM

```
    B      MATRIX         3 ROWS          4 COLUMNS
 4.0000000D 00  -1.0000000D 00  -3.0000000D 00   2.0000000D 00
-2.0000000D 00   5.0000000D 00  -1.0000000D 00  -3.0000000D 00
 2.0000000D 00   1.3000000D 01  -9.0000000D 00  -5.0000000D 00

    APSE   MATRIX         4 ROWS          3 COLUMNS
 9.5029697D-02  -5.6580181D-02   2.0318850D-02
-3.0790872D-02   3.3135355D-02   3.7824320D-02
-6.7364802D-02   3.1884964D-02  -3.9074711D-02
 5.0640825D-02  -3.6730228D-02  -8.9090341D-03

    AASA   MATRIX         3 ROWS          4 COLUMNS
 4.0000000D 00  -1.0000000D 00  -3.0000000D 00   2.0000000D 00
-2.0000000D 00   5.0000000D 00  -1.0000000D 00  -3.0000000D 00
 2.0000000D 00   1.3000000D 01  -9.0000000D 00  -5.0000000D 00

    AERR   MATRIX         3 ROWS          4 COLUMNS
-6.6613381D-16   6.6613381D-16   2.2204460D-16  -6.6613381D-16
 1.3322676D-15   0.0            -8.8817842D-16   6.6613381D-16
 2.2204460D-15   3.9968029D-15  -4.2188475D-15  -6.6613381D-16

    ASAA   MATRIX         4 ROWS          3 COLUMNS
 9.5029697D-02  -5.6580181D-02   2.0318850D-02
-3.0790872D-02   3.3135355D-02   3.7824320D-02
-6.7364802D-02   3.1884964D-02  -3.9074711D-02
 5.0640825D-02  -3.6730228D-02  -8.9090341D-03

    AERR   MATRIX         4 ROWS          3 COLUMNS
-1.3877788D-17   1.2143064D-17   3.4694470D-18
 2.3418767D-17  -1.0408341D-17   1.6479873D-17
 0.0            -8.6736174D-19  -1.4745150D-17
-1.8214596D-17   7.8062556D-18  -4.3368087D-18
```

(b) Output.

Figure 6.– Continued.

PSEUDO TEST PROGRAM CASE 2 FROM ASP MANUAL PAGE 137 VASP PROGRAM

A MATRIX 4 ROWS 4 COLUMNS

2.0000000D 00	1.0000000D 00	-2.0000000D 00	-2.0000000D 00
1.0000000D 00	2.5000000D 01	-8.0000000D 00	6.0000000D 00
-2.0000000D 00	-8.0000000D 00	4.0000000D 00	0.0
-2.0000000D 00	6.0000000D 00	0.0	4.0000000D 00

APSE MATRIX 4 ROWS 4 COLUMNS

5.5097063D-02	-1.1026095D-03	-4.6911023D-02	-6.3283103D-02
-1.1026095D-03	2.9737044D-02	-7.5512045D-03	9.7564235D-03
-4.6911023D-02	-7.5512045D-03	4.2366935D-02	5.1455110D-02
-6.3283103D-02	9.7564235D-03	5.1455110D-02	7.5111096D-02

AASA MATRIX 4 ROWS 4 COLUMNS

2.0000000D 00	1.0000000D 00	-2.0000000D 00	-2.0000000D 00
1.0000000D 00	2.5000000D 01	-8.0000000D 00	6.0000000D 00
-2.0000000D 00	-8.0000000D 00	4.0000000D 00	-2.2204460D-16
-2.0000000D 00	6.0000000D 00	0.0	4.0000000D 00

AERR MATRIX 4 ROWS 4 COLUMNS

-4.4408921D-16	0.0	4.4408921D-16	6.6613381D-16
-4.4408921D-16	-3.5527137D-15	2.2204460D-16	4.4408921D-16
6.6613381D-16	6.6613381D-16	-6.6613381D-16	-2.2204460D-16
2.2204460D-16	-2.2204460D-16	0.0	-4.4408921D-16

ASAA MATRIX 4 ROWS 4 COLUMNS

5.5097063D-02	-1.1026095D-03	-4.6911023D-02	-6.3283103D-02
-1.1026095D-03	2.9737044D-02	-7.5512045D-03	9.7564235D-03
-4.6911023D-02	-7.5512045D-03	4.2366935D-02	5.1455110D-02
-6.3283103D-02	9.7564235D-03	5.1455110D-02	7.5111096D-02

AERR MATRIX 4 ROWS 4 COLUMNS

-1.4745150D-17	2.7647155D-18	8.6736174D-18	1.3877788D-17
0.0	0.0	2.6020852D-18	8.6736174D-19
8.6736174D-18	1.7347235D-18	-5.2041704D-18	-1.1275703D-17
1.3877788D-17	-2.6020852D-18	-8.6736174D-18	-1.3877788D-17

(b) Output – Continued.

Figure 6.– Continued.

```
PSEUDO TEST PROGRAM   CASE 3                                                   VASP PROGRAM

    C       MATRIX          4 ROWS          2 COLUMNS
  0.0               0.0
  0.0              -3.9100000D 00
  3.5000000D-02     0.0
 -2.5300000D 00     3.1000000D-01

    APSE   MATRIX          2 ROWS          4 COLUMNS
  0.0              -3.1331471D-02   5.5012930D-03  -3.9518081D-01
  0.0              -2.5575417D-01   2.8046074D-04   3.8798916D-06

    AASA   MATRIX          4 ROWS          2 COLUMNS
  0.0               0.0
 -1.0889456D-17    -3.9100000D 00
  3.5000000D-02     0.0
 -2.5300000D 00     3.1000000D-01

    AERR   MATRIX          4 ROWS          2 COLUMNS
  0.0               0.0
 -1.0889456D-17     4.4408921D-16
  0.0               0.0
  2.2204460D-16     0.0

    ASAA   MATRIX          2 ROWS          4 COLUMNS
  0.0              -3.1331471D-02   5.5012930D-03  -3.9518081D-01
  0.0              -2.5575417D-01   2.8046074D-04   3.8798916D-06

    AERR   MATRIX          2 ROWS          4 COLUMNS
  0.0               0.0             0.0             0.0
  0.0               2.7755576D-17  -5.4210109D-20  -1.1013546D-18
```

(b) Output – Concluded.

Figure 6.– Concluded.

APPENDIX A

DESCRIPTION OF INTERNAL SUBROUTINES

25. READ1

DESCRIPTION

This subroutine reads a single matrix from cards, without a header card. It is called by READ, after the latter has read the header card. The dimensions of the matrix to be read are in array NZ. If this is zero, no array will be read. In any event, the routine then prints either the array just read, using NZ for dimensions, or, if NZ = 0, the array already stored, using NA for dimensions.

The subroutine reads the data from cards, each row of the matrix starting on a new card, using format (8F10.2). If the card data is in exponential form, it must use a D exponent.

USAGE

CALL READ1(A,NA,NZ,NAM)

Input Arguments

Matrix:	A (if NZ = 0)
Dimension array:	NA,NZ
Constant:	NAM, containing a four-character (or less) name for the matrix, which will be used by PRNT

Output Arguments

Matrix:	A (if NZ $\neq$ 0)
Dimension array:	NA

26. ASPERR

DESCRIPTION

This is an installation dependent subroutine. It is called by the various subroutines when they detect an error. It is intended to provide an error walkback, so that the programmer can determine which call of a given subroutine is in error. It also counts the number of errors and calls EXIT after ten entries into ASPERR.

USAGE

CALL ASPERR

It has no arguments. The user may, if he wishes, call this program to help him track down errors.

Subroutine ASPERR calls in turn a system program which provides the actual walkback. In Ames OS this system routine is called ERRTRA, while in Ames TSS, it is called TRACE. The calling statement should be changed to match the user's operating system, or else deleted altogether.

27. BLKDATA

DESCRIPTION

This is an installation dependent subroutine. It loads certain common areas used by VASP with appropriate constants as follows:

1. COMMON/FØRM/NEPR, FMT1(6), FMT2(6)

These three variables control the printing procedure, and are set to 7, (1P7D16.7), and (3x,1P7D16.7), respectively. They assume a line length of at least 115 characters.

2. COMMON/LINES/NLP, LIN, TITLE(23)

NLP controls the number of lines per page, and is set at 45 to agree with the NASA–Ames system. It should be changed to match each installation.

LIN is a counter which keeps track of the number of lines printed on each page. It is incremented and used only in LNCNT.

TITLE contains 72 blank characters, which can be loaded as desired by use of RDTITL, plus 20 more characters containing "VASP PROGRAM." Subroutines LNCNT prints TITLE at the head of every page.

3. COMMON/MAX/MAXRC

MAXRC is used by most subroutines to check the reasonableness of the matrix dimensions. The user should set MAXRC to match the storage available for each matrix. It is preset to 6400.

28. PSEU

SUMMARY

PSEU is a FORTRAN routine to find the Moore-Penrose generalized inverse of a non-negative definite double-precision matrix. It has a separate entry PSEUP for input of a matrix that is already symmetric. A symmetric matrix is always used for the actual diagonalization process. This process is done in a self-contained subroutine, ANDRA. The routine "never" fails, since it includes the singular case. However, it may fail to give the correct rank. To control this, an option to do side calculations is available. After the first pivots have been found, if the rank is not maximum, the result of each pivot step is used in two axiomatic expressions (subroutine BDNRM). This side calculation yields a measure of the worth of the pseudoinverse obtained so far. This result is multiplied by a parameter factor raised to the power of the current rank (nonlinear penalty function). The routine can backtrack from the first bad step and stop with the previous rank. It has an option to do the minimum calculations for getting a rank only. The generalized inverse is useful for least-squares solutions of $Ax = b$; it works when A is singular. This method is best suited to symmetric matrices. The routine has suitable error exits.

USAGE

CALL PSEU(A,B,C,EE,DEP,IP,D)

or

CALL PSEUP(A,B,C,EE,DEP,ID,D)

Note: PSEUDO uses PSEU entry.

Input Arguments

		Description
Matrix:	A	The array to be inverted, left intact, must be symmetric if PSEUP call is used. Non-negative definite, or nearly so.
Control arrays:	DEP	Values DEP1, DEP2, DEP3
	DEP1	Default: If zero, user gets 2.D−6 used instead. This number is multiplied times the largest magnitude on the diagonal of B at start. If any trial pivots are found *less* than this, they are avoided as *zero*.
	DEP2	Default: If zero, user gets 1.D0 used instead. Needed only if iteration. The routine computes two numbers, p, q, which would be zero if the first two Moore axioms were satisfied. This number is raised to the power of the number of pivots found as a factor to use to make the product with the sum of p and q larger. Making this product larger tends to make the routine reject the current pivot. Values between 1 and 2 work for ordinary purposes. Note: PSEUDO uses default values of DEP1 and DEP2.

	DEP3	This is for output only. It holds the last pivot actually accepted. This gives the user or calling routine an estimate of the size of pivots found, in case effective rank is not that desired, operating with given value of DEP1. If iterating, this may be the last pivot *rejected.*
	IP	Parameter array of integers IP1, IP2, IP3, IP4.
	IP1	If zero, do *not* iterate with side calculations. If 1, iterate. Note: Other values should not be used, since DECOM employs peculiar values.
	IP2	If zero, do all calculations, otherwise do rank only. Note: Setting this to zero for each call is very useful in avoiding confusion between ranks determined from different calls. Used also to *output* the effective rank. PSEUDO sets IP1 and IP2 to zero.
	IP3	The row size of the matrix input.
	IP4	The column size of the matrix input. Note: IP4 need not be specified for PSEUP entry.

Output Arguments

Matrix:	B	Holds the pseudoinverse output. (In rank only case, holds a diagonal matrix with 0's and 1's corresponding to pivots accepted or rejected.)
Matrix:	C	In nonsingular case, holds the matrix T of the diagonalization case. In singular case, holds that certain matrix U described in ASP manual.
Matrix:	EE	Holds the pseudoinverse of the *original B.* Note: A and B are the same size. The other matrices are square, of the size of C, which is determined by the *smaller* dimension of A. D is either five times the size of C, if iterating, or the same size as C.

Matrix:	D	In the nonsingular case, D holds a copy of the B formed from A. (It equals A for a PSEUP entry.) In the singular case, it holds a pseudo-inverse for a "B" permuted so that independent variables are all moved to the left-most positions. Note: D has possibly four other matrices. Let these be D1, D2, D3, and D4, in order. They are used only if iterating (D1 also used by DECOM). D1, D2 hold old results. D3, D4 holds intermediate values when doing the side calculations. PSEUDO does not provide for D1 through D4.

Notes on Usage

Symmetry

This method is well suited to symmetric, non-negative definite matrices. The PSEUP entry assumes this. Matrices formed by computer arithmetic will *not* always be symmetric. Hence, the routine always forces the symmetric matrix B to actually be symmetric, by taking the average of the element and its transpose. The nonsymmetric entry, unfortunately, approximately squares the ratio between largest and smallest eigenvalue. There is a nonsymmetric feature. The routine choses AA^T, instead of the other way around, if A is a square matrix. This arbitrary choice agrees with the DECOM routine and the ASP routines. As a result, in the singular case, multiplying A by its pseudo-inverse from the *left* is more likely to give a diagonal matrix of 1's and 0's, than multiplying from the *right* side of A.

Pivot Size

DEP1 is used to compute a "smallest allowable pivot." In no case is it reasonable or desirable to worry about exact equality in the use of such tolerances. Fortunately, work with ill-conditioned systems shows a series of pivots that decrease steadily in magnitude. Furthermore, the first "bad," erroneous pivot is, at most, 10 to 1000 times smaller than its predecessors. Since ANDRA is choosing largest pivots first, the user has considerable latitude in actual choice. All positive elements can be accepted, if the matrix is *known* to be nonsingular, by choosing DEP1 very small.

By choosing DEP1 very large – say, nearly 1.0 – the routine can be forced to reject pivots after the first. At present, there is no way of making it start iterating without having found at least one pivot. In other words, ANDRA always finds all the pivots it can before any side calculations are done. If this first rank is maximal, it never iterates. The first pivots are not in doubt, so these rules are more efficient. The routine always uses a tolerance for pivot acceptance; however, it uses a new tolerance 50,000 times smaller than the last pivot found, for each call to find one pivot in iterative mode. The expensive test of matrix norms is avoided when no new pivot occurs. The PSEU routine has only a finite number of tries to find a new pivot before it quits. The exact number is the same as the maximum rank. Since ANDRA has usually found several pivots initially, this is ample.

Iteration

If DEP2 is larger than 1, it is raised to a power, used as a factor, and tends to make the routine stop with a smaller rank. DEP2 of 1 actually works for most iterations.

Subroutine ANDRA

The basic algorithm can be used as a separate routine by itself (see ANDRA documentation). The routine requires considerable setting and testing of parameters. It has an escape exit for too many iterations (calls to find only one pivot) without finding any. It returns a matrix, T, such that, if X if pseudoinverse of positive definite matrix A, then

$$T^tT = X$$

Accuracy

In double precision, the accuracy has been very good. Maximum accuracy can be obtained by using symmetric matrices and the PSEUP entry. The test program included in this manual as example 6 shows errors (determined by calculating $AA^{\#}A-A^{\#}$ and $A^{\#}AA^{\#}-A$) on the order of 10^{-15} or less.

The routine was also tested on the ill-conditioned 7 X 7 matrix in the ASP manual (NASA CR 475, p. 150). The exact inverse is given on page 151, and the error obtained from the ASP program using the equivalent of the PSEUP entry (p. 152) is on the order of 10^{-1}. The error obtained using the VASP program and the PSEU entry was on the order of 10^{-5} or less.

Singular Case

The routine forms a new inverse from the original symmetric matrix. Since there are several steps more between the inverse and the original input A, it is only natural that accuracy should fall off. In many cases, this inverse will give a diagonal matrix of 0's and 1's when used as a left inverse of A (or possibly as a right inverse). The work of reinverting B requires no extra matrices; it does destroy the usual values of C and D. No iteration can be done in the stage *after* B is found to be singular. It can be asked for in the starting stage. Error exits are made if the rank changes during reinversion. The smallest allowable pivot is redetermined.

Error Exits (Messages)

The error exits are reasonably self-explanatory. Unless otherwise noted, the errors cause a return from PSEU without completion of the calculations. Subsequent calculations in other portions of the program are suspect.

Message	Explanation
Dimension error	The total number of matrix elements was too large or too small.
Diagonal elements of matrix = 0	Symmetric matrix B has no positive diagonal elements. Check input A.

Rank has decreased	Singular case. Reinverting, and the new rank is *less* than that of the earlier phase.
Rank has increased	Singular case. Reinverting, and the new rank is *greater* than in the earlier phase. Computation continues.
Rank greater than matrix size	RANK returned from ANDRA is greater than maximal rank.

Timing

The ANDRA routine by itself is very fast. The iteration mode is slower by a large factor than the regular mode of subroutine PSEU.

The time estimates below (in hundredths of a second) are as used on the NASA Ames 360/50. High and low estimates are given, because real-time figures reflect an unknown percentage of time devoted to another CPU user.

Case	High	Low
PSEU, 2 X 2 matrix	2	1
PSEUP, 4 X 4 matrix, reinvert	14	10
PSEU, 7 X 7, no pivot rejection	42	30
PSEU, 7 X 7, rank 6, reinversion	103	62
PSEU, 7 X 7, iteration, no tests	53	30
PSEU, 7 X 7, iteration, one test	182	118
PSEU, 7 X 7, iteration, some tests of pivots	253	170
PSEU, 7 X 7, iteration, many pivot tests	501	286
PSEU, 7 X 7, iteration, nearly all tests	607	324
PSEU, 4 X 2, reinversion	3	1
PSEU, 4 X 2, reinversion	2	2

METHOD

Summary of Method

PSEU has two entry points. The nonsymmetric entry forms A^tA or AA^t, whichever is smaller. At the end, A^t is used again to form the pseudoinverse. Square A uses A^tA. The result is always forced symmetric afterward, even for symmetric entry. ANDRA is called to diagonalize this result in B. Most of the pivots are found and the steps made on the first call. If not iterating, this part is not repeated. If singular (rank of symmetric input not maximal), a transforming matrix is computed. A copy of the original symmetric B is transformed and reinverted by ANDRA. The result is retransformed by premultiplication and postmultiplication. If iterating, the pivot tolerance is decreased and ANDRA is called to find one pivot at a time. A side calculation is done to measure the quality of pseudoinverse formed at each step. The routine backs up one step and stops with rank one less if it makes a bad step. The result, if singular, is sent through the reinversion above. The use of PSEUP by DECOM avoids reinverting in the singular case, also it never uses a nonsquare input. There is a "find rank only" option.

If PSEU is used without iteration, four I/O matrices are needed plus a dummy matrix. Iteration uses four additional dummy matrices. Iteration cannot be done during the reinversion. Besides those mentioned, entries BDNRM, MULT, and NORM are used for iteration. TTRM is also used except in rank only case.

ANDRA (diagonalization algorithm). For a detailed description of the method, see the documentation of ANDRA itself. A mathematical description and examples are given in NASA CR-475. Subroutine PSEU calls ANDRA to do each pivoting step, after first forming a *symmetric* matrix B, which is indeed forced to be perfectly symmetric.

The first call of ANDRA is an initialization call. An identity matrix T is formed. The rank counter is set to zero. On an initialization call, the routine proceeds to search the diagonal for pivots, as always. But after finding a pivot, it always goes back and looks for another pivot, regardless of the iteration option. The process of searching for pivots continues until the number of tries is one greater than the row size (no such test is made in the iteration case). If the routine fails to find a single pivot in the initialization call, it exits with an error message. Pivots are accepted if and only if they are not less than a threshold input at every call. Supposing that a pivot has been found in the diagonal, the next step is always the same. First the pivot is reduced to unity. That is, both the pivot row and column are divided by the square root of the pivot in B. Only the *row* of T is so reduced. The next step is to eliminate the pivot coefficient from all other rows not yet used as pivots. This part is the same as in other inversion methods. Both B and T are treated exactly alike here. Note that the actual algorithm checks the diagonal of a row to see if it is already marked as a pivot. If so, that entire row, and the row in T, are skipped. The pivot is then marked by an artificial code. The routine always tests for this code and does not use this row again. The code is put in the actual pivot position. Thus the rows and columns are left in their starting places in the working matrix B. PSEU converts the result to a matrix of 1's and 0's that shows the independent and dependent variables.

The code is tested for an integer. This is a considerable economy. The resulting T is never singular. If B were nonsingular and X the desired inverse of B,

$$T^t T = X$$

This part is done by subroutine entry TTRM, using coding shared with the iteration method. The final answer is put back in matrix B. (PSEU always uses the original A again rather than the original B, after this to give an answer for A. Thus, ANDRA is always supplied with a symmetric matrix B.)

If B were singular at the start, a further reinversion would have to be done. See the next section.

The Singular Case

Suppose that the rank of B in the diagonalization by ANDRA does not turn out to be maximal, then PSEU must perform a number of matrix multiplications and call ANDRA and TTRM to reinvert. The accuracy is bound to suffer, even though the reinversion is done on an exact copy of the original B. A very short justification is given belov , followed by a close description of how the work is actually done.

There exists a permutation matrix P, such that

$$\bar{E} = PTBT^tP^t$$

is a matrix of 0's and 1's (were it not for round-off error), with all the 1's contiguous, starting in the first diagonal. If B had been so permuted before diagonalizing, then this different T resulting would be the one that gives an inverse that corresponds correctly to the old. But, since one is using a premultiplication and a postmultiplication, simple substitution of a permuted matrix does not work. (It would if matrix multiplication were commutative.) Thus, if it is necessary to transform the original starting B, reinvert, then transform back again.

The *permuted* form of T (which does not actually occur) has a nonsingular corner submatrix, followed by the rest of diagonal set to 1's. These latter 1's correspond to the dependent equations of the original.

The rule for constructing the transforming matrix U is given below. This matrix is made from T and put into the same storage T. The explicit construction of U is more efficient (in FORTRAN). From here on, the explanation concerns what is actually done, rather than the mathematical reasons.

Let d_i denote the *ith* diagonal element of B. (In case the reader has forgotten, this has been changed to a diagonal matrix of 0's and 1's.) Given T, there are two cases:

Case One: For U_{ij} not on the diagonal
Use $-t_{ij}$, if $d_i = 0$;
Use 0, if $d_i = 1$

Case Two: For U_{ij} on the diagonal
Use the corresponding value of d_i

Next, using a copy of the original B, form

$$C = U^tBU$$

The result is actually put in the same storage that held B originally. The smallest allowable pivot for ANDRA is recalculated. This result, C, is sent to ANDRA to do the diagonalization again. The fact that C has rows and columns of 0's that ANDRA has to skip makes the diagonalization inefficient, but this cannot be helped. No iteration is done here. Let T_2 denote the result of this second ANDRA call. Then the new pseudoinverse is:

$$X_2 = T_2{}^t T_2$$

Transform this back to get a correct answer:

$$X = UX_2 U^t$$

The rest of the computation is as usual. Note that if the rank changes in the second ANDRA call, error exits are taken.

Iteration

The main method itself is purely algebraic. The iteration option is a way of estimating the amount of error in the generalized inverse and using this to stop with a smaller effective rank. Let B denote a matrix and X its pseudoinverse (after taking so many pivot steps in ANDRA). Then the two Moore-Penrose axioms read:

$$BXB = B$$
$$XBX = X$$

If the iteration mode is selected, ANDRA first finds all the pivots it can. Then subroutine BDNRM is called twice. Each call returns the value

$$\text{norm}(Q^*P^*Q - Q)/\text{norm}(Q)$$

The values of P and Q are B and X in one call, X and B in the other. The resulting two small scalars (which would be zero if the axioms were perfectly satisfied) are added together. The result is taken as a factor times DEP2 raised to the current number of pivots. From successive iterations, one obtains a sequence of positive numbers, decreasing as one approaches the largest possible rank. As long as the new result is not larger, then a new pivot is searched for. If not, PSEU reverts to the previous values, before the current pivot was used.

In practice a number of modifications are made. First, the pivot used last is returned as DEP3, even if rejected, so that the user can reconsider acceptance of it. Second, if maximum rank is achieved prior to iteration, no side calculations are done. Third, the smallest possible pivot allowable is set to 0.00002 times the most recent pivot in order to reject many spurious pivots without doing the lengthy side calculations. This modification is based on actual observation of pivot behavior. The successive pivots of an ill-conditioned matrix usually decrease fairly rapidly. But there is usually a hugh jump in order of magnitude between the last good pivot and the first bad one. Parts of the side calculations are actually done in single-precision, to save time. Please note that a single iteration, besides the ANDRA call, makes ten subroutine calls, and one library routine call. Naturally, this is slow.

Matrix Storage Flow

This section uses the same names as the Fortran IV routines. It tells what is put into each matrix of PSEU at various times. The call is CALL PSEU(A,B,C,EE,DEP,IP,D). The matrices A and B are the same size (possibly nonsquare). Matrix C is square with dimension equal to the smaller dimension of A. The other matrices are the same size as C. Matrix D is divided into five matrices. Let these be denoted as D, D1, D2, D3, and D4. The last four are used only in iteration.

Maximal Rank Case

A symmetric matrix from A is placed in B (either directly, as in PSEUP, or indirectly, from matrix multiplication). A copy of B is put in D, unless the rank only, *no* iteration is used. ANDRA is called to diagonalize B and place the result in C.

If the result is accepted, TTRM puts the generalized inverse of B into EE. Then the inverse of A is put into B. The A transpose may have to be used to get an answer for A.

Singular Case

The matrix U of the method is computed from C and put into C. (D holds original B.)

$$EE = C^t \times D$$
$$B = EE \times C$$

ANDRA is called to diagonalize B. Answer goes to EE. TTRM puts pseudoinverse of B into D.

$$B = C \times D$$
$$EE = B \times C^t$$

The pseudoinverse is now in EE, where the maximal rank case puts it. Routine now forms pseudoinverse of A in B.

Iteration

Before each call of ANDRA the current values of B and C are stored in D1 and D2, respectively. B and C are changed when a new pivot is used in ANDRA. BDNRM computes a number to decide if the pivot is to be rejected. EE, D3, and D4 are used as working storage in BDNRM. EE actually has a matrix put in it that would be zero if the Moore-Penrose axiom were perfectly satisfied. If the pivot is rejected, the old values from D1 and D2 are put back into B and C. The work of the singular case is done next if the call was not made from DECOM.

Rank Only

If iteration is used, a full complement of matrices must be used. In the ordinary case, matrix D may be omitted, and also matric E is not used. Naturally, no pseudoinverse is returned.

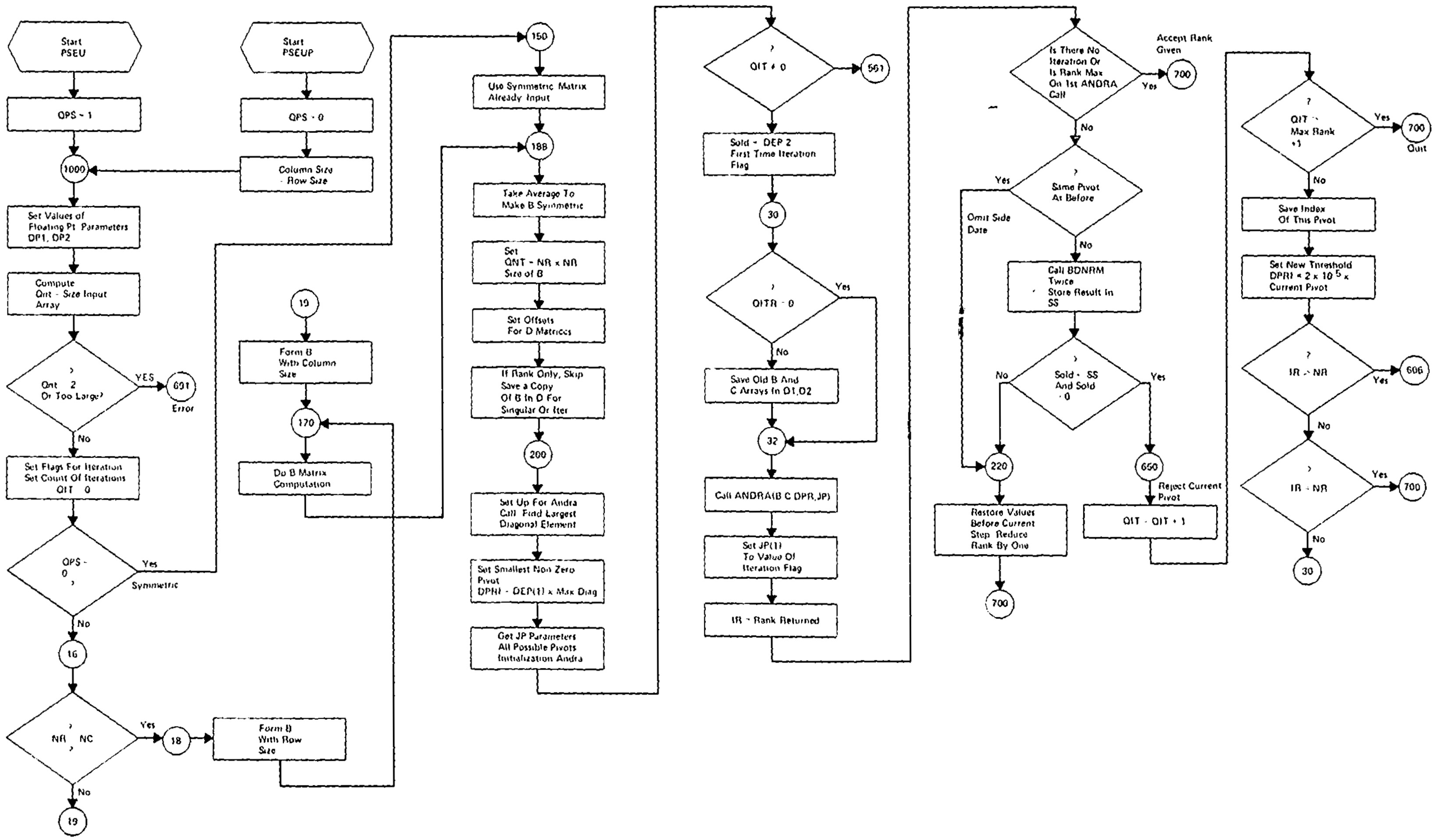

Figure 7.– Information Systems Co. flow chart – subroutine PSEU (A,B,C,EE,DEP,IP,D).

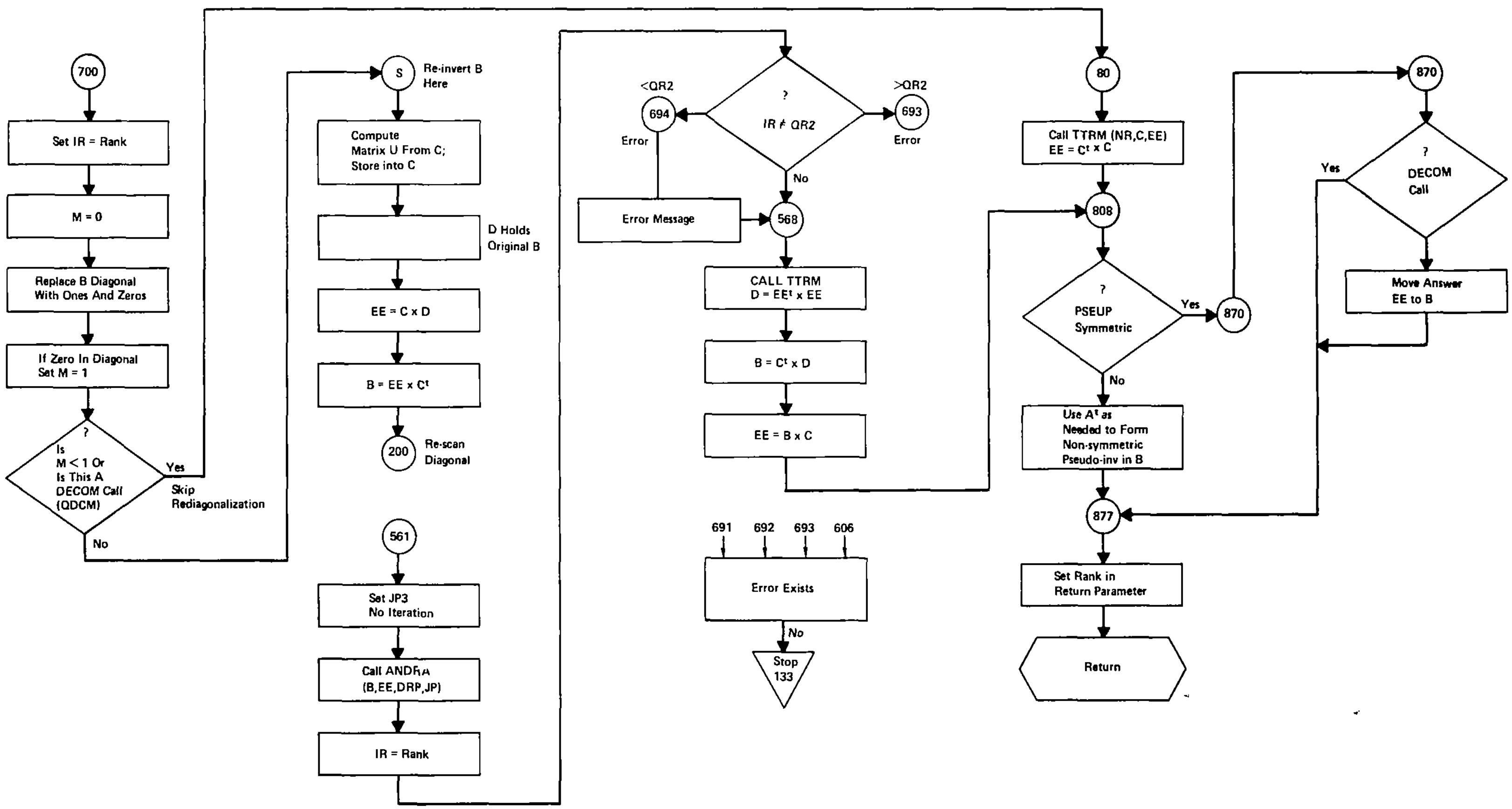

Figure 7.– Concluded.

29. BDNRM

DESCRIPTION

This subroutine computes the quantity

$$\text{norm}(QPQ^T - Q)/\text{norm}(Q)$$

where the values of P and Q are in the square arrays CT and EE or EE and CT, depending on the sign of NR. If P = Q#, the return value is zero. This routine can thus be used to test the quality of a pseudoinverse.

USAGE

CALL BDNRM(NR,CT,EE,D,KRV)

Input Arguments

Matrices:	CT, EE with dimensions NR X NR
Constants:	NR, size of matrices and the sign controls multiplication procedure

Output Arguments

None:	This is a function subroutine

Dummy Arguments

Matrix:	D dummy array of size $5*NR^2$
Constant Array:	KRV designates location of submatrices of D $KRV1 = NR^2$ $KRV2 = 2*NR^2$ $KRV3 = 3*NR^2$ $KRV4 = 4*NR^2$

30. ANDRA

SUMMARY

ANDRA is a Fortran routine to diagonalize a positive definite symmetric matrix. The routine was originally designed to be used by subroutine PSEU. The routine has a parameter to command it to initialize on the first call. Two different modes can be used for pivoting steps. In the first mode, the routine does only one pivot to eliminate only one row at a time. In the second mode, as many pivots as possible are done in one call. Pivots are chosen in order of decreasing magnitude. They are rejected if smaller than a parameter threshold. The original matrix input is destroyed and replaced with artificial values. However, symmetry is kept after each pivot. The answer matrix, T, is such that if X is the inverse of the input,

$$X = T^t T$$

The routine has error exits for matrices of the wrong size, and for those that allow no pivot on the first try.

USAGE

CALL ANDRA(B,T,DPR,JP)

Name	Description
B	Input symmetric matrix. Destroyed.
T	Answer. T^tT = inverse of B.
DPR	Parameter array of size 2.
DPR1	DPR1 is the tolerance for trial pivots. Any less than this are rejected as zero.
DPR2	DPR2 is the last pivot actually used. Unchanged if no new pivot found.
JP	Integer parameter array of size 5.
JP1	Zero if all pivoting to be done on one call; nonzero if only one pivot per call.
JP2	Zero if initialization call. Subroutine sets to one when a pivot is found.
JP3	Holds the effective rank = number of pivots found.
JP4	The integer giving the row and column size. May range from one to a nominal figure.
JP5	The integer row where the last pivot was found. The rows are left in the same positions as in the input matrix.

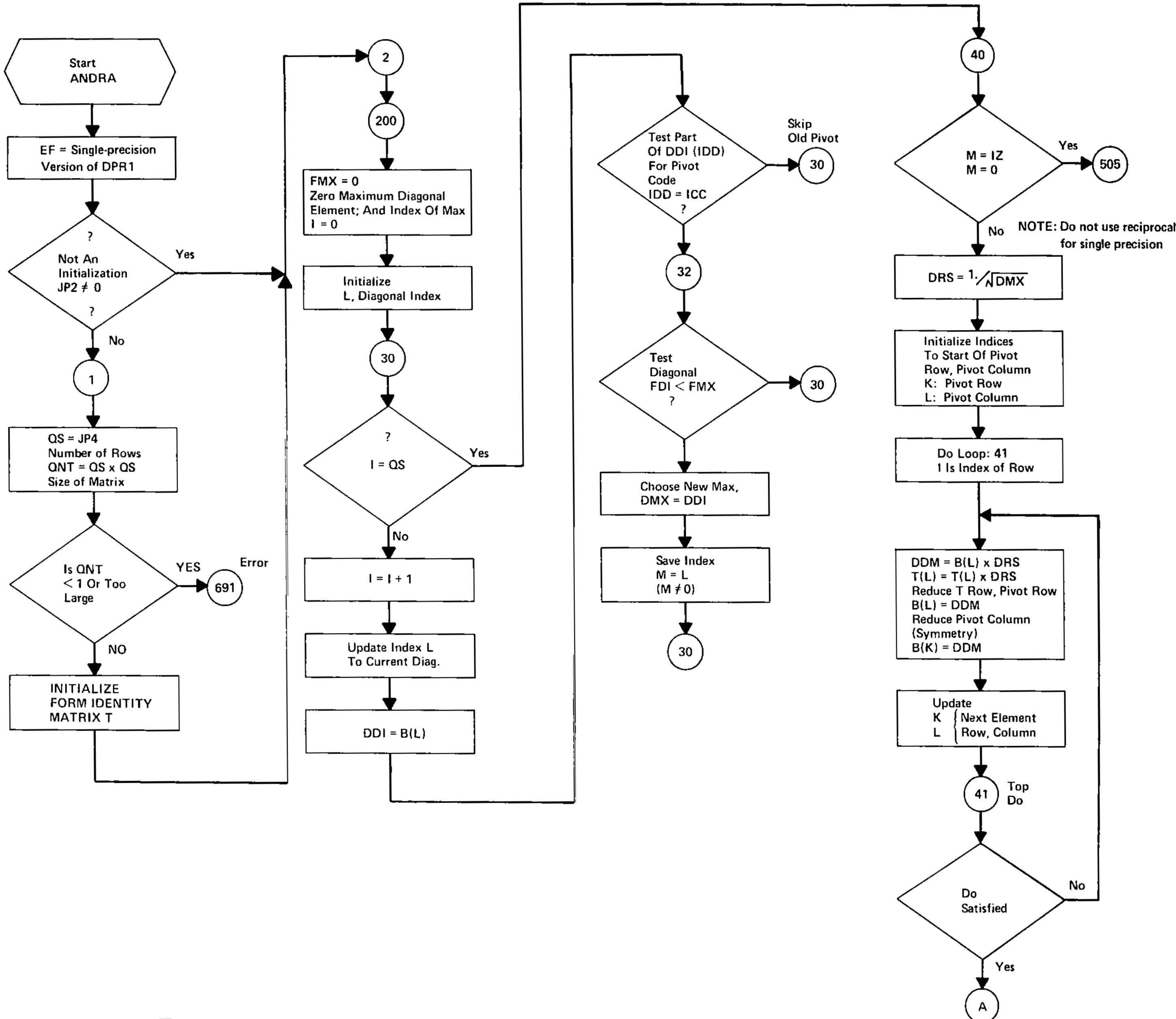

Figure 8.– Information Systems Co. flow chart – subroutine ANDRA (B,T,DPR,JP).

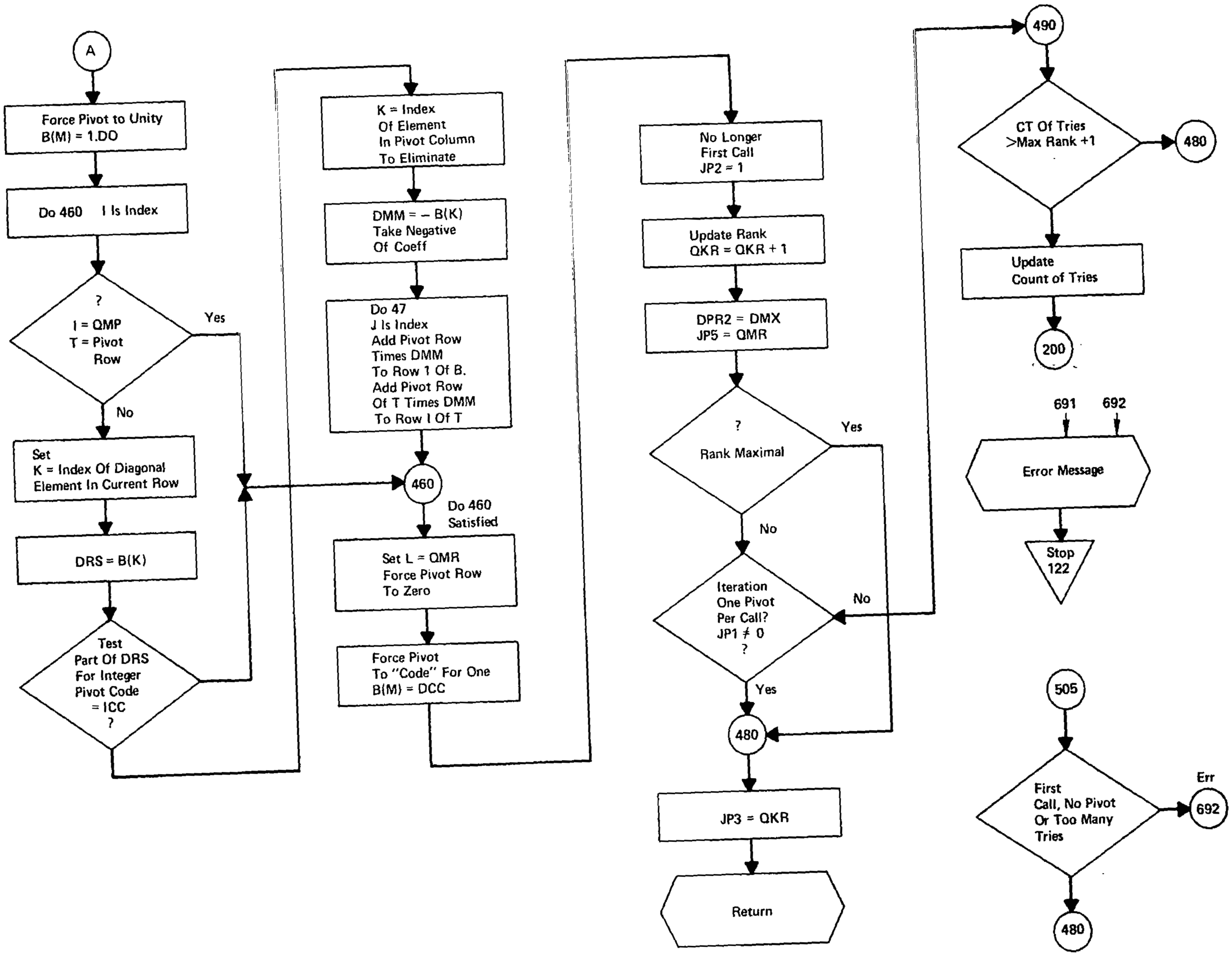

Figure 8.– Concluded.

METHOD

Mathematical

The method is described in the ASP manual, pages 137 to 139.

The square matrix T is initialized to be the identity.

Step 1

The diagonal of B is scanned to find the largest pivot. Pivots are only taken from the diagonal. If no pivot is found, skip to step 3.

The square root of the pivot is taken. The pivot row and pivot column are divided by the square root. Thus, the pivot, at the intersection of the row and the column, is reduced to unity. The corresponding *row* of T is also divided by the square root.

Step 2

The new, reduced pivot row is used to eliminate the elements of the pivot column. Let K be the pivot row and column. The pivot row is multiplied times the element in the j,k position. The resulting row-vector is subtracted from the jth row. This process is repeated for each row j that has not yet been a pivot row. Exactly the same operations are carried out on the corresponding rows and columns of T, except that, of course, the multiplier for a pivot row comes from B. Then the pivot row of B, except for the pivot, is set to zero. The pivot row and its corresponding row in T are never used again.

Step 3

If the rank is maximal, exit. If no pivot has been found, a test is made to see if this should be an error exit, or normal exit. Otherwise, repeat step 1.

Computational

In practice, a number of modifications are made. The actual calculations are rewritten to optimize speed and storage. The reciprocal of the square root is used, instead of a division. For single precision, straight division would probably be best. In step 3, an artificial code is put into the pivot position. This code is chosen as one that cannot be the result of floating-point arithmetic. Such a technique works in a great many different Fortrans. If a row is found to be marked by a pivot code, it is skipped in steps 1, 2, and 3 above.

The pivot position is forced to be exactly 1 before step 2 is applied. The pivot-code is actually tested for as an integer. The pivot size is tested for in single precision. These modifications are for speed. A count is kept of the number of pivot searches. If this count is one greater than the number of rows, the routine always stops searching for pivots. The result, if B has maximum rank, is a matrix T such that T^tT = inverse of B. The input B consists of 0's everywhere except the diagonal, which holds pivot codes.

Error Messages From ANDRA

Message	Explanation
Dimension error	The total number of matrix elements was too large or too small. The parameter JP(4) cannot be less than one nor more than MAXRC.
Finds no pivots	ANDRA could find not a single pivot in its very first search of diagonal.

31. DECOM

SUMMARY

Fortran IV subroutine DECOM generates four double-precision output matrices from the symmetric, non-negative definite input matrix B. One output is a matrix S such that if E is a unity matrix of rank r, then

$$B = SEE^tS^t$$

This matrix is obtained as the inverse of a matrix T, by calling subroutine INV; T comes from subroutine PSEU. It is defined by $TBT^t = E$, a diagonal matrix with elements 0 or 1. E is also returned, along with a permutation matrix P such that

$$PEP^t = I_r$$

a diagonal matrix with all 1's moved to the uppermost left corner. Given these matrices, and the ability to find a pesudoinverse of A, a decomposition of any matrix is possible. PSEU and DECOM are called and the matrices then multiplied as described in the method to give a Kronecker decomposition. The routine calls PSEUP and INV to do most of the calculation. Besides returning the matrices P and E, it does nothing that could not be done by successive calls of other matrix routines. It has parameters and error exits similar to that of PSEUP.

USAGE

CALL DECOM(A,B,C,E,J,DCM,KP,D)

Arguments	Description
A	The symmetric non-negative definite input.
B	The output matrix E, diagonal of 0 and 1, with 1's in the independent columns. B, C, E, J, D, and D1 are all of same size as A.
C	The output T, such that TAT^t = diagonal of 0's and 1's.

E	Holds the inverse of A (B does not hold the inverse of A). (Not E of ASP.)
J	A square integer matrix for housekeeping in INV and DECOM.
DCM	Parameters, just as in subroutine PSEUP.
DCM1	Multiplied times the largest magnitude of diagonal of A, to give a lower limit for an acceptable pivot in PSEUP. Set at $2(10)^{-6}$ if zero is input.
DCM2	Used only if the user elects to iterate in PSEUP. Set at 1.D0 (no effect) if zero is input. Note: DECGEN uses the default options for DCM1 and DCM2.
DCM3	The last pivot accepted by subroutine PSEUP, during diagonalization of input matrix A.
KP	Integer control parameters, just as for subroutine PSEUP.
KP1	Zero, do not iterate in PSEUP. One, iterate in PSEUP.
KP2	Zero, do all calculations. Nonzero, do rank only. Changed to reflect the rank on *output*. Should be set to zero or minus one before each call. Note: DECGEN uses KP1 and KP2 = 0.
KP3	The row size of the matrix input.
KP4	The column size. Note: This parameter is forced negative as a signal if T cannot be inverted by INV.
D	D has five parts, as does the "dummy" array in PSEUP. Let these be denoted D, D1, D2, D3, and D4. These five equal arrays must be included in the size of parameter D if iteration by PSEUP is selected. If no iteration is used, D2,D3, and D4 may be omitted. D holds the *inverse* of output C. D1 holds the permutation matrix P. Note: If *rank only* is computed, D1 is computed, but D is not. A, B, C, and D1 are thus the only matrices with useful values returned.

METHOD

The results from DECOM are an effective rank r; matrices B and D, which are used in further calculations to get a Kronecker decomposition, or to see which variables are dependent; and the permutation matrix P in D1. This section describes the sequence to obtain the Kronecker decomposition in two different cases. The goal is two matrices G and H. DECOM does not produce these matrices; they are produced either by DECGEN or by the user according to the following steps.

Let R be the matrix to decompose. Matrices G and H are desired such that

$$R = HG$$

H is to have only r nonzero columns; G is to have only r nonzero rows. Small r is the rank of R.

Case 1

Matrix R is symmetric, non-negative definite. Input R as the square input A to DECOM. Then H and G are produced afterward from the matrices in the call statement as follows:

Parameter B is a diagonal matrix with r 1's; H and G are computed by:

$$H = D \times B$$
$$G = (D \times B)^t$$

$A = \text{original} = DBB^tD^t$

Case 2

R is nonsymmetric, possibly not even positive definite. Form RR^t (subcase a) or else form R^tR (subcase b). The subcases are chosen to give the smaller dimensions. If R is square, use RR^t to agree with both PSEU and DECOM. Let this symmetric result be the input A to DECOM as in case 1. Obtain D and B as before and save them. In subcase a, $X = R^t \times E$, but in subcase b, $X = E \times R^t$. Then for subcase a, take

$$H = DB$$
$$G = (XDB)^t$$

Similarly, in subcase b, take

$$H = X^tDB$$
$$G = (DB)^t$$

Note: The H and G matrices produced have the same dimensions as the smaller dimension of R. If the rank of R is not maximal, there will be zero rows or columns in H and G. If the matrix D1 is used instead of B in the above calculations, the zero rows or columns will be at the right or bottom, and the dimensions may be easily reduced. This latter is the procedure used in subroutine DECGEN.

Computation

In practice, the subroutine is very short; it calls on PSEUP and INV to do the computations. No flowchart is needed, since there are no loops of any consequence.

Step 1

The matrix size is tested for reasonableness, with an error exit if it is not. KP(1) is set to special negative values to suppress reinversion by PSEUP, and to change somewhat the matrix outputs. This change is not discussed in PSEUP.

Step 2

Entry point PSEUP is used to diagonalize the input. C holds a matrix T such that $TAT^t = B$, a matrix of 0's and 1's. If the rank only option is input, the routine skips to step 4.

Step 3

Subroutine INV puts an inverse of T into D. The flag PIV is tested. If zero, INV failed; the routine prints an error message. INV uses matrix J.

Step 4

The matrix E, which is matrix of 0's and 1's, is scanned along its diagonal. A matrix P of 0's and 1's is constructed such that

$$PEP^t = I_r$$

I_r has all 1's moved to extreme upper left corner. A record of successive diagonal positions that are 0 is kept. As each 1 is found in the diagonal in position k, the record is checked to see if there is an earlier 0 (or 1) that needs to have a 1 permuted into its place j by permutation p. If so, a 1 is put into position j, k of P. Premultiplication by P will move position k, k to j, k. Postmultiplication by P^t will move j, k to position j, j. Position k, k is also marked as a hole that could be filled by a 1 lower on the diagonal, since it vacates its old position. The record in the first column of J has the structure of a queue. Matrix P is in D1, the second matrix of dummy array D.

Step 5

Return.

NOMENCLATURE

The nomenclature used in DECOM is compatible with that used in PSEU, but differs from that used in the ASP manual description of the decomposition routine, p. 154. Also, since DECGEN requires dummy storage, the nomenclature there is different again. The following table lists the correlations:

DECOM	DECGEN	ASP
A	DUM1	AA^T
B	DUM(N7)	E
C	DUM(N4)	
E	DUM(N5)	
D	DUM(N2)	S
D1	DUM(N3)	P
J	DUM(N6)	

APPENDIX B

LISTINGS OF ALL VASP SUBROUTINES

```
      SUBROUTINE READ (I, A, NA, B, NB, C, NCX, D, ND, E, NE)
      DIMENSION A(1), B(1), C(1), D(1), E(1)
      DIMENSION NA(2), NB(2),NCX(2), ND(2), NE(2),NZ(2)
      DOUBLE PRECISION A, B, C, D, E
      READ(5,100) LAB,               NZ(1), NZ(2)
      CALL READ1(A, NA,NZ,  LAB)
      IF(I .EQ. 1) GO TO 999
      READ(5,100) LAB,               NZ(1), NZ(2)
      CALL READ1(B, NB,NZ,  LAB)
      IF(I .EQ. 2) GO TO 999
      READ(5,100) LAB,               NZ(1), NZ(2)
      CALL READ1(C, NCX,NZ, LAB)
      IF(I .EQ. 3) GO TO 999
      READ(5,100) LAB,               NZ(1), NZ(2)
      CALL READ1(D, ND,NZ,  LAB)
      IF(I .EQ. 4) GO TO 999
      READ(5,100) LAB,               NZ(1), NZ(2)
      CALL READ1(E, NE,NZ,  LAB)
  100 FORMAT(A4,4X,2I4)
  999 RETURN
      END
```

```
      SUBROUTINE RDTITL
      COMMON /LINES/NLP,LIN,TITLE(23)
      READ  (5,100)  (TITLE(I),I=1,18)
  100 FORMAT  (18A4)
      CALL LNCNT(100)
      RETURN
      END
```

```
      SUBROUTINE  PRNT(AR,NAR,NAM,IP)
C     SUBR PRNT    PRINTS DOUBLE PRECISION MATRIX
      COMMON  /FORM/NEPR,FMT1(6),FMT2(6)
      COMMON/LINES/NLP,LIN,TITLE(23)
      COMMON  /MAX/MAXRC
C- NOTE NLP NO. LINES/PAGE VARIES WITH THE INSTALLATION.
      DATA KZ,KW,KB /1H0,1H1,1H /
      REAL*8 AR                                                    D
      DIMENSION AR(1),NAR(2)
      NAME = NAM
C-IF IP =1,HEADLINE SAME PAGE, IF IP =2,  HEADLINE, NEW PAGE
C      IP=3,  NO HEADLINE, SAME PAGE,    IP=4, NO HEADLINE, NEW PAGE
      II = IP
      NR=NAR(1)
      NC=NAR(2)
      NLST = NR * NC
      IF(NLST .GT. MAXRC .OR. NLST .LT. 1.OR.NR.LT.1)  GO TO 16
      IF(NAME  .EQ. 0) NAME = KB
C- SKIP HEADLINE IF REQUESTED.
      GO TO (11,10,132,12),        II
   10 CALL LNCNT(100)
   11 CALL LNCNT(2)
    3 WRITE(6,177) KZ,NAME,NR,NC
  177 FORMAT(A1,5X,A4,8H  MATRIX,5X,I3,5H ROWS,5X,I3,8H COLUMNS)
      GO TO 13
   12 CALL LNCNT(100)
      GO TO 13
  132 CALL LNCNT(2)
      WRITE (6,891)
  891 FORMAT (1H0)
C- BELOW COMPUTE NR OF LINES/ ROW
   13 J=(NC-1)/NEPR+1

      NLPW  = J
      JST =  1
C- COMPUTE LAST ROW POSITION -1
      NLST  = NLST -NR
      MN=NC
      IF  (NC.GT.NEPR)   MN=NEPR
      KLST=NR*(MN-1)
```

```
91    CONTINUE
      DO 912 J = JST, NR
      CALL LNCNT(NLPW)
      KLST = KLST +1
      WRITE (6,FMT1)    (AR(N), N = J, KLST, NR)
      IF  (NC.LE.NEPR)   GO TO 912
      NLST = NLST +1
      KNR=KLST+NR
      WRITE (6,FMT2)(AR(N),N=KNR,NLST,NR)
912   CONTINUE
      RETURN
   16 CALL LNCNT(1)
      WRITE  (6,916)   NAM,NAR
  916 FORMAT  (' ERROR IN PRNT  MATRIX 'A4,' HAS NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE LNCNT  (N)
      COMMON/LINES/NLP,LIN,TITLE(23)
      LIN=LIN+N
      IF  (LIN.LE.NLP)   GO TO 20
      WRITE  (6,1010)  (TITLE(I),I=1,23)
 1010 FORMAT  (1H1,23A4/)
      LIN=2+N
      IF  (N.GT.NLP)  LIN=2
   20 RETURN
      END
```

```
      SUBROUTINE ADD (A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      COMMON  /MAX/MAXRC
      DOUBLE PRECISION A,B,C
      IF((NA(1).NE.NB(1)).OR.(NA(2).NE.NB(2))) GO TO 999
      NC(1)=NA(1)
      NC(2)=NA(2)
      L=NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 C(I)=A(I)+B(I)
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6,50) NA,NB
   50 FORMAT  (' DIMENSION ERROR IN ADD     NA='2I6,5X,'NB='2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE SUBT(A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      DOUBLE PRECISION A,B,C
      COMMON  /MAX/MAXRC
      IF((NA(1).NE.NB(1)).OR.(NA(2).NE.NB(2))) GO TO 999
      NC(1)=NA(1)
      NC(2)=NA(2)
      L=NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 C(I)=A(I)-B(I)
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6,50) NA,NB
   50 FORMAT  (' DIMENSION ERROR IN SUBT    NA='2I6,5X,'NB='2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE MULT(A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      DOUBLE PRECISION A,B,C
      COMMON  /MAX/MAXRC
      NC(1) = NA(1)
      NC(2) = NB(2)
      IF(NA(2).NE.NB(1)) GO TO 999
      NAR = NA(1)
      NAC = NA(2)
      NBC = NB(2)
      NAA=NAR*NAC
      NBB=NAR*NBC
      IF  (NAR.LT.1.OR.NAA.LT.1.OR.NAA.GT.MAXRC.OR.NBB.LT.1.OR.
     1     NBB.GT.MAXRC)  GO TO 999
      IR = 0
      IK=-NAC
      DO 300 K=1,NBC
      IK = IK + NAC
      DO 300 J=1,NAR
      IR=IR+1
      IB=IK
      JI=J-NAR
      C(IR)=0.
      DO 300 I=1,NAC
      JI = JI + NAR
      IB=IB+1
      C(IR)=C(IR)+A(JI)*B(IB)
  300 CONTINUE
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6,500) (NA(I),I=1,2),(NB(I),I=1,2)
  500 FORMAT  (' DIMENSION ERROR IN MULT    NA='2I6,5X,'NB='2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE SCALE (A, NA, B, NB, S)
      DIMENSION A(1),B(1),NA(2),NB(2)
      COMMON  /MAX/MAXRC
      DOUBLE PRECISION A, B, S
      NB(1) = NA(1)
      NB(2) =NA(2)
      L = NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 B(I)=A(I)*S
 1000 RETURN
  999 CALL LNCNT(1)
      WRITE  (6,50) NA
   50 FORMAT  (' DIMENSION ERROR IN SCALE   NA='2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE TRANP(A,NA,B,NB)
      DIMENSION A(1),B(1),NA(2),NB(2)
      DOUBLE PRECISION A,B
      COMMON  /MAX/MAXRC
      NB(1)=NA(2)
      NB(2)=NA(1)
      NR=NA(1)
      NC=NA(2)
      L=NR*NC
      IF  (NR   .LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      IR=0
      DO 300 I=1,NR
      IJ=I-NR
      DO 300 J=1,NC
      IJ=IJ+NR
      IR=IR+1
  300 B(IR)=A(IJ)
      RETURN
  999 CALL LNCNT(1)
      WRITE  (6,50)  NA
   50 FORMAT  (' DIMENSION ERROR IN TRANP   NA='2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE INV(A,NA,DET,L)
      DIMENSION A(1), L(1),NA(2)
      DOUBLE PRECISION A, DET, BIGA ,  HOLD
      COMMON  /MAX/MAXRC
      IF  (NA(1).NE.NA(2))  GO TO 600
C     SEARCH FOR LARGEST ELEMENT
      DET= 1.
      N=NA(1)
      NSQ=N*N
      IF  (N.LT.1.OR.NSQ.GT.MAXRC)   GO TO 600
      NK = - N
      DO 80 K= 1, N
      NK = NK + N
      L(K) = K
      NPK=N+K
      L(NPK)=K
      KK = NK + K
      BIGA = A(KK)
      DO 20 J= K, N
      IZ = N*(J - 1)
      DO   20 I= K, N
      IJ = IZ + I
   10 IF(DABS(BIGA) - DABS(A(IJ))) 15, 20, 20
   15 BIGA = A(IJ)
      L(K) = I
      NPK=N+K
      L(NPK)=J
   20 CONTINUE
C     INTERCHANGE ROWS
      J = L(K)
      IF(J - K) 35, 35, 25
   25 KI = K - N
      DO 30 I = 1, N
      KI = KI + N
      HOLD = -A(KI)
      JI = KI - K + J
      A(KI) = A(JI)
   30 A(JI) = HOLD
```

```
C        INTERCHANGE COLUMNS
   35 NPK=N+K
      I=L(NPK)
      IF (I - K) 45, 45, 38
   38 JP = N*(I - 1)
      DO 40 J= 1, N
      JK = NK + J
      JI = JP + J
      HOLD = -A(JK)
      A(JK) = A(JI)
   40 A(JI) = HOLD
C        DIVIDE COLUMN BY MINUS PIVOT(VALUE OF PIVOT ELEMENTS IS CONTAINED
C        IN BIGA)
   45 IF(BIGA) 48, 46, 48
   46 DET = 0.
      CALL LNCNT (1)
      KKK=KK-1
      WRITE  (6,1046)  KKK
 1046 FORMAT  (' INV ERROR  DETERMINANT OF A=0   RANK OF A=',I4)
      CALL ASPERR
      RETURN
   48 DO 55 I= 1, N
      IF (I - K) 50, 55, 50
   50 IK = NK + I
      A(IK) =-A(IK)/(BIGA)
   55 CONTINUE
C        REDUCE MATRIX
      DO 65 I= 1, N
      IK = NK + I
      HOLD = A(IK)
      IJ = I - N
      DO 65 J= 1, N
      IJ = IJ + N
      IF(I  - K ) 60, 65, 60
   60 IF(J- K) 62, 65, 62
   62 KJ = IJ -I + K
      A(IJ) = HOLD* A(KJ) + A(IJ)
   65 CONTINUE
C        DIVIDE ROW BY PIVOT
```

```
      KJ = K - N
      DO 75 J= 1, N
      KJ = KJ + N
      IF(J - K) 70, 75, 70
   70 A(KJ) = A(KJ)/BIGA
   75 CONTINUE
C     PRODUCT OF PIVOTS
      DET=DET*BIGA
C     REPLACE PIVOT BY RECIPROCAL
      A(KK) = 1./BIGA
   80 CONTINUE
C     FINAL ROW AND COLUMN INTERCHANGE
      K = N
  100 K = K - 1
      IF(K) 150, 150, 105
  105 I = L(K)
      IF (I - K ) 120, 120, 108
  108 JQ = N*( K - 1)
      JR = N*(I- 1)
      DO 110 J= 1, N
      JK = JQ + J
      HOLD = A(JK)
      JI = JR + J
      A(JK) = - A(JI)
  110 A(JI) = HOLD
  120 NPK=N+K
      J=L(NPK)
      IF(J - K) 100, 100, 125
  125 KI = K - N
      DO 130 I= 1, N
      KI = KI + N
      HOLD = A(KI)
      JI = KI - K + J
      A(KI) = - A(JI)
  130 A(JI) = HOLD
      GO TO 100
```

```
 150 RETURN
 600 CALL LNCNT (1)
     WRITE  (6,1600)  NA
1600 FORMAT  (' INV REQUIRES SQUARE MATRIX   NA=',2I4)
     CALL ASPERR
     RETURN
     END
```

```
      SUBROUTINE NORM(A,NA,ANORM)
      DIMENSION NA(2),A(1)
      DOUBLE PRECISION A,ANORM,SUM,ROWMAX,COLMAX
      COMMON  /MAX/MAXRC
      NAR = NA(1)
      NAC = NA(2)
      L=NAR*NAC
      IF  (NAR  .LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      COLMAX = 0.
      ROWMAX = 0.
      K = 0
      DO 300 I = 1,NAC
      SUM = 0.
      DO 301 J = 1,NAR
      K = K + 1
  301 SUM = SUM + DABS(A(K))
      IF (COLMAX.LT.SUM) COLMAX = SUM
  300 CONTINUE
      DO 302 I = 1,NAR
      SUM = 0.
      K = I - NAR
      DO 303 J = 1,NAC
      K = K + NAR
  303 SUM = SUM + DABS(A(K))
      IF (ROWMAX.LT.SUM) ROWMAX=SUM
  302 CONTINUE
      ANORM = DMIN1(COLMAX,ROWMAX)
      RETURN
  999 CALL LNCNT (1)
      WRITE  (6,50)  NA
   50 FORMAT  (' DIMENSION ERROR IN NORM     NA='2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE UNITY(A,NA)
      DIMENSION A(1),NA(2)
      DOUBLE PRECISION A
      IF(NA(1).NE.NA(2)) GO TO 999
      CALL SCALF(A,NA,A,NA,0.D0)
      J = - NA(1)
      NAX = NA(1)
      DO 300 I=1,NAX
      J=NAX +J+1
  300 A(J)=1.
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6, 50)(NA(I),I=1,2)
   50 FORMAT  (' DIMENSION ERROR IN UNITY    NA='2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE TRCE  (A,NA,TR)
      DOUBLE PRECISION A(1),TR
      DIMENSION NA(2)
      COMMON  /MAX/MAXRC
      IF (NA(1).NE.NA(2)) GO TO 600
      TR=0.D0
      N=NA(1)
      NN=N*N
      IF  (N.LT.1.OR.NN.GT.MAXRC)   GO TO 600
      DO 10 I=1,N
      M=I+N*(I-1)
   10 TR=TR+A(M)
      RETURN
  600 CALL LNCNT(1)
      WRITE (6,1600) NA
 1600 FORMAT (' TRACE REQUIRES SQUARE MATRIX    NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE EQUATE(A,NA,B,NB)
      DIMENSION A(1),B(1),NA(2),NB(2)
      DOUBLE PRECISION A, B
      COMMON  /MAX/MAXRC
      NB(1) = NA(1)
      NB(2) =NA(2)
      L=NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 B(I)=A(I)
 1000 RETURN
  999 CALL LNCNT (1)
      WRITE  (6,50)  NA
   50 FORMAT  (' DIMENSION ERROR IN EQUATE  NA='2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE JUXTC(A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      DOUBLE PRECISION A,B,C
      COMMON  /MAX/MAXRC
      IF (NA(1).NE.NB(1)) GO TO 600
      NC(1)=NA(1)
      NC(2)=NA(2)+NB(2)
      L=NA(1)*NA(2)
      NNC=NC(1)*NC(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 600
      IF  (NC(2).LT.1.OR.NNC.GT.MAXRC)   GO TO 600
      MS=NA(1)*NA(2)
      DO 10 I=1,MS
   10 C(I)=A(I)
      MBS=NA(1)*NB(2)
      DO 20 I=1,MBS
      J=MS+I
   20 C(J)=B(I)
      RETURN
  600 CALL LNCNT(1)
      WRITE (6,1600) NA,NB
 1600 FORMAT (' DIMENSION ERROR IN JUXTC,  NA=',2I6,5X,'NB=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE JUXTR(A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      DOUBLE PRECISION A,B,C
      COMMON  /MAX/ MAXRC
      IF(NA(2).NE.NB(2))GO TO 600
      NC(2)=NA(2)
      NC(1)=NA(1)+NB(1)
      L=NA(1)*NA(2)
      NNC=NC(1)*NC(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 600
      IF  (NC(2).LT.1.OR.NNC.GT.MAXRC)   GO TO 600
      MCA=NA(2)
      MRA=NA(1)
      MRB=NB(1)
      MRC=NC(1)
      DO 10 I=1,MCA
      DO 10 J=1,MRA
      K=J+MRA*(I-1)
      L=J+MRC*(I-1)
   10 C(L)=A(K)
      DO 20 I=1,MCA
      DO 20 J=1,MRB
      K=J+MRB*(I-1)
      L=MRA+J+MRC*(I-1)
   20 C(L)=B(K)
      RETURN
  600 CALL LNCNT(1)
      WRITE(6,1600) NA,NB
 1600 FORMAT(' DIMENSION ERROR IN JUXTR,  NA=',2I6,5X,'NB=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE EAT(A, NA, TT, B, NB, C, NC, DUMMY, KDUM)
      DIMENSION A(1),B(1),DUMMY(1),NA(2),NB(2),ND(2),C(1),NC(2)
      DOUBLE PRECISION A, T, TT, ANAA, TMAX, B, DUMMY, C , S, SC
      COMMON  /MAX/MAXRC
      NR=NA(1)
      NCC=NA(2)
      NC(1)=NR
      NC(2)=NCC
      NB(1)=NR
      NB(2)=NCC
      LD=NR*NCC
      IF (NR.NE.NCC.OR.NR.LT.1          .OR.LD.GT.MAXRC) GO TO 998
      NDD=2*NA(1)*NA(1)
      IF(KDUM  .LT.NDD) GO TO 998
      NDD= NA(1)*NA(1)+1
      T=TT
      CALL NORM(A,NA,ANAA)
      TMAX= 10.01/ANAA
      K=0
  101 IF (TMAX-T ) 103,104,104
  103 K=K+1
      T=TT/2**K
      IF (K-1000) 101,102,102
  104 SC=T
      CALL SCALE(A,NA,A,NA,T)
      DO 401 I= 1, 2
  401 NB(I) = NA(I)
      CALL UNITY(B,NB)
      CALL SCALE(B,NB,DUMMY(1),NB,T)
      S = T/2.
      CALL SCALE(A,NA,DUMMY(NDD),NA,S)
      N = 35
      II=2
      CALL ADD (DUMMY(1),NA,DUMMY(NDD),NA,DUMMY(NDD),NA)
      CALL ADD(A,NA,B,NB,DUMMY(1),NA)
      CALL EQUATE(A,NA,C,NC)
  106 CALL MULT(A,NA,C,NC,B,NB)
```

```
      S=1.D0/II
      CALL SCALE(B,NB,C,NC,S)
      S = T/(II+1)
      CALL SCALF(C,NC,B,NB,S)
      CALL ADD(B,NB,DUMMY(NDD),NB,DUMMY(NDD),NB)
      CALL ADD(C,NC,DUMMY(1),NC,DUMMY,NC)
      N=N-1
      IF (N) 107,107,105
 105  II=II+1
      GO TO 106
 107  CALL EQUATE(DUMMY(1), NB, B, NB)
      IF (K) 109,108,212
 109  CALL LNCNT (1)
      WRITE (6,110)
 110  FORMAT (' ERROR IN EAT     K IS NEGATIVE')
 112  IF (K-1) 213,212,212
 213  K=1
 212  DO 111 J=1,K
      T=2*T
      CALL EQUATE(B, NB, DUMMY(1), NB)
      CALL MULT(DUMMY(1), NA, DUMMY(NDD), NA,C,NC)
      CALL ADD(DUMMY(NDD), NC, C, NC, DUMMY(NDD), NC )
 111  CALL MULT(DUMMY(1), NB, DUMMY(1), NB, B, NB)
      TT=T
 108  CONTINUE
      CALL EQUATE(DUMMY(NDD),NC,C,NC)
      S=1.D0/SC
      CALL SCALE(A,NA,A,NA,S)
      RETURN
 102  CALL LNCNT (1)
      WRITE (6,119)
 119  FORMAT  (' ERROR IN EAT     K=1000')
      CALL ASPERR
      RETURN
 998  CALL LNCNT (1)
      WRITE(6, 50)  NA,KDUM,NDD
  50  FORMAT  (' DIMENSION ERROR IN EAT      NA='2I6,    'KDUM='I5,5X,
     1    'KDUM(MIN)='I5)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE ETPHI(A,NA,TT,B,NB,DUMMY,KDUM)
      DIMENSION A(1),B(1),DUMMY(1),NA(2),NB(2),ND(2)
      DOUBLE PRECISION A, T, TT, ANAA, TMAX, B, DUMMY ,S, SC
      COMMON  /MAX/MAXRC
      NR=NA(1)
      NCC=NA(2)
      NB(1)=NR
      NB(2)=NCC
      LD=NR*NCC
      IF (NR.NE.NCC.OR.NR.LT.1           .OR.LD.GT.MAXRC) GO TO 998
      NDD=2*NA(1)*NA(1)
      IF(KDUM  .LT.NDD) GO TO 998
      NDD= NA(1)*NA(1)+1
      T=TT
      CALL NORM(A,NA,ANAA)
      TMAX= 10.01/ANAA
      K=0
  101 IF (TMAX-T ) 103,104,104
  103 K=K+1
      T=TT/2**K
      IF (K-1000) 101,102,102
  104 SC=T
      CALL SCALE(A,NA,A,NA,T)
      CALL UNITY(B,NB)
      II=2
      N = 35
      CALL ADD(A,NA,B,NB,DUMMY(1),ND)
      CALL EQUATE(A,NA,DUMMY(NDD),ND)
  106 CALL MULT(A,NA,DUMMY(NDD),ND,B,NB)
      S=1.D0/II
      CALL SCALE(B,NB,DUMMY(NDD),ND,S)
      CALL ADD(DUMMY(NDD),ND,DUMMY(1),ND,B,NB)
      CALL EQUATE(B,NB,DUMMY(1),ND)
      N=N-1
      IF (N) 107,107,105
  105 II=II+1
      GO TO 106
```

```
107 IF (K) 109,108,212
109 CALL LNCNT (1)
    WRITE (6,110)
110 FORMAT (' ERROR IN ETPHI  K IS NEGATIVE')
112 IF (K-1) 213,212,212
213 K=1
212 DO 111 J=1,K
    T=2*T
    CALL EQUATE(B, NB, DUMMY(1), ND)
    CALL EQUATE(DUMMY(1), ND, DUMMY(NDD), ND)
111 CALL MULT(DUMMY(NDD),ND,DUMMY(1),ND,B,NB)
    TT=T
108 CONTINUE
    S=1.D0/SC
    CALL SCALE(A,NA,A,NA,S)
    RETURN
102 CALL LNCNT (1)
    WRITE (6,119)
119 FORMAT   (' ERROR IN ETPHI   K=1000')
    CALL ASPERR
    RETURN
998 CALL LNCNT (1)
    WRITE  (6,50)  NA,KDUM,NDD
 50 FORMAT  (' DIMENSION ERROR IN ETPHI   NA='2I6,   'KDUM='I5,5X,
   1    'KDUM(MIN)='I5)
    CALL ASPERR
    RETURN
    END
```

```
      SUBROUTINE AUG(F,NF,G,NG,RI,NRI,H, NH,Q,NQ, C,NC,Z,NZ, II)
      DIMENSION F(1),G(1),RI(1),H(1),Q(1),Z(1),C(1)
      DIMENSION NNP1(2),NNP2(2),NNP3(2),NNP4(2),NF(2),NG(2),NRI(2),
     1NH(2),NZ(2),NC(2),NN(2),NQ(2)
      DOUBLE PRECISION F, G, RI,H,Q,C,Z
      IF((NF(1).NE.NF(2)).OR.(NRI(1).NE.NRI(2)).OR.(
     1NQ(1).NE.NQ(2))) GO TO 995
      NNZ=2*NF(1)
      IF( (NG(1).NE.NF(1)).OR.(NG(2).NE.NRI(1)))GO TO 995
      IF(II.EQ.1) GO TO 206
      IF((NH(1).NE.NQ(1)).OR.(NH(2).NE.NF(2))) GO TO 995
  206 TWO = 2
      N = NF(1)
      NSQ = N*N
      NZ(1)=NNZ
      NZ(2)=NNZ
      NP1=1
      NP2 = NP1 + NSQ
      NP3 = NP2+NSQ
      NP4 = NP3 + NSQ
      CALL TRANP(G,NG,Z(NP4),NNP4)
      CALL MULT(RI,NRI,Z(NP4),NNP4,C,NC)
      CALL MULT(G,NG,C,NC,Z(NP4), NNP4)
      IF(II .EQ. 1) GO TO 204
      CALL TRANP(H,NH,Z(NP3), NNP3)
      CALL MULT(Q,NQ,H,NH,Z(1), NNP1)
      CALL MULT(Z(NP3),NNP3,Z(   1),NNP1,Z(NP2),NNP2)
      GO TO 205
  204 CALL EQUATE(Q, NQ, Z(NP2), NQ)
  205 NPAIR=MOD(N,2)
      IF(NPAIR.EQ.0) NPAIR=TWO
      NN(1) = N
      NN(2) = 1
      GO TO (201,202),NPAIR
  201 DO 300 I=1,N,2
      NP2 = N*(N+I-1)+1
      NTH3=TWO*N*(I-1)+N+1
```

```
 300 CALL EQUATE(Z(NP2),NN,Z(NTH3),NN)
     DO 302 I=2,N,2
     NP4=N*(3*N+I-1)+1
     NTH2=TWO*N*(N+I-1)+1
 302 CALL EQUATF(Z(NP4),NN,Z(NTH2),NN)
     GO TO (202,203),NPAIR
 202 DO 301 I=2  ,N,2
     NP2 = N*(N+I-1)+1
     NTH3=TWO*N*(I-1)+N+1
 301 CALL EQUATF(Z(NP2),NN,Z(NTH3),NN)
     DO 304 I=1,N,2
     NP4=N*(3*N+I-1)+1
     NTH2=TWO*N*(N+I-1)+1
 304 CALL EQUATF(Z(NP4),NN,Z(NTH2),NN)
     GO TO (203,201),NPAIR
 203 DO 303 I=1,N
     IJ = I+N
     DO 303 J=1,N
     JJ = J+N
     L1=NNZ*(J-1)+I
     L2=NNZ*(IJ-1)+JJ
     L3=N*(J-1)+I
     Z(L1)=-F(L3)
 303 Z(L2)=F(L3)
     GO TO 1000
 995 CALL LNCNT (2)
     WRITE  (6,50)  NF,NG,NRI,NH,NO
  50 FORMAT  (' DIMENSION ERROR IN AUG',T35,'NF',9X,'NG',9X,'NRI',8X,
    1    'NH',9X,'NO'/29X,5(3X,2I6))
 999 CALL ASPERR
1000 RETURN
     END
```

```
      SUBROUTINE RICAT(PHI,NPHI,C,NC,NCONT,K,NK,PT,NPT,W,KDUM)
      DIMENSION NCONT(3),NPHI(2),NC(2),NK(2),NN(2),NM(2),NPT(2)
      DIMENSION PHI(1),C(1),K(1),PT(1),W(1)
      DOUBLE PRECISION PHI, C, K, PT, SUM, SUMN, AL, W,DET
      TWO=2
      N = NPHI(1)/TWO
      NSQ=N*N
      NSQ4=4*NSQ
      NP1=1
      NP2= NSQ+NP1
      NP3=NSQ+NP2
      NP4= NSQ+NP3
      IF  (KDUM.LT.NSQ4 )  GO TO 600
      IF  (NPHI(2).NE.NPHI(1).OR.NC(2).NE.N.OR.NPT(1).NE.N.OR.NPT(2).
     1  NE.N)   GO TO 600
      NPRINT=NCONT(1)
      NBLOCK=NCONT(2)/NPRINT
      NZ=NCONT(3)
C     REARRANGE PHI MATRIX
      NN(1)=N
      NN(2)=1
      DO 300 I=1,N
      NTH1 =TWO*N*(I-1)+1
      NTH3=NTH1+N
      NW1=N*(I-1)+1
      NW2=NW1+N*N
      CALL EQUATE(PHI(NTH1),NN,W(NW1),NN)
  300 CALL EQUATE(PHI(NTH3),NN,W(NW2),NN)
      NM(1)=TWO*N*N
      NM(2)=1
      CALL EQUATE(W(1),NM,PHI(1),NM)
      DO 301 I=1,N
      NTH2=TWO*N*(N+I-1)+1
      NTH4=NTH2+N
      NW3 = N*(TWO*N+I-1)+1
      NW4= NW3+N*N
      CALL EQUATE(PHI(NTH2),NN,W(NW3),NN)
```

```
  301 CALL EQUATE(PHI(NTH4),NN,W(NW4),NN)
      NWW=TWO*N*N+1
      CALL EQUATE(W(NWW),NM,PHI(NWW),NM)
C     MAIN COMPUTATION
C     CALL UNITY(PT,NPT)
      DO 306 I= 1,N
  306 K(I) = 0.
      NTOT=0
      DO 403 I=1,NBLOCK
      DO 104 J=1,NPRINT
      CALL MULT(PHI(NP3), NPT, PT, NPT, W(1), NPT)
      CALL ADD (PHI(1), NPT, W(1), NPT, W(1), NPT)
      CALL INV(W(1), NPT, DET, W(NP2))
      CALL MULT(PHI(NP4), NPT, PT, NPT, W(NP2), NPT)
      CALL ADD(PHI(NP2), NPT, W(NP2), NPT, W(NP2), NPT)
      CALL MULT(W(NP2), NPT, W(1), NPT, PT, NPT)
      SUMN=0.
      SUM=0.
      DO 202 IL=1,N
      ILP=IL+NP3
      NIL=(IL-1)*N+IL
      SUM=SUM+DABS(PT(NIL))
  202 SUMN=SUMN+DABS(PT(NIL)        -W(ILP))
      AL=SUMN/SUM
      DO 201 IL=1,N
      NIL=(IL-1)*N+IL
      ILP=IL+NP3
  201 W(ILP)    =PT(NIL)
  204 DO 104 M=2,N
      N1=M-1
      DO 104 L=1,N1
      L1=N*(L-1)+M
      L2=N*(M-1)+L
      PT(L1)=(PT(L1) + PT(L2))/2.
      PT(L2)=PT(L1)
      IF(AL-.00001) 203,203,104
  104 CONTINUE
      NTOT=I*NPRINT
      GO TO 305
```

```
  203 NTOT=NTOT+J
  305 CALL  MULT (C,NC,PT,NPT,K,NK)
  103 GO TO (403,400,401,402), NZ
  400 CALL LNCNT (1)
      WRITE (6, 50) NTOT
   50 FORMAT (10X,I4,14H   ITERATIONS
      CALL PRNT  (PT,NPT,'P(T)',1)
      GO TO 403
  401 CALL LNCNT (1)
      WRITE (6, 50) NTOT
      CALL PRNT   (K,NK,'K(T)',1)
      GO TO 403
  402 CALL LNCNT (1)
      WRITE (6, 50) NTOT
      CALL PRNT   (K,NK,'K(T)',1)
      CALL PRNT  (PT,NPT,'P(T)',1)
      IF(AL-.00001) 210,210,403
  403 CONTINUE
C     REARRANGE PHI MATRIX
  210 CALL EQUATE(PHI(1),NM,W(1),NM)
      DO 303 I=1,N
      NTH1 =TWO*N*(I-1)+1
      NTH3=NTH1+N
      NW1=N*(I-1)+1
      NW2=NW1+N*N
      CALL EQUATE(W(NW1),NN,PHI(NTH1),NN)
  303 CALL EQUATE(W(NW2),NN,PHI(NTH3),NN)
      CALL EQUATE(PHI(NWW),NM,W(NWW),NM)
      DO 304 I=1,N
      NTH2=TWO*N*(N+I-1)+1
      NTH4=NTH2+N
      NW3 = N*(TWO*N+I-1)+1
      NW4= NW3+N*N
      CALL EQUATE(W(NW3),NN,PHI(NTH2),NN)
  304 CALL EQUATE(W(NW4),NN,PHI(NTH4),NN)
      RETURN
```

```
 600 CALL LNCNT (2)
     WRITE  (6,1600)  NPHI,NC,NPT,KDUM,NSQ4
1600 FORMAT  (' DIMENSION ERROR IN RICAT',T35,'NPHI',7X,'NC',9X,'NPT'
    1  ,6X,'KDUM',3X,'KDUM(MIN)'/29X,3(3X,2I4),4X,I4,5X,I4)
     CALL ASPERR
     RETURN
     END
```

```
      SUBROUTINE SAMPL(PHI,NPHI,H,NH,Q,NQ,R,NR,P,NP,K,NK,NCONT,DUM,KDUM)
      DIMENSION  NPHI(2),NH(2),NQ(2),NR(2),NP(2),NK(2),NCONT(3),NZD(12)
      DOUBLE PRECISION PHI(1),H(1),Q(1),R(1),P(1),K(1),DUM(1)
      DOUBLE PRECISION SUM,SUMN,AL
C     DIMENSION OF DUM MUST BE AT LEAST 6*N*N
C     CHECK FOR CONFORMABLE MATRICES
      N=NPHI(1)
      M=NH(1)
      NK(1)=N
      NK(2)=M
      NSQ=N*N
      ND1=1
      ND2=NSQ+1
      ND6=5*NSQ+1
      ND3=ND2+NSQ
      NSQ6=6*NSQ
      IF (NPHI(2).NE.N.OR.NH(2).NE.N.OR.NQ(1).NE.N.OR.NQ(2).NE.N.OR.NR(1
     1).NE.M.OR.NR(2).NE.M.OR.NP(1).NE.N.OR.NP(2).NE.N.OR.KDUM.LT.NSQ6)
     2GO TO 900
      NFIN=NCONT(2)
      NPRINT=NCONT(1)
      NZ=NCONT(3)
C     START OF MAIN COMPUTATION,  P(0) IS INPUT DATA IN P MATRIX
      KLN=0
      JFN=0
      I=0
      AL=1.
  100 CALL MULT(H,NH,P,NP,DUM,NZD)
C     DUM=H*P
      CALL TRANP (H,NH,DUM(ND2),NZD(3))
C     DUM2=HPRIME
      CALL MULT  (DUM,NZD,DUM(ND2),NZD(3),DUM(ND3),NZD(5))
C     DUM3=H*P*HPRIME
      CALL ADD   (DUM(ND3),NZD(5),R,NR,DUM,NZD)
C     DUM=H*P*HPRIME+R
      CALL PSEUDO (DUM,NZD,DUM(ND2),NZD(3),DUM(ND3),KDUM-3*NSQ)
C     DUM2=INVERSE
      CALL TRANP (H,NH,DUM,NZD)
C     DUM=HPRIME
      CALL MULT  (DUM,NZD,DUM(ND2),NZD(3),DUM(ND3),NZD(5))
```

```
      CALL MULT  (P,NP,DUM(ND3),NZD(5),DUM,NZD)
C     DUM=A
  110 CALL MULT  (PHI,NPHI,DUM,NZD,K,NK)
      CALL MULT  (DUM,NZD,H,NH,DUM(ND2),NZD(3))
C     DUM2=A*H
      CALL MULT  (DUM(ND2),NZD(3),P,NP,DUM,NZD)
C     DUM=A*H*P
      CALL SUBT  (P,NP,DUM,NZD,DUM,NZD)
C     DUM= P-A*H*P
      CALL TRANP (PHI,NPHI,DUM(ND2),NZD(3))
      CALL MULT  (DUM,NZD,DUM(ND2),NZD(3),DUM(ND3),NZD(5))
      CALL MULT  (PHI,NPHI,DUM(ND3),NZD(5),DUM,NZD)
C     DUM=PHI(P-A*H*P)PHIPRIME
      CALL ADD   (DUM,NZD,Q,NQ,P,NP)
      DO 120 M=2,N
      N1=M-1
      DO 120 L=1,N1
      L1=N*(L-1)+M
      L2=N*(M-1)+L
      P (L1)=(P (L1)+P (L2))/2.
  120 P (L2)=P (L1)
  130 IF (I.EQ.0)  GO TO 150
      SUM=0.
      SUMN=0.
      DO 140 IL=1,N
      IJ=(IL-1)*N+IL
      SUM=SUM+DABS(P(IJ))
      NDL=ND6+IL
  140 SUMN=SUMN+DABS(P(IJ)-DUM(NDL))
      AL=SUMN/SUM
  150 DO 151 IL=1,N
      IJ=(IL-1)*N+IL
      NDL=ND6+IL
  151 DUM(NDL)    =P(IJ)
      ILST=I
      I=I+1
      IF (AL.LE..00001)  GO TO 300
      IF  (I.GE.NFIN)   GO TO 310
C     INTERMEDIATE PRINT
```

```
      IF (I.LT.NPRINT  ) GO TO  100
      NPRINT=NPRINT+NCONT(1)
  152 GO TO  (170,156,155,155),NZ
  155 CALL LNCNT(2)
      WRITE  (6,1152) ILST
 1152 FORMAT  ('0STEP NUMBER=',I4,' IN SAMPL')
      CALL PRNT  (K,NK,4HK(T),1)
      GO TO  (170,170,170,156),NZ
  156 CALL LNCNT(2)
      WRITE  (6,1152) I
  160 CALL PRNT  (P,NP,4HP(T),1)
  170 IF (JFN.EQ.0)  GO TO 100
      RETURN
  300 CALL LNCNT(2)
      WRITE   (6,1300)
 1300 FORMAT   ('0P NO LONGER CHANGING,   EXIT FROM SAMPL')
  310 JFN=1
      GO TO 152
  900 CALL LNCNT (2)
      WRITE (6,1000) NPHI,NH,NQ,NR,NP,KDUM,NS06
 1000 FORMAT  ('DIMENSION ERROR IN SAMPL',T35     ,3X,4HNPHI,8X,2HNH,
     19X,2HNQ,9X,2HNR,9X,2HNP,5X,4HKDUM,3X,9HKDUM(MIN)/29X,5(3X,2I4),
     23X,I4,5X,I4)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE TRNSI(F,NF,G,NG,J,NJ,R,NR,K,NK,H,NH,X,NX,T,DUMMY,KDUM)
      DOUBLE PRECISION F,G,J,K,H,X,T,Y,R,DUMMY
      DIMENSION F(1),NF(1),G(1),NG(1),J(1),NJ(1),K(1),NK(1),H(1),NH(1),
     1X(1),NX(1),T(1),R(1),NR(1),DUMMY(1)
      DIMENSION NNF(2),NNG(2),NU(2),NNX(2),NY(2)
      DATA STAR/'*'/
      IF(NF(2).NE.NG(1) .OR. NJ(2).NE.NR(1) .OR. NK(2).NE.NX(1) .OR.
     1NJ(1).NE.NK (1) .OR. NR(2).NE.NX(2) .OR.NH(2).NE.NX(1) .OR.
     2NF(2) .NE.NX(1)) GO TO 999
      MAX = NF(1)*(NF(2) +NG(2)+ 1) +NH(1)+NK(1)
      IF (KDUM .LT. MAX) GO TO 910
      I1 = 1
      NSQ =NF(I1) *NF(I1)
      NX4 = NSQ   *4
C
      IF(KDUM .LT. NX4) GO TO 900
C    TRNSI PROGRAM
      N2 = I1 + NSQ
      N3 = N2 + NSQ
      N4 = N3 + NSQ
      L3 = N2 +NF(I1)*NG(2)
      L4 = L3 + NJ(I1)*NR(2)
      L5 = L4 + NH(1)
      L6 = L5 + NJ(I1)*NR(2)
      T1 =T(1)
      N = (T(2) + .5*T1)/T1
      NXR =NX(I1)
      LAST = L6 -I1
      TT = T(4)
      CALL PRNT (F,NF,'  F ',1)
      CALL EAT(F,NF,T1,DUMMY(N3),NNF,DUMMY(N4),NNF,DUMMY(I1),KDUM)
  100 FORMAT(1H0 1P8E16.7)
      CALL  PRNT(DUMMY(N3), NF, 'EAT ', 1)
      CALL EQUATE(DUMMY(N3),NF,DUMMY(I1),NNF)
      CALL  PRNT(DUMMY(N4), NF, 'INT ', 1)
      CALL MULT(DUMMY(N4), NF,G,NG,DUMMY(N2),NNG)
      CALL MULT(J,NJ,R,NR,DUMMY(L3),NU)
      CALL LNCNT (100)
      CALL LNCNT (3)
```

```
      WRITE(6, 50)
   50 FORMAT(1HO 'TRANSIENT RESPONSE, * INDICATES CONTROL CHANGES')
      WRITE(6, 51) NXR,    NH(I1),NK(I1)
   51 FORMAT(1HO 4X,'TIME    FIRST',I3,'  ELEMENTS CONTAIN X, NEXT',I3,'
     1  ELEMENTS CONTAIN Y = HX, LAST',I3,'  ELEMENTS CONTAIN U =JR -KX'
     2)
  201 NQ=1
      IU = I1
      CALL MULT(K,NK, X,NX, DUMMY (L5), NU)
      CALL SUBT(DUMMY(L3), NU,DUMMY(L5), NU, DUMMY(L5),NU)
  203 CALL MULT(H,NH,X,NX,DUMMY(L4 ),NY)
      IF (IU .NE. I1) GO TO 205
      WRITE(6, 101) STAR,TT,(X(IP), IP=1,NXR),(DUMMY(IP),IP=L4,LAST)
  101 FORMAT(1HO A1,F8.2,1P7D15.7/(10X,1P7D15.7))
      GO TO 206
  205 WRITE(6, 102)       TT,(X(IP), IP=1,NXR),(DUMMY(IP),IP=L4,LAST)
  102 FORMAT(1HO 1X,F8.2,1P7D15.7/(10X,1P7D15.7))
  206 IU = I1  +IU
      IF (TT .GE. DABS( T(3)))  RETURN
      TT = TT + T1
  202 CALL MULT(DUMMY(I1), NNF, X, NX, DUMMY(L6 ),NNX)
      CALL MULT(DUMMY(N2),NG, DUMMY(L5),  NU, X, NNX)
      CALL ADD(X,NX,DUMMY(L6 ),NNX, X,NNX)
      IF(NQ .GE. N) GO TO 201
      NQ = NQ + 1
      GO TO 203
C
C    DIMENSION ERROR DIAGNOSTIC
  999 WRITE(6, 990)
  990 FORMAT(1HO 'DIMENSION ERROR IN TRNSI'/25X,'COL SIZE OF 1ST MATRIX
     1      ROW SIZE OF 2ND MATRIX')
      WRITE(6,991) NF(2), NG(1)
      WRITE(6,992) NJ(2),NR(1)
      WRITE(6,993) NK(2),NX(1)
      WRITE(6,995) NH(2),NX(1)
      WRITE(6,996) NF(2), NX(1)
      WRITE(6,994) NJ(1),NK(1),NR(2),NX(2)
```

```
  991 FORMAT(1HO 'INTEGRAL (EXP(F.T)) G ' I13,20X, I8)
  992 FORMAT(1HO 'J R' 17X,I15,20X,I8)
  993 FORMAT(1HO 'K X' 17X,I15,20X,I8)
  994 FORMAT(1HO 'JR -KX      ROW SIZE OF J IS'I5,3X,'OF K IS',I5,3X,'COL
     1 SIZE OF R IS' I5,3X, 'OF X IS' I5)
  995 FORMAT(1HO 'H X' 17X,I15,20X,I8)
  996 FORMAT(1HO 'EXP(F.T) X' 10X,I15,20X,I8)
      GO TO 1000
  900 WRITE(6, 52) NX4,KDUM
   52 FORMAT(1HO 'DUMMY MUST BE DIMENSIONED AT LEAST' I4, ' X 1' 'BUT IS
     1 DIMENSIONED ONLY' I4,' X 1')
      GO TO 1000
  910 WRITE(6, 52) MAX,KDUM
 1000  CALL  ASPERR
      RETURN
      END
```

```
      SUBROUTINE PSEUDO(A,NA,B,NB,DUM,KDUM)
      DIMENSION A(1),B(1),DUM(1),NA(2),NB(2),IP(4),DEP(3)
      DOUBLE PRECISION A,B,DUM,DEP
      NNW=3*NA(1)*NA(2)
      DO 10 I=1,2
      DEP(I)=0.0
   10 IP(I)=0
   20 IP(3)=NA(1)
      IP(4)=NA(2)
      NB(1)=NA(2)
      NB(2)=NA(1)
      IF  (KDUM.LT.NNW)  GO TO 999
      NEE=NA(1)*NA(2) + 1
      ND=2*NEE - 1
C   IF A IS (1,1) MATRIX ROUTINE INVERTS A(1) AND PUTS IT IN B(1)
      IF  (NA(1).EQ.1.AND.NA(2).EQ.1)   GO TO 600
      CALL PSEU(A,B,DUM,DUM(NEE),DEP,IP,DUM(ND))
      GO TO 1000
  600 B(1)=1./A(1)
      GO TO 1000
  999 WRITE(6,500) KDUM,NNW
  500 FORMAT  (' DIMENSION ERROR IN PSEUDO   KDUM='I5,3X,'KDUM(MIN)='I5)
 1000 RETURN
      END
```

```
C   SUBROUTINE DECGEN       DECOMPOSE A REAL     GENERAL MATRIX
      SUBROUTINE DECGEN (R,NR,G,NG,H,NH,DUM,KDUM)
      DIMENSION  NR(2),NH(2),NG(2),ND(2),NA(2),KP(4)
      COMMON   /MAX/MAXRC
      DOUBLE PRECISION  R(1),G(1),H(1),DUM(1),DCM(3)
      NJ=NR(1)*NR(2)+1
      CALL TRANP (R,NR,DUM(NJ),ND)
      IF(NR(1).GT.NR(2))GO TO 10
      CALL MULT (R,NR,DUM(NJ),ND,DUM,NA)
      ICS=1
      GO TO 30
   10 CALL MULT (DUM(NJ),ND,R,NR,DUM,NA)
      ICS=2
      GO TO 30
C   ENTRY DECSYM           DECOMPOSE A REAL SYMETRIC MATRIX
      ENTRY DECSYM (R,NR,G,NG,H,NH,DUM,KDUM)
      ICS=0
      IF   (NR(1).NE.NR(2))   GO TO 601
      CALL EQUATE (R,NR,DUM,NA)
C          DUMMY STORAGE  MUST BE AT LEAST 7*NA(1)**2
   30 M=NA(1)
      MM7=7*M*M
      IF   (KDUM.LT.MM7   )   GO TO 601
      KP(1)=0
      KP(2)=0
      KP(3)=M
      KP(4)=M
      DCM(1)=0.
      DCM(2)=0.
      MS=M*M
      N2=MS+1
      N3=N2+MS
      N4=N3+MS
      N5=N4+MS
      N6=N5+MS
      N7=N6+MS
      CALL DECOM (DUM,DUM(N7),DUM(N4),DUM(N5),DUM(N6),DCM,KP,DUM(N2))
```

```
      CALL TRANP  (DUM(N3),NA,DUM(N6),ND)
      IF (ICS.EQ.2)  GO TO  100
      CALL MULT (DUM(N2),NA,DUM(N6),ND,H,NH)
      NH(2)=KP(2)
      IF (ICS.EQ.1)  GO TO 60
      CALL TRANP (H,NH,G,NG)
      GO TO 150
   60 CALL MULT  (DUM(N5),NA,H,NH,G,NG)
      CALL TRANP (G,NG,DUM(N6),ND)
      CALL MULT  (DUM(N6),ND,R,NR,G,NG)
      GO TO 150
  100 CALL MULT (DUM(N2),NA,DUM(N6),ND,H,NH)
      NG(2)=KP(2)
      CALL TRANP  (H,NH,G,NG)
      CALL MULT  (G,NG,DUM(N5),NA,H,NH)
      CALL TRANP  (H,NH,DUM(N6),ND)
      CALL MULT  (R,NR,DUM(N6),ND,H,NH)
  150 DUM(N6)=KP(2)
      RETURN
  601 CALL LNCNT (1)
      WRITE  (6,1601) NR,KDUM,MM7
 1601 FORMAT(' DIMENSION ERROR IN DECSYM (DECGEN)   NR='2I6,3X,
     1     'KDUM='I4,3X,'KDUM(MIN)='I4)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE READ1 (A,NA,NZ,NAM)
      COMMON  /MAX/MAXRC
      DIMENSION  A(1   ),NA(2),NZ(2)
      DOUBLE PRECISION A
      IF  (NZ(1).EQ.0)   GO TO 410
      NR=NZ(1)
      NC=NZ(2)
      NLST=NR*NC
      IF(NLST .GT. MAXRC .OR. NLST .LT. 1.OR.NR.LT.1)  GO TO 16
      DO 400 I = 1, NR
  400 READ (5,101) (A(   J), J = I,NLST,NR)
      NA(1)=NR
      NA(2)=NC
  410 CALL   PRNT (A,NA,NAM,1)
  101 FORMAT (8F10.2)
      RETURN
   16 CALL LNCNT(1)
      WRITE  (6,916)  NAM,NR,NC
  916 FORMAT  (' ERROR IN READ1   MATRIX 'A4,' HAS NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE ASPERR
      DATA I /10/
      CALL TRACE
C    ERRTRA IS THE 360/67 TRACE ROUTINE      TRACE IS FOR TSS
C     CALL ERRTRA
C     THIS IS AN INSTALLATION DEPENDENT SUBROUTINE
C     SUBROUTINE ERRTRA IS A SUBROUTINE SUPPLIED BY THE AMES OPERATING
C          SYSTEM TO PROVIDE AN ERROR WALKBACK
C     THE STATEMENT   CALL ERRTRA  SHOULD BE EITHER
C          1) CHANGED TO MATCH THE USERS OPERATING SYSTEM,
C       OR 2) DELETED ALTOGETHER.
      I=I-1
      IF (I.GT.0)  RETURN
      I=10
      WRITE (6,100)
  100 FORMAT (' TOO MANY ERRORS.   EXIT CALLED')
      CALL EXIT
      RETURN
      END
```

```
      BLOCK DATA
      COMMON  /FORM/NEPR,FMT1(6),FMT2(6)
      COMMON/LINES/NLP,LIN,TITLE(23)
      COMMON  /MAX/MAXRC
      DATA  MAXRC/6400/
C- NOTE NLP NO. LINES/PAGE VARIES WITH THE INSTALLATION.
      DATA LIN,NLP/1,45/
      DATA  NEPR,FMT1     /7,'(1P7D16.7)'/
      DATA  FMT2/'(3X,1P7D16.7)'/
      DATA  TITLE  /19*'    ','VASP PROGRAM    '/
      END
```

```
      SUBROUTINE  PSEU(A,B,C,EE,DEP,IP,  D)
C SUBR PSEU GENERALIZED INV BY ANDR.  ALG. J ANDREWS, INF SYS CO. APR 69
C- SUBR TO COMPUTE PSEUDO-INVERSE OF GENERAL MATRIX, RETURN FINAL PIVOT
C... NOTE IMPLIT STATEMENTS MUST BE -FIRST- CAN BE REPLACED BY TYPE
      IMPLICIT REAL*8 (D), INTEGER*2 (Q)
      COMMON  /MAX/MAXRC
C DOUBLE PRECISION IS THE ONLY THING ESSENTIAL.
      INTEGER*2 M
      DOUBLE PRECISION A,B,C,EE, D
      DIMENSION A(400),B(400),C(400),EE(400),  D(2000),
     1 KRV(4),
     2 DEP(3), DPR(2), IP(4), JP(5)
      DATA ICC, DFZO /Z40000000, 0.D0 /
      EQUIVALENCE(DDI,FDI,IDD),(DMX,FMX)
      EQUIVALENCE(DDI,DSUM),(DFZO,FZRO,IZ,QZ),(QLL,QR1), (KRV(1),KRC),
     1 (KRV(2),KRC2),(KRV(3),KRC3),(KRV(4),KRC4)
      QPS = 1
      GO TO 1000
       ENTRY    PSEU  P(A,B,C,EE,DEP,IP,  D)
      QPS = IZ
      IP(4) = IP(3)
1000  CONTINUE
      DP1 = DEP(1)
      EF2 = SNGL(DEP(2))
C- SET DEFAULT VALUES OF TOLERANCES
      IF(DEP(1) .EQ. DFZO) DP1 = 2.D-6
      IF(EF2 .EQ. FZRO) EF2 = 1.0
      NCA = IP(4)
C NUMBER OF ROWS OF ORIGNAL INPUT MATRIX
      QR = IP(3)
C- SET SW FOR =0, DO ALL STEPS, NOT=0, THEN WANT RANK ONLY.
      QNT =  QR*NCA
C- TEST DIMENSIONS INPUT FOR REASONABLENESS.
      IF(QNT .LT. 2 .OR. QNT .GT. MAXRC.OR.QR.LT.1)  GO TO 691
C- IF DIMENSIONS ABSURD, PSEU ERR EXIT 1.
      QDCM = IP(1)
      QITR = QDCM
      IF(QDCM .LT. QZ) QITR = QDCM +1
      NR = QR
      QIT = IZ
```

```
C- TEST TO SEE IF SYMMETRIZATION IS NEEDED.
       IF(QPS ) 16, 150, 16
C- TEST TO FIND SMALLER DIMENSION OF MATRIX.
16     IF(QR - NCA) 18,18,19
19     NR = NCA
       QX = 1
       QR2 = QR
       QLL = QR
       QTP = IZ
       GO TO 170
18     CONTINUE
       QX = NR
       QR2 = 1
       QLL = NCA
       QTP  = 1
170    CONTINUE
C- SET ROW-COLUMN LIMIT TO APPROPRIATE CASE, EITHER ROW OR COLM DIMENS.
       DO 181 I =1, NR
       DO 181 K =1, NR
       DSUM = DFZO
       DO 183 J = 1, QLL
       QN = QX*J - QR
       L = QN + QR2*I
       M = QN + QR2*K
183    DSUM = DSUM + A(L)*A(M)
C- SUM OF A(I,LL) * A(K,LL),, LL RUNS 1 TO ROW OR TO COLM LIMIT
C. B = A*A TRANS HAS COLM LIMIT, B = A TRANS * A HAS ROW LIMIT.
       L = (K-1)*NR + I
181    B(L) = DSUM
       GO TO 188
C- HERE MOVE A TO B. A IS ALREADY POSITIVE DEFINITE.
150    DO 151 L = 1, QNT
151    B(L) = A(L)
C-. FORCE SYMMETRIZATION OF B, TO COMPENSATE FOR ROUND-OFF,MULTIPLIC.
188    DO 189 I =1, NR
       DO 189 K =1, NR
C,  B(I,K) = B(K,I) = 1/2 (B(I,K) + B(K,I) )
       L = I + (K-1)*NR
       M = K + (I-1)*NR
```

```
      DSUM = (B(L) + B(M) ) * 0.5D0
      B(L) = DSUM
  189 B(M) = DSUM
C HERE SET UP CALL -INITIAL- OF ANDRA. ONLY COMES HERE ONCE PER MATRIX.
      QNT = NR*NR
      KRC = QNT
      KRC2 = QNT + KRC
      KRC3 = QNT + KRC2
      KRC4 = QNT + KRC3
C-* OMIT SAVING  OF B, IF RANK ONLY AND NO ITERATION
      IF(IP(2) .NE. IZ .AND. QITR .EQ. IZ) GO TO 200
      DO 1891 I =1, QNT
 1891 D(I) = B(I)
 200  CONTINUE
C- SEARCH DIAGONAL OF INPUT FOR LARGEST ELEMENT. USE TO DEFINE FL. PT.
      QR1 = NR + 1
      L = 1
      DMX = DFZO
      M = IZ
      DO 23 I = 1, NR
      DDI = DABS( B(L) )
      IF(FMX .GE. FDI) GO TO 23
      M = L
      FMX = FDI
 23   L = QR1 + L
      IF(M .EQ. IZ) GO TO 692
C- SET TOLERANCE FOR ANDRA  LIMIT OF SIZE OF DIAGONAL.
C TOLERANCE OF ZERO IN ANDRA CALL.
      DPR(1) = DABS(DP1* B(M))
C - ASK FOR ALL ROWS, DONE IN 1 CALL
      JP(1) = IZ
C- JP2 FIRST TIME INITIALIZATION FOR ANDRA
      JP(2) = IZ
      IF(QIT .NE. QZ) GO TO 561
      JP(4) = NR
      M = IZ
      SOLD = - EF2
C --  HAVE FINISHED PRELIM. PART
C  INITIALIZATION FOR ANDRA (DIAGONALIZATION) NOW COMPLETED.
```

```
CALL ANDRA  -TO DIAGONALIZE SYMMETRIC MATRIX.
C CALL ANDRA REDUCES ROWS BY MODIFIED GAUSS METHOD, USING SQRT(PIVOT).
30      CONTINUE
        IF(QITR .EQ. QZ) GO TO 32
C- SAVE OLD VALUES  IN CASE  PIVOT IS REJECTED, UNDER ITERATION OPT.
        DO 31 L =1, QNT
        J = KRC + L
        K = KRC2 + L
        D(J) = B(L)
31      D(K) = C(L)
 32      CALL ANDRA(B,C,DPR,JP)
        JP(1) = QITR
        IR = JP(3)
C- CHECK COMPLETION- IS MATRIX ALL DONE IS MATRIX INVERTIBLE..
        IF(QITR .EQ. IZ .OR. IR .EQ. NR .AND. QIT .EQ. IZ) GO TO 700
CHECK IF ITERATING WITH RHO TEST OR NOT
C* QUIT IF NO ITERATION  OR NO NEW PIVOT FOUND
C- OMIT ITERATION CALCS. IF NO NEW PIVOT. DECREASE TOLERANCE
        IF(JP(5) .EQ. M) GO TO 220
C COMPUTE RHO FOR ESTIMATING THRESHHOLD TO STOP  SS IS RHO
        SS = (BDNRM(NR,C,EE,D,KRV) + BDNRM(-NR,C,EE,D,KRV) ) *EF2  ** IR
C WHY ONLY SNGLE PREC./THIS IS ONLY A ROUGH TEST TO STOP ITERATION.
C THAT-S WHY. SIMILARLY, OTHER USES OF SINGLE PREC.
        IF(SOLD .LT. SS .AND. SOLD .GT. FZRO) GO TO 650
C- IF SUBSTANTIAL IMPROVEMENT TRY AGAIN,
C, OTHERWISE QUIT, RETURN THE A PSEUDO INVERSE, EVEN IF OFF.
220     CONTINUE
        QIT = QIT +1
        SOLD = SS
C/ SAVE PREVIOUS ROW  IN WHICH A PIVOT WAS FOUND
        M = JP(5)
        IF(QIT .EQ. NR)  GO TO 700
C- PUT IN SMALLER TOLERANCE IN CASE DIAGONAL TOO SMALL OTHERWISE.
        DPR(1) = DPR(2) * 2.D-5
C- TRY TO REDUCE 1 MORE ROW.
        IF(IR - NR) 30, 700,  606
650     CONTINUE
C* RESTORE B AND C TO THEIR PREVIOUS VALUES. THE LAST PIVOT HAS BEEN
CREJECTED (BACK-TRACK), WHILE ITERATING.
```

```
      JP(3) = JP(3)  -1
      DO 653 I =1, QNT
      J = KRC + I
      K = KRC2 + I
      B(I) = D(J)
653   C(I) = D(K)
700   CONTINUE
      IR = JP(3)
      M = IZ
C- HERE WISH TO REPLACE MARKERS IN DIAGONAL WITH LEGITIMATE 1.DO
      L = 1
      DO 704 I = 1, NR
      DDI = B(L)
      IF(IDD) 701, 702, 701
701   IF(IDD .NE. ICC) GO TO 7101
      B(L) = 1.DO
      GO TO 704
C AT 7101 FORCE SMALL TRASH TO ZERO.
7101  B(L) = DFZO
702   M = 1
704   L = QR1 + L
C - IF ALREADY TRIED ANOTHER REDUCTION, TO GET MATRIX IN -UPPER- DIAG.
CQR   OMIT PART OF CALCULATIONS IF ONLY RANK IS DESIRED.
      IF(IP(2) .NE. IZ) GO TO 877
C9 QDCM SUPPRESSES LAST PHASE IF DECOM WAS CALLER..
      IF(M .LT. 1 .OR. QDCM .LT. QZ) GO TO 80
C BELOW HAVE SING. MATRIX THAT NEEDS FURTHER WORK.
C- HAVE MATRIX DIAGONALIZED WITH 1S, 0S INTERSPERSED  (A IS SINGULAR)
C- RE-DO TO GET PS-INV THAT MOVES ALL 1S OF DIAGONAL TO UPPER LEFT DIAG.
C.TO COMPUTE U MATRX AS IN ASP, FOR TRANSFORMING ORIG B IN SINGULAR CASE
      L =1
      DO 527 I =1,NR
      DO 525 J =1, NR
      K = (J-1)*NR + I
      IF(B(L) ) 521,522,521
522   C(K) = - C(K)
      C(L) = DFZO
      GO TO 525
521   C(K) = DFZO
```

```
      C(L) = 1.D0
525   CONTINUE
527   L = QR1 + L
C-SAVE RANK SO FAR, SHOULD BE SAME SIZE AFTER RE-INVERSION
      QR2 = IR
      DO 54 I = 1, NR
      DO 54 K =1, NR
      DSUM = DFZO
      QN = (K-1)*NR
      DO 53 J =1, NR
      M = (I-1)*NR + J
      L = QN + J
53    DSUM = C(M)*D(L) + DSUM
      L = QN + I
      EE(L) = DSUM
54    CONTINUE
      DO 56 I =1, NR
      DO 56 K =1, NR
      DSUM = DFZO
      QN = (K-1)*NR
      DO 55 J =1, NR
      L = (J-1)*NR + I
      M = QN + J
55    DSUM = EE(L)*C(M) + DSUM
      L = QN + I
      B(L) = DSUM
56    CONTINUE
C, SET UP FOR SECONDARY ANDRA CALL  NO ITERATION   JP4 = NR
      QIT =1
C GO FIND LARGEST DIAG. ELEMENT AGAIN
      GO TO 200
561   JP(3) = IZ
       CALL ANDRA(B,EE,DPR,JP)
      IR = JP(3)
C- TEST FOR A CHANGE IN RANK ... ERROR
      IF(QR2 - IR) 693, 568, 694
568    CALL  T TRM(NR,EE,D)
C- TRANSFORM C SHARP IN D.. BS = ((U)* D *(U TRP) )
      DO 58 I =1, NR
```

```
      DO 58 K =1, NR
      DSUM = DFZO
      QN = (K-1)*NR
      DO 57 J =1, NR
      M = (J-1)*NR + I
      L = QN + J
57    DSUM = C(M)*D(L) + DSUM
      L = QN + I
      B(L) = DSUM
58    CONTINUE
      DO 60 I =1,NR
      DO 60 K = 1, NR
      DSUM = DFZO
      DO 59 J =1, NR
      QN = (J-1)*NR
      M = QN + K
      L = QN + I
59    DSUM = B(L)*C(M) + DSUM
      L = (K -1)*NR + I
      EE(L) = DSUM
60    CONTINUE
C- NOW RE-ENTER MAIN SEQUENCE WITH PS-INV. IN EE.
      GO TO 808
C GO FIX UP B PSUEDO-INVERSE. PRESUMABLY HAVE DIAGONALIZED
C HAVE DIAGONALIZED WITH ALL 1S IN UPPER LEFT
C- HERE WE HAVE FINISHED DIAGONALIZ. WANT TO GET PSUEDO INV. IN B.
870   IF(QDCM .LT. QZ) GO TO 877
C NEED TO SAVE DIAGONALIZED B FOR USE BY DECOM CALL (QDCM NEG. FLAG)
      DO 871 I = 1,QNT
C- A WAS SYMMETRIC.  JUST MOVE EE TO B  RETURN FROM PSEUP ENTRY
871   B(I) = EE(I)
      GO TO 877
80    CONTINUE
C NOW FORM (T TRP) * T = APPROX B SHRP PSUEDINV IN MATRX EE
      CALL T TR M(NR,C,EE)
808   IF(QPS .EQ. QZ) GO TO 870
      IF(QTP) 819, 819,818
C HERE B = (A TRANS)*E = (A TRP)*(A*A TRP)-SHRP  NRA .LE. NCA
818   DO 8181 I = 1,NCA
```

```
      DO 8181 J = 1,NR
      DSUM = DFZO
      QN = (J -1)*NR
      DO 8182 K = 1,NR
      L = (I -1)*QR + K
      M = QN + K
8182  DSUM = DSUM + A(L)*EE(M)
      L = (J-1)*NCA + I
8181  B(L) = DSUM
      GO TO 877
819   DO 8191 I = 1,NR
      DO 8191 J =1, QR
      DSUM = DFZO
      DO 8192 K = 1,NR
      L = (K -1)*QR + J
      M = (K -1)*NCA + I
8192  DSUM = DSUM + A(L)*EE(M)
C- NOTE NCA IS USED, BECAUSE A-SHARP IS TRANSPOSED IN DIMENSIONS
      L = (J -1)*NCA + I
C HERE B = EE (A TRANS) = (A TRP*A)-SHRP * (A TRANS)  NRA .GT. NCA
8191  B(L) = DSUM
C- HERE GET READY TO RETURN
877   CONTINUE
C- MOVE RANK TO RETURN PARAMETER
      IP(2) = IR
      DEP(3) = DPR(2)
C. ABOVE RETURN FINAL PIVOT FROM ANDRA ALG. DIAGONALIZATION
      RETURN
  691 CALL LNCNT(1)
      WRITE (6,1691) QR,NCA
 1691 FORMAT (' DIMENSION ERROR IN PSEU NA='2I6)
      GO TO 1700
  692 CALL LNCNT(1)
      WRITE (6,1692)
 1692 FORMAT (' ERROR IN PSEU - DIAGONAL ELEMENTS OF MATRIX=0')
      GO TO 1700
  693 CALL LNCNT(1)
      WRITE (6,1693)
 1693 FORMAT (' ERROR IN PSEU   RANK HAS DECREASED   COMPUTATION ENDED')
```

```
      GO TO 1700
  694 CALL LNCNT(1)
      WRITE (6,1694)
 1694 FORMAT (' ERROR IN PSEU-RANK HAS INCREASED-COMPUTATION CONTINUES')
      CALL ASPERR
      GO TO 568
  606 CALL LNCNT(1)
      WRITE  (6,1606)
 1606 FORMAT  (' ERROR IN PSEU  RANK IS GREATER THAN MATRIX SIZE')
 1700 CALL ASPERR
      RETURN
      END
```

```
      FUNCTION BDNRM(NR,CT,EE,D,KRV)
      INTEGER*2 QF
      DOUBLE PRECISION CT,EE,D, AN,BR, DFZO,DSUM
      DIMENSION CT(400),EE(400), D(2000), NV(2), KRV(4)
C- D HOLDS 5 MATRICES. THE FIRST AND THE LAST 2 ARE USED HERE
      DIMENSION PPP(2)
      EQUIVALENCE(AN,FN), (BR,FR)
      DATA DFZO /0.D0/
C, EQUIVALENCES BELOW JUST TO SAVE STORAGE
      EQUIVALENCE(DFZO,IZ),(AN,DSUM),(BR,PPP(1),I),(PPP(2),K),
     1 (NV(1),L),(NV(2),M),(IR,NL)
C TEST,, IF NR NEG.,THEN TRANSPOSE ROLES OF D       AND (CT TRANS  *CT)
      QF = NR
      KD3 = KRV(3)
      KD4 = KRV(4)
      IF(NR) 10,10, 20
      ENTRY TTRM(NR,CT,EE)
C TO DO T TR * T ONLY ENTRY TTRM
      QF = IZ
      GO TO 20
10    NR = -NR
20    IR = NR
      DO 30 I = 1, IR
      LL = (I-1)*IR
      DO 30 K = 1, IR
      DSUM = DFZO
      KK = (K -1)*IR
      DO 29 J = 1, IR
      L = J + LL
      M = J + KK
29    DSUM = DSUM + CT(L)*CT(M)
C ABOVE FORMAING T TRANSPOSE TIMES T. WHICH  IS APPROX . OF B SHARP
      L = I + KK
      IF(QF) 31, 39, 32
31    KK = KD3 + L
      D(KK) = D(L)
      EE(L) = DSUM
```

```
      GO TO 30
C-39 COMPUTE T TRANSPOSE * T ONLY.. PROVIDES INVERSE B SHARP
39    EE(L) = DSUM
      GO TO 30
32    EE(L) = D(L)
      KK = KD3 + L
      D(KK) = DSUM
30    CONTINUE
      IF(QF) 41,66,41
41    NV(1) = IR
      NV(2) = IR
C- LET P = 1ST MATRIX = EE,, Q = 2D = D(KD3+1)
CCC ROLES OF P AND Q ARE GIVEN TO 2 MATRICES AT 31, SWITCHED AT 32.
COMPUTE D(K4) = Q*P, EE=D(K4)*Q, EE = EE -D(K3) = Q*P*Q - Q
C-FUNCTION IS NRM(Q*P*Q -Q)/NRM(Q)  RESULT = A SCALAR
      CALL MULT(D(KD3+1),NV,EE,NV,D(KD4+1),NV)
      NL = IR*IR
      CALL MULT(D(KD4 +1),NV,D(KD3+1), NV,EE,NV)
      DO 8 I = 1, NL
      KK = KD3 + I
8     EE(I) = EE(I) - D(KK)
      CALL NORM(EE,NV,AN)
      CALL NORM(D(KD3+1),NV,BR)
COUQTIENT NEARS 0.0 AS BSHRP APPROACHES THAT FITTING 2 MOOR-PENRSE AXIOM
      BDNRM = EN / ER
9     RETURN
   66 BDNRM=EN
C66 IS A DUMMY REALLY WANT  MATRX MULT. ONLY.
      GO TO 9
C  SIDE COMPUTATIONS   J W ANDREWS  INF. SYSTEMS CO. MAY 1969
      END
```

```
      SUBROUTINE ANDRA(B,T,DPR,JP)
C- SUBROUTINE ANDRA DIAGONALIZES POS.DEF.SYMM. J ANDREWS  I. S. CO.
C - SUBR ANDRA CALLED BY PSEU J W ANDREWS, INF. SYSTEMS CO. APRIL 1969
      IMPLICIT REAL*8 (D), INTEGER*2 (Q)
      DOUBLE PRECISION B, T
      DIMENSION B(400), T(400), DPR(2), JP(5)
      EQUIVALENCE(DDI,FDI,IDD),(DCC,ICC),(DMX,FMX),(DRS,IIS)
      EQUIVALENCE (DFZO,FZRO,IZ)
      DATA ICC, DFZO /Z40000000, 0.D0/
C-DPR1 IS MAGNITUDE THAT IS CONSIDERED ZRO PIVOT MUST BE NO SMALER.
C- DPR(2) IS TO RETURN FINAL PIVOT, SO THAT USER MAY TEST SMALLNESS.
CC- ANDRA CAN BE USED ALL BY ITSELF TO GET INV., RANK OF POS SYMM.
C NOTE THAT DSQRT HAS TO BE TAKEN OF PIVOTS ALONG THE DIAGONAL.
C- NOTE I AM DELIBERATELY ALLOWING SOME PARAMETERS TO CHANGE ON SUBSE-
C-QUENT CALL  DPR(1) CHANGES PIVOT SIZE  A ROUGH TOLERANCE FOR ZRO.
      EF = SNGL(DPR(1) )
C- TEST- IS THIS AN INTIALIZATION CALL/
      IF(JP(2)) 2, 1, 2
C INTIALIZE- FORM IDENTITY MATRIX
1     QS = JP(4)
      QNT = QS*QS
      IF(QS .LT. 1 .OR. QNT .GT. 6400) GO TO 691
      DO 18 I = 1, QNT
18    T(I) = DFZO
      L = 1
      QR1 = QS +1
      DO 1810 I = 1, QS
      T(L) = 1.D0
1810  L = QR1 + L
      DPR(2) = DFZO
C- SET RANK TO ZRO. TRIAL PIVOT VALUE TO ZERO.
      QKR = IZ
C  SET PIVOT CHOICE ITERATION AT 0  ALLOWANCE OF NO. ROWS+1  ITER.
      QITR = IZ
2     CONTINUE
200    CONTINUE
C- ZERO OUT MAX DIAG. AND CT DIAG. TEMPORARY VARIABLES
```

```
      FMX = FZRO
      I = IZ
      M = IZ
C- BELOW SEE IF ALL DIAG ELEMENTS TESTED YET
      L = 1 - QR1
30    IF(I .EQ. QS) GO TO 40
      I = I +1
      L = QR1 + L
      DDI = B(L)
C- GET CURRENT DIAG. ELEMENT FOR INTEGER, SINGLE PREC. TEST
C- UPDATE L TO GET -NEXT- DIAG. ELEMENT
C-BELOW TEST FOR DIG. ELEMENT @ALREADY@ REDUCED TO 1.(CODEMARKED),ICC
      IF(IDD .EQ. ICC) GO TO 30
      IF(FDI - FMX) 30,30, 32
C- TEST FOR NEGLIGIBLE FL. PT. QTY.-TREAT THESE, AND NEG., AS ZEROS.
32    IF(FDI .LT. EF) GO TO 30
C- SET NEW MAX, 2BLE PREC., SAVE BEST ROW FOR PIVOT  @QMR@
      QMR = I
      DMX = DDI
      M = L
      GO TO 30
40    CONTINUE
400   IF(M .EQ. IZ) GO TO 505
      DRS = 1. DO /DSQRT(DMX)
C SET INDEX OF FIRST ROW, QMR (PIVOT) COLUMN
      K = (QMR -1)*QS  +1
      L = QMR
      DO 41 I = 1, QS
      DDM = B(L)*DRS
      T(L) = T(L)*DRS
      B(L) = DDM
C--SYMMETRICALLY, FORCE COLUMN TO SAME VALUE IN B ONLY
      B(K) = DDM
      K = K +1
41    L = QS + L
C FORCE PIVOT ELEMENT TO EXACT VALUE OF UNITY
      B(M) = 1.DO
C- NOW REDUCE ALL OTHER ROWS OF B, T, ELIMINATING COLUMN OF PIVOT VARIAB
      DO 460 I = 1, QS
```

```
C- TEST FOR PIVOTAL ROW. OTHER ROWS
      IF(I .EQ. QMR) GO TO 460
COEFF, TO BE ZEROED  CAN NOT BE PREVIOUS PIVOT.
      J = I - QS
      K = QS*I + J
      DRS = B(K)
C  BELOW TEST FOR A ROW ALREADY REDUCED, TO SKIP
      IF(IIS .EQ. ICC) GO TO 460
C- GET COEFF IN PIVOT COLUMN TO BE ELIMINATED
      K = QMR*QS + J
      DMM = - B(K)
      L = QMR
      K = I
      DO 47 J =1, QS
C- L IS ROW USED TO REDUCE, WITH PIVOT.
C  K IS CURRENT ROW THAT PIVOT GETS ELIMINATED FROM.
      B(K) = B(K)  + B(L)*DMM
      T(K)  = T(K) + T(L)*DMM
      L = QS + L
47    K = QS + K
460   CONTINUE
      L = QMR
      DO 461 I =1, QS
C  FORCE MOST OF PIVOT ROW TO ZERO. COMPLETES REDUCTION WITH 1 PIVOT/
      B(L) = DFZO
461   L = QS + L
C FORCE PIVOT TO @CODE@ FOR ONE  ..
      B(M) = DCC
C- SIGNAL NO LONGER FIRST TIME CALLED.
      JP(2) = 1
C--   UPDATE EFFECTIVE RANK FOUND
      QKR = QKR +1
      DPR(2) = DMX
      JP(5) = QMR
C- NOW TEST -IS THIS AN ITERATION TO DO ONLY 1 ROW AT A TIME/
      IF(QKR .EQ. QS) GO TO 480
      IF(JP(1) .EQ. IZ) GO TO 490
C( AT THIS POINT, EITHER STOP WITH ONE ROW OR TRY NEXT.
C HERE GET READY TO RETURN. RANK PARAMETER.
```

```
480    JP(3) = QKR
       RETURN
C IF ENOUGH TRIES TO D ALL ROWS PLUS 1 MORE, QUIT.
490    IF(QITR .EQ. QR1) GO TO 480
       QITR = QITR +1
       GO TO 200
  691 CALL LNCNT(1)
       WRITE (6,1691) QS,QNT
 1691 FORMAT  (' DIMENSION ERROR IN ANDRA NR=',I4,5X,'NR*NC='I4)
       RETURN
  692 CALL LNCNT(1)
       WRITE (6,1692)
 1692 FORMAT (' ERROR IN ANDRA, FINDS NO PIVOTS')
CHECK FOR DIAGONAL ALLOWING NO PIVOTS//
505    IF(JP(2) .EQ. IZ .OR. QKR .GT. QR1) GO TO 692
       GO TO 480
       END
```

```
      SUBROUTINE DECOM (A,B,C,E,JL,DCM,KP,  D)
C- SUBR DECOM FINDS 3 MATRICES FROM WHICH USER CAN GET DECOMPOSITION
C INTO KRONECKER PRODUCT, POSSIBLY USING A SEPARATE PSEU CALL
C.. SUBR. DECOM  INFORMATION SYSTEMS CO. MAY 1969.  J W ANDREWS
      IMPLICIT REAL*8 (D), INTEGER*2 (Q)
      DOUBLE PRECISION A,B,C,E, D, PIV
      COMMON  /MAX/MAXRC
      DIMENSION A(400),B(400),C(400),E(400),  D(2000),
     2 JL(400),DCM(3),KP(4),NV(2),  ND(2)
      EQUIVALENCE (NV, I),  (PIV, ND(1)), (DFZO,IZ,QZ), (ND(2), J),
     1 (ND(1), QR1)
      DATA  DFZO /0.D0/
C- SET PARAMETERS TO CALL PSEU. TO FIND T TRANSFORMATION IN C.
C-- ASSUME INPUT A IS POS. DEFINITE SYMMETRIC
      QS = KP(3)
      QNT = QS*QS
C-ERR EXIT IF ABSURD DIMENSIONS.
      IF(QS .LT. 2 .OR. QNT .GT.MAXRC)  GO TO 691
C- SET COLUMN SIZE = RANK SIZE
      KP(4) = QS
      QL = KP(1)
C- SET SPECIAL PARAMS FOR PSEU CALL THESE ARE TO SUPPRESS THE WORK OF
C RE-INVERTING PSEUDO INVERSE IN THE CASE WHERE A SINGULAR...
      KP(1) =  - KP(1) -1
C- CALL PSEU P TO GET MATRIX T. IN C
CNOTE THE LAST 3 MATRICES OF THE 5 IN D USED ONLY IF PSEUP @ITERATES@
      I = KP(2)
          CALL PSEU P(A,B,C,E,DCM,KP,  D)
      KP(1) = QL
      IF(I .NE. IZ) GO TO  38
C/ PLEASE DO NOT TRY TO TAKE A.S.P. NAMES FOR MATRICES HERE.
C- SUCH MATRICES WERE NOT RETURNED BY ASP, NOR BY MY. ROUTINE.
C
      DO 13 I = 1, QNT
13    D(I) = C(I)
      NV(1)=QS
      NV(2)=QS
      ND(1) = 2
C- ND IS PART OF FLAG (PIV) RETURNED BY INV.
        CALL INV(D,NV,PIV,JL)
```

```
C- TEST TO SEE IF PIV IS ZERO = ERR, MATRIX NOT INVERTIBLE.  -ND1-
      IF(ND(1) .EQ. IZ) GO TO 692
38    CONTINUE
      KD - QNT
C- P IS TO HOLD PERMUTATION MATRIX SUCH THAT THAT  P*BE* PTR = ER
C- ER HAS ALL ONES MOVED TO EXTREME UPPER LEFT OF DIAGONAL.
C*NOW SET UP TO MAKE P  PERMUTATION MATRIX  P = D(KD +1)
      QR1 = QS +1
C- ZERO OUT P, WILL BE ZEROS AND ONES
      DO 39 I = 1, QNT
C-ZERO HOUSEKEEPING ARRAY  ONLY NEED FIRST COLUMN.
      JL(I) = IZ
      K = KD + I
39    D(K) = DFZO
      L =1
      M =1
      QL =1
      DO 780 K = 1, QS
      IF( B(L) ) 7803, 7801, 7803
7803  J = JL(QL)
CHECK FOR ROW OF DIAG. THAT NEEDS A 1 MOVED INTO IT
      IF( J .EQ. IZ) GO TO 786
      I = (K-1)*QS + J + KD
C-PUT 1 IN P TO MOVE K,K TO J,K POSITION  (2 PERMUTATIONS) P,, P TRANS
C*  THE EFFECT IS TO MOVE K,K TO J,K/ THENCE TO J,J.
      D(I) = 1.D0
      QL = QL +1
C/MARK THIS 1 AS ZRO TO BE FILLED-- IT IS MOVED UP AND OUTOF HERE
7801  JL(M) = K
      M = M +1
      GO TO 780
786   J = KD + L
C.MAKE PART OF IDENTITY AT 786 DON-T NEED TO MOVE 1 TO A HOLE.
      D(J) = 1.D0
780   L = QR1 + L
C RETURN. MATRICES COMPLETED E WITH IR 1@S DELIBERATELY LEFT OUT.
      RETURN
  691 CALL LNCNT(1)
      WRITE (6,1691)  QS,QNT
```

```
1691 FORMAT  (' DIMENSION ERROR IN DECOM  NC=',I4,5X,'NR*NC=',I4)
     RETURN
 692 CALL LNCNT(1)
     WRITE (6,1692)
1692 FORMAT  (' ERROR IN DECOM   PIVOT=ZERO')
     KP(4)=-QS
     GO TO 38
     END
```

APPENDIX C

USE OF VASP ON AMES' TSS

NONCONVERSATIONAL (BATCH OR RJE)

In using VASP on TSS, the system must be told about the job library in which the VASP subroutines are located, the source of input data, and the location to send output data; and the block data program must be loaded.

A procedure has been written for doing this automatically. The call to the procedure is

VASP$$ [input data set] [,output data set]

The procedure will then perform the steps indicated above. If the first parameter is omitted, the data will be taken from SYSIN, which is from cards in your data deck. If an input data set is named, then the data will be taken from the named data set, which must have been stored previously.

Likewise, if the second parameter is omitted, the output will be placed in SYSOUT, for printing on the high-speed printer. If an output data set is named, the output will be placed in that data set.

If the name of the input or output data set must be changed, use the procedure call

CHNGIN [new input data set name]

CHNGOUT [new output data set name]

These two procedures will then change the DDEF to the new data set name. If the parameter is omitted, the new data set name will be SYSIN or SYSOUT. A listing of these procedures is included in this appendix.

CONVERSATIONAL

Provisions have also been made to allow conversational use of the VASP program, so that the user can easily perform matrix operations. The operations can be strung together in a sequence as desired with as much output as desired. The user indicates the operations by use of Fortran statements, and may not only call the VASP subroutines, but also may execute any other Fortran statements that he wishes.

Data are requested for the program by means of subroutine INPUT, allowing free-form data from the typewriter. If Fortran type input is used, the data should also be obtained from the typewriter. If you try to use an input data set, INPUT will also read the same data set. Variables may also be set by Fortran arithmetic statements.

Output may be from the VASP subroutine PRNT, or any Fortran WRITE or PRINT statement. Two standard formats are available if desired for unlabeled output.

The program automatically dimensions 14 arrays to the desired size, and the user may supply his own names to 7 of them.

Usage

The use of conversational VASP is demonstrated by the accompanying figure (fig. 9). Lower case letters are input and upper case are the computer responses. Detailed comments on the various statements follow. To start, the user calls VASP$$ (line 1) as for nonconversational usage. If desired, an output data set may be named. Line 2 lists the DDNAME being used.

The next two lines (lines 3 and 4) indicate where input and output are to reside. The computer then gives an underscore, after which the procedure "CONVASP" is called. The parameters of this procedure are first the total number of elements in a matrix, followed by up to seven matrix names. If the parameters are defaulted, the system will select matrices with 9 elements, and name the matrices A, B, C, W, X, Y, Z. In addition, 7 dummy matrices D1 through D7 are available for use. In the figure, all matrices are to be dimensioned 16 (line 5), the second matrix is to be renamed F, and the Z matrix is to be renamed FSTAR. That is, if you wish to rename a specific matrix, put a dollar sign in front of the original name and then equate it to the desired name as in the example. Fourteen arrays, NA through ND7, used for dimension information, are also defined and renamed to agree with the working matrices.

Lines 6, 7, and 8 then define the matrices available. Note that no 1-element variables are defined. The user may define them in his program but they will not be available from one computation to the next.

The computer will then ask for FORTRAN STATEMENTS?. At this point, a data set SOURCE.MNPG$$ has been set up for editing and the necessary DIMENSION and other initializing statements have been stored. These statements are listed in figure 12, lines 4600 through 6000. The computer prompts the user with 100 and the user may enter any Fortran statements he wishes. The full power of the text editor is available at this point.

In the example, we have entered four statements, lines 10 through 13. Note that we have defined a single variable t for use in the etphi statement. The value of this variable will not be remembered by the system.

After completing the desired Fortran statements, the user requests compilation by entering _CMPL (line 14). The computer then indicates that compilation is proceeding (line 15) and will give the usual error messages if the compile is unsuccessful. After compilation the program is automatically executed, and the first item in the execution is a request for data from the INPUT subroutine (line 16). Data are entered free style as in line 17, with the elements of the matrices

```
vasp$$
 DDNAME=JBLB0001
INPUT FROM TERMINAL
OUTPUT TO TERMINAL
convasp 16,,f,$z=fstar
*****MATRICES AVAILABLE, ALL DIMENSIONED 16, ARE;
        A,F,C,W,X,Y,FSTAR;  FOR INPUT OR COMPUTATIONS
        D1,D2,D3,D4,D5,D6,D7;  FOR COMPUTATIONS ONLY
*****FORTRAN STATEMENTS?
0000100  t=1.0
0000200  call etphi (f,nf,t,a,na,d1,32)
0000300  call prnt (f,nf,'f    ',1)
0000400  call prnt (a,na,'a    ',1)
0000500_cmpl
*****MNPG$$ NOW COMPILING*****
DATA?
f=1.1,1.2,1.3,1.4,2.1,2.2,2.3,2.4,3.1,3.2,3.3,3.4,4.1,4.2,5.3,4.4,nf=4,4*

      F      MATRIX         4 ROWS         4 COLUMNS
  1.1000000D 00    2.1000000D 00    3.1000000D 00    4.1000000D 00
  1.2000000D 00    2.2000000D 00    3.2000000D 00    4.2000000D 00
  1.3000000D 00    2.3000000D 00    3.3000000D 00    4.3000000D 00
  1.4000000D 00    2.4000000D 00    3.4000000D C0    4.4000000D 00

      A      MATRIX         4 ROWS         4 COLUMNS
  7.7251647D 03    1.3690166D 04    1.9656167D 04    2.5622168D 04
  8.0162773D 03    1.4208825D 04    2.0399372D 04    2.6590919D 04
  8.3083898D 03    1.4725483D 04    2.1143577D 04    2.7559671D 04
  8.6005023D 03    1.5243142D 04    2.1885782D 04    2.8529422D 04
*****COMPUTING DONE*****
recmpt
DATA?
f=1.11,1.22,1.33
f(4)=2.11,2.22,2.33,
f(7)=3.11,3.22,3.33
nf=3,3
*
```

Figure 9.– Example of conversational VASP.

```
     F       MATRIX          3 ROWS          3 COLUMNS
 1.1100000D 00    2.1100000D 00    3.1100000D 00
 1.2200000D 00    2.2200000D 00    3.2200000D 00
 1.3300000D 00    2.3300000D 00    3.3300000D 00

     A       MATRIX          3 ROWS          3 COLUMNS
 1.5033413D 02    2.6866611D 02    3.8799809D 02
 1.5705153D 02    2.8346117D 02    4.0787080D 02
 1.6476893D 02    2.9625623D 02    4.2874352D 02
***COMPUTING DONE***
rewrt
***FORTRAN STATEMENTS?
0000100  call mult (a,na,x,nx,y,ny)
0000200  call prnt (y,'y   ',1)
0000300_revise 200
0000200  call prnt (y,ny,'y   ',1)
0000300_insert 150
0000150  print 6,f
cmpl
*****MNPG$$ NOW COMPILING*****
DATA?
nx=3,1,x=1.0,0.,0.*
 1.110000D 00 1.220000D 00 1.330000D 00 2.110000D 00 2.220000D 00 2.330000D 00
 3.110000D 00 3.220000D 00 3.330000D 00 3.200000D 00 3.300000D 00 3.400000D 00
 4.100000D 00 4.200000D 00 4.300000D 00 4.400000D 00

     Y       MATRIX          3 ROWS          1 COLUMNS
  1.5033413D 02
  1.5705153D 02
  1.6476893D 02
*****COMPUTING DONE*****
rewrt 150
0000100  CALL MULT (A,NA,X,NX,Y,NY)
0000150  call add (x,nx,y,ny,w,nw)
0000250  call prnt (w,nw,'w   ',1)
0000350  call prnt x,nx,'x   ',1)
0000450_cmpl
*****MNPG$$ NOW COMPILING*****
```

Figure 9.– Example of conversational VASP – Continued.

```
0000350 E *** (  NOT FOUND WHERE REQUIRED
0000350   CALL PRNT X,NX,'X    ',1)
#350,  call prnt (x,nx,'x    ',1)
#
 MODIFICATIONS?
n
DATA?
*

     W     MATRIX        3 ROWS        1 COLUMNS
   1.5133413D 02
   1.5705153D 02
   1.6476893D 02

     X     MATRIX        3 ROWS        1 COLUMNS
   1.0000000D 00
   0.0
   0.0
*****COMPUTING DONE*****
Logoff
     9.729 CPU SECONDS, 05/11/71 AT 11:35, FST04       R3984
```

Figure 9.– Example of conversational VASP – Concluded.

being entered columnwise. Do not forget to input the matrix dimensions such as NF in the example. Data entry is ended with an *. Execution of the program continues; lines 18 through 27 display the requested output, and line 28 indicates completion.

At this point (line 29) the computer gives an underscore and the user may do anything he wishes. In the example, we are going to recompute with the same program, using new data. Accordingly, the user asks for RECMPT (line 29). The program is again executed, and new data are asked for (line 30). They are entered in lines 31 through 35, using a different style than in line 17 to show the flexibility available. On completion of the data entry, the results are printed in lines 36 through 44.

At this point, it is desired to rewrite the entire program, so the user issues the command REWRT (line 45). The system, as at line 9, prompts the user with "FORTRAN STATEMENTS," and a line number (lines 46 and 47), after which the user enters Fortran statements as desired. In the example, line 48 is entered incorrectly and then corrected (lines 49 and 50). Following this, a line 150 was inserted (lines 51 and 52). Then a CMPL was issued (line 53) to compile and execute the program. New data were entered at line 56, and lines 57 through 59 are the output requested by the statement "print 6,f." Note that all 16 elements of f are printed using one of the two FORMAT statements compiled into the program for convenience (see lines 5900 and 6000 of VASPPROC, fig. 12):

```
6  FORMAT (1X,1P6D13.6)

13 FORMAT (1X,1P4D20.13)
```

These statements request the output of a 6 decimal number or a 13 decimal number. In the example, we are printing a 6 decimal number. The remainder of the output is then printed (lines 60 through 63).

Now, it is desired to rewrite only a portion of the program from line 150 on. Accordingly, the REWRT command is issued with a parameter (line 65). The system then erases SOURCE.MNPG$$ from line 150 inclusive to the end. It then lists that portion of the program being used, in this case, line 100 only (line 66) and prompts the user for additional lines with a line number (line 67). The user then adds lines as desired (lines 67 through 69) and requests a compile (line 70). It can be seen that line 69 is missing a left parenthesis so the compiler prints a diagnostic and requests the line be corrected (lines 72 and 73). The correction is entered (line 74), after which the compilation is completed (lines 75 through 77). No data are needed, so the data request (line 78) is answered with * only (line 79). The results are printed on lines 80 through 88. Since no more computations were desired, a Logoff command was issued (lines 89 and 90).

Housecleaning

A procedure called "CLRVASP" is available. This procedure erases all data sets that have been set up by the various other procedures, and allows the user to keep his storage low. Use of the routine is not required since the other procedures have appropriate erase statements as needed.

LISTINGS AND FLOWCHARTS

Figure 10 shows all the procedures associated with VASP, and indicates what each one does. A complete listing of the procedures is given in figure 11. Figure 12 is a listing of data set VASPPROC. If the user executes this data set, it will generate all the procedures and place them in the user's USERLIB.

TSS ACCESS

For access to the VASP program, an Ames TSS user should issue the following statements:

SHARE VASP, FSTJSW, VASP

which allows access to the VASP subroutines

SHARE VASPPROC, FSTJSW, VASPPROC
EXECUTE VASPPROC

which first allows access to a data set containing the various procedures, and then enters these procedures in the user's USERLIB. Note that the EXECUTE command sets up a batch job, and that the procedures will not be available until that batch job is completed, and the user has issued either a LOGOFF or ABEND command. After once issuing these commands the user need only call the procedure, as discussed earlier.

Further, for conversational use, issue the command

SHARE VASP1, FSTJSW, VASP1

which allows access to the proper version of subroutine INPUT.

VASP$$
JBLB VASP
LOAD BLKDTA$$
Input & Output DDEF
Default Options

CHNGIN
Change Input DDEF

CHNGOUT
Change Output DDEF

CONVASP
JBLB VASP1
DISPLAY Matrix Names
Edit SOURCE. VASPMN$$
Compile VASPMN$$
Load VASPMN$$
Edit SOURCE. MNPG$$
EXCERPT beginning of Fortran Programs
Display FORTRAN STATEMENTS?

CMPL
Add end of Fortran Program
Display MNPG$$ Now Compiling
Compile MNPG$$
Call MNPG$$
Display COMPUTING DONE

CLRVASP
Erase all Programs & Data used by Conversational VASP

RECMPT
Call MNPG$$

REWRT
EDIT SOURCE. MNPG$$
EXCISE Program
Display FORTRAN STATEMENTS

or

REWRT N
Edit SOURCE. MNPG$$
EXCISE from Statement N to last
List program

Figure 10.– Flowchart VASP procedures.

```
CHNGIN  0000000 PROCDEF CHNGIN
CHNGIN  0000100  PARAM $INPUT
CHNGIN  0000200RELEASE FT05F001
CHNGIN  0000300  IF '$INPUT' ¬='';DDEF FT05F001,,$INPUT;DISPLAY 'INPUT FROM DATA SET $INPUT'
CHNGIN  0000400  IF '$INPUT' ='';DISPLAY 'INPUT FROM TERMINAL'

CHNGOUT 0000000 PROCDEF CHNGOUT
CHNGOUT 0000100  PARAM $OUTPUT
CHNGOUT 0000200  RELEASE FT06F001
CHNGOUT 0000300  IF '$OUTPUT' ¬='';DDEF FT06F001,,$OUTPUT;DISPLAY 'OUTPUT PLACED IN DATA SET $OUTPUT'
CHNGOUT 0000400  IF '$OUTPUT' ='';DISPLAY 'OUTPUT TO TERMINAL'

0000900
CLRVASP 0000000 PROCDEF CLRVASP
CLRVASP 0000050  END
CLRVASP 0000100  UNLOAD MNPG$$
CLRVASP 0000200  UNLOAD VASPMN$$
CLRVASP 0000300  ERASE SOURCE.VASPMN$$,SOURCE.MNPG$$,USERLIB(VASPMN$$)
CLRVASP 0000400  ERASE USERLIB(MNPG$$)
CLRVASP 0000500  RELEASE VASP1
CLRVASP 0000600  DISPLAY '*****ALL CONVERSATIONAL VASP PROGRAMS CLEARED*****'
CLRVASP 0000700  DISPLAY '*****YOU MAY RESTART WITH CONVASP********'

CMPL    0000000 PROCDEF CMPL
CMPL    0000030  DEFAULT SYSINX=E
CMPL    0000050  EDIT SOURCE.MNPG$$
CMPL    0000100_EXCERPT SOURCE.VASPMN$$,,1600,1700
CMPL    0000150  END
CMPL    0000200  DISPLAY '*****MNPG$$ NOW COMPILING*****'
CMPL    0000220  DEFAULT LIMEN=N
CMPL    0000250  UNLOAD MNPG$$
CMPL    0000270  ERASE USERLIB(MNPG$$)
CMPL    0000300  FTN MNPG$$,Y
CMPL    0000400  LOAD MNPG$$
CMPL    0000600  CALL MNPG$$
CMPL    0000700  DEFAULT LIMEN=W
CMPL    0000800  DISPLAY '*****COMPUTING DONE*****'
```

Figure 11.– List of VASP procedures.

```
CONVASP 0000000 PROCDEF CONVASP
CONVASP 0000020  PARAM $N,$A,$B,$C,$W,$X,$Y,$Z
CONVASP 0000040  DDEF VASP1,VP,VASP1,OPTION=JOBLIB
CONVASP 0000080  JOBLIBS SYSULIB
CONVASP 0000110  DISPLAY  '*****MATRICES AVAILABLE, ALL DIMENSIONED $N, ARE;'
CONVASP 0000140  DISPLAY  '           $A,$B,$C,$W,$X,$Y,$Z;  FOR INPUT OR COMPUTATIONS'
CONVASP 0000170  DISPLAY  '           D1,D2,D3,D4,D5,D6,D7;  FOR COMPUTATIONS ONLY '
CONVASP 0000200  DEFAULT SYSINX=E
CONVASP 0000250  DEFAULT LIMEN=N
CONVASP 0000300  EDIT SOURCE.VASPMN$$
CONVASP 0000340_EXCISE 1,LAST
CONVASP 0000380  INSERT 100
CONVASP 0000400  IMPLICIT REAL*8(A-H,O-Z)
CONVASP 0000430  COMMON /ASP/  $A($N),$B($N),$C($N),$W($N),$X($N),-
CONVASP 0000460               1 $Y($N),$Z($N),D1($N),D2($N),D3($N),D4($N),D5($N),D6($N),D7($N),-
CONVASP 0000490               2 N$A(2),N$B(2),N$C(2),N$W(2),N$X(2),-
CONVASP 0000520               3 N$Y(2),N$Z(2),ND1(2),ND2(2),ND3(2),ND4(2),ND5(2),ND6(2),ND7(2)
CONVASP 0000550  COMMON /MAX/ MAXRC
CONVASP 0000580  MAXRC=$N
CONVASP 0000700  10 PRINT 15
CONVASP 0000800  15 FORMAT (' DATA?')
CONVASP 0000840  CALL INPUT ('$A',$A,'$B',$B,'$C',$C,'$W',$W,-
CONVASP 0000880            1 '$X',$X,'$Y',$Y,'$Z',$Z,'N$A',N$A,-
CONVASP 0000920            2 'N$B',N$B,'N$C',N$C,'N$W',N$W,'N$X',N$X,-
CONVASP 0000960            3 'N$Y',N$Y,'N$Z',N$Z)
CONVASP 0001000  13 FORMAT (1X,1P4D20.13)
CONVASP 0001100  6  FORMAT (1X,1P6D13.6)
CONVASP 0001200  RETURN
CONVASP 0001300  END
CONVASP 0001400_END
CONVASP 0001450  ERASE USERLIB(VASPMN$$)
CONVASP 0001500  FTN VASPMN$$,Y
CONVASP 0001600  XLIST VASPMN$$
CONVASP 0001700  LOAD VASPMN$$
CONVASP 0001770  DEFAULT LIMEN=N
CONVASP 0001800  EDIT SOURCE.MNPG$$
CONVASP 0001900_EXCISE 1,LAST
CONVASP 0001950  INSERT 1,1
CONVASP 0002000_EXCERPT SOURCE.VASPMN$$,,100,1500
CONVASP 0002100  DISPLAY '*****FORTRAN STATEMENTS?'
CONVASP 0002150  DEFAULT LIMEN =W
CONVASP 0002200  INSERT 100
```

Figure 11.– List of VASP procedures – Continued.

```
RECMPT   0000000 PROCDEF RECMPT
RECMPT   0000100  DEFAULT LIMEN =N
RECMPT   0000200  CALL MNPG$$
RECMPT   0000300  DEFAULT LIMEN=W
RECMPT   0000400  DISPLAY '***COMPUTING DONE***'


REWRT    0000000 PROCDEF REWRT
REWRT    0000100  PARAM $LINE
REWRT    0000200  DEFAULT LIMEN =W
REWRT    0000400  DEFAULT SYSINX=E
REWRT    0000500  EDIT SOURCE.MNPG$$
REWRT    0000600_EXCISE $LINE,LAST
REWRT    0000700  IF '$LINE'='100';DISPLAY '***FORTRAN STATEMENTS?'
REWRT    0000800  IF '$LINE'¬='100';LIST 100,LAST
REWRT    0000900  DEFAULT SYSINX=G
REWRT    0001000  INSERT $LINE


VASP$$   0000000 PROCDEF VASP$$
VASP$$   0000100  PARAM $INPUT,$OUTPUT
VASP$$   0000150  DEFAULT $N=9,$A=A,$B=B,$C=C,$W=W,$X=X,$Y=Y,$Z=Z,$LINE=100
VASP$$   0000200  JBLB VASP
VASP$$   0000300LOAD BLKDTA$$
VASP$$   0000400  IF '$INPUT' ¬='';DDEF FT05F001,,$INPUT;DISPLAY 'INPUT FROM DATA SET $INPUT'
VASP$$   0000500  IF '$INPUT' ='';DISPLAY 'INPUT FROM TERMINAL'
VASP$$   0000600  IF '$OUTPUT' ¬='';DDEF FT06F001,,$OUTPUT;DISPLAY 'OUTPUT PLACED IN DATA SET $OUTPUT'
VASP$$   0000700  IF '$OUTPUT' ='';DISPLAY 'OUTPUT TO TERMINAL'
```

Figure 11.– List of VASP procedures – Concluded.

```
00030  LOGON USERID,,9
00060  PROCDEF CHNGIN
00090_EXCISE 1,LAST
00100 PROCDEF CHNGIN
00200  PARAM $INPUT
00300RELEASE FT05F001
00400  IF '$INPUT' ¬='';DDEF FT05F001,,$INPUT;DISPLAY 'INPUT FROM DATA SET $INPUT'
00500  IF '$INPUT' ='';DISPLAY 'INPUT FROM TERMINAL'
00540_ PROCDEF CHNGOUT
00580_EXCISE 1,LAST
00600 PROCDEF CHNGOUT
00700  PARAM $OUTPUT
00800  RELEASE FT06F001
00900  IF '$OUTPUT' ¬='';DDEF FT06F001,,$OUTPUT;DISPLAY 'OUTPUT PLACED IN DATA SET $OUTPUT'
01000  IF '$OUTPUT' ='';DISPLAY 'OUTPUT TO TERMINAL'
01040_ PROCDEF CLRVASP
01080_EXCISE 1,LAST
01100 PROCDEF CLRVASP
01200  END
01300  UNLOAD MNPG$$
01400  UNLOAD VASPMN$$
01500  ERASE SOURCE.VASPMN$$,SOURCE.MNPG$$,USERLIB(VASPMN$$)
01600  ERASE USERLIB(MNPG$$)
01700  RELEASE VASP1
01800  DISPLAY '*****ALL CONVERSATIONAL VASP PROGRAMS CLEARED*****'
01900  DISPLAY '*****YOU MAY RESTART WITH CONVASP********'
01940_ PROCDEF CMPL
01980_EXCISE 1,LAST
02000 PROCDEF CMPL
02100  DEFAULT SYSINX=E
02200  EDIT SOURCE.MNPG$$
02300__EXCERPT SOURCE.VASPMN$$,,1600,1700
02400  END
02500  DISPLAY '*****MNPG$$ NOW COMPILING*****'
02600  DEFAULT LIMEN=N
02700  UNLOAD MNPG$$
02800  ERASE USERLIB(MNPG$$)
02900  FTN MNPG$$,Y
03000  LOAD MNPG$$
```

Figure 12.– List of data set VASPPROC.

```
03100  CALL MNPG$$
03200  DEFAULT LIMEN=W
03300  DISPLAY '*****COMPUTING DONE*****'
03340_ PROCDEF CONVASP
03380_EXCISE 1,LAST
03400 PROCDEF CONVASP
03500  PARAM $N,$A,$B,$C,$W,$X,$Y,$Z
03600  DDEF VASP1,VP,VASP1,OPTION=JOBLIB
03700  JOBLIBS SYSULIB
03800  DISPLAY  '*****MATRICES AVAILABLE, ALL DIMENSIONED $N, ARE;'
03900  DISPLAY  '          $A,$B,$C,$W,$X,$Y,$Z;  FOR INPUT OR COMPUTATIONS'
04000  DISPLAY  '          D1,D2,D3,D4,D5,D6,D7;  FOR COMPUTATIONS ONLY '
04100  DEFAULT SYSINX=E
04200  DEFAULT LIMEN=N
04300  EDIT SOURCE.VASPMN$$
04400__EXCISE 1,LAST
04500  INSERT 100
04600  IMPLICIT REAL*8(A-H,O-Z)
04700  COMMON /ASP/  $A($N),$B($N),$C($N),$W($N),$X($N),-
04800             1 $Y($N),$Z($N),D1($N),D2($N),D3($N),D4($N),D5($N),D6($N),D7($N),-
04900             2 N$A(2),N$B(2),N$C(2),N$W(2),N$X(2),-
05000             3 N$Y(2),N$Z(2),ND1(2),ND2(2),ND3(2),ND4(2),ND5(2),ND6(2),ND7(2)
05100  COMMON /MAX/ MAXRC
05200  MAXRC=$N
05300  10 PRINT 15
05400  15 FORMAT (' DATA?')
05500  CALL INPUT ('$A',$A,'$B',$B,'$C',$C,'$W',$W,-
05600           1 '$X',$X,'$Y',$Y,'$Z',$Z,'N$A',N$A,-
05700           2 'N$B',N$B,'N$C',N$C,'N$W',N$W,'N$X',N$X,-
05800           3 'N$Y',N$Y,'N$Z',N$Z)
05900  13 FORMAT (1X,1P4D20.13)
06000  6  FORMAT (1X,1P6D13.6)
06100  RETURN
06200  END
06300__END
06400  ERASE USERLIB(VASPMN$$)
06500  FTN VASPMN$$,Y
06600  XLIST VASPMN$$
06700  LOAD VASPMN$$
```

Figure 12.– List of data set VASPPROC –Continued.

```
06800  DEFAULT LIMEN=N
06900  EDIT SOURCE.MNPG$$
07000__EXCISE 1,LAST
07100  INSERT 1,1
07200__EXCERPT SOURCE.VASPMN$$,,100,1500
07300  DISPLAY '*****FORTRAN STATEMENTS?'
07400  DEFAULT LIMEN =W
07500  INSERT 100
07540_ PROCDEF RECMPT
07580_EXCISE 1,LAST
07600 PROCDEF RECMPT
07700  DEFAULT LIMEN =N
07800  CALL MNPG$$
07900  DEFAULT LIMEN=W
08000  DISPLAY '***COMPUTING DONE***'
08040_ PROCDEF REWRT
08080_EXCISE 1,LAST
08100 PROCDEF REWRT
08200  PARAM $LINE
08300  DEFAULT LIMEN =W
08400  DEFAULT SYSINX=E
08500  EDIT SOURCE.MNPG$$
08600__EXCISE $LINE,LAST
08700  IF '$LINE'='100';DISPLAY '***FORTRAN STATEMENTS?'
08800  IF '$LINE'¬='100';LIST 100,LAST
08900  DEFAULT SYSINX=G
09000  INSERT $LINE
09040_ PROCDEF VASP$$
09080_EXCISE 1,LAST
09100 PROCDEF VASP$$
09200  PARAM $INPUT,$OUTPUT
09300  DEFAULT $N=9,$A=A,$B=B,$C=C,$W=W,$X=X,$Y=Y,$Z=Z,$LINE=100
09400  JBLB VASP
09500LOAD BLKDTA$$
09600  IF '$INPUT' ¬='';DDEF FT05F001,,$INPUT;DISPLAY 'INPUT FROM DATA SET $INPUT'
09700  IF '$INPUT' ='';DISPLAY 'INPUT FROM TERMINAL'
09800  IF '$OUTPUT' ¬='';DDEF FT06F001,,$OUTPUT;DISPLAY 'OUTPUT PLACED IN DATA SET $OUTPUT'
09900  IF '$OUTPUT' ='';DISPLAY 'OUTPUT TO TERMINAL'
```

Figure 12.– List of data set VASPPROC – Continued.

```
LIST
  END
  TALK *
  VASP PROCEDURES NOW READY.  DO 'ABEND' TO MAKE THEM AVAILABLE
  LOGOFF
```

Figure 12.– List of data set VASPPROC – Concluded.

REFERENCE

1. Kalman, R. E.; and Engler, T. S.: A User's Manual for the Automatic Synthesis Program (Program C). NASA CR–475, 1966.

NASA-Langley, 1971 —— 8 A-3882

"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 Technology Surveys.

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

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546