TM-71-1015-1

TECHNICAL MEMORANDUM

MODAL ANALYSIS COMPUTER PROGRAM PACKAGE

Bellcomm

BELLCOMM, INC.
955 L'ENFANT PLAZA NORTH, S.W., WASHINGTON, D.C. 20024

COVER SHEET FOR TECHNICAL MEMORANDUM

TITLE- Modal Analysis Computer Program Package

TM-71-1015-1

DATE- January 7, 1971

FILING CASE NO(S)- 320

AUTHOR(S)- D. L. Mather

FILING SUBJECT(S) (ASSIGNED BY AUTHOR(S))- Modal Analysis Computer Program

ABSTRACT

The fundamental purpose of the modal analysis computer program package (MAP) is to find some or all of the frequencies λ (in cps) and some or all of the (orthogonal) mode shapes y of any linear discrete system which is governed by a generalized eigenvalue equation of the form

$$Ky = \lambda My$$

where (1) M and K are real symmetric matrices, (2) M is positive definite, (3) M is not ill-conditioned with respect to inversion, and (4) M and k are possibly, but not necessarily, large matrices.

The given problem is reduced to a real symmetric algebraic eigenproblem $Px = \lambda x$ by the method of Cholesky (i.e., by the square root method). The real symmetric matrix P is reduced to a real symmetric tridiagonal matrix by the method of Householder. The eigenvalues of the tridiagonal matrix (which are the same as those of the original problem) are found by a Sturm sequence bisection method. The user can have the eigenvectors of the tridiagonal matrix computed by either inverse iteration (or back substitution) or a combination of inverse iteration and Gram-Schmidt orthogonalization. The eigenvectors of the tridiagonal matrix are then transformed back into those of P or into those of the original problem. MAP is capable of finding all or only some of the eigenvalues, and eigenvectors for all or only some of the computed eigenvalues. All accumulations are done in double precision to improve the numerical accuracy.

Only five of the twenty subroutines in the MAP program package are directed toward a modal analysis, as such. The remaining fifteen programs are general purpose mathematical routines which can be used to solve a variety of problems in numerical linear algebra that involve real symmetric positive definite matrices and real symmetric matrices. Each of these general purpose routines has sufficient introductory commentary to make it readily useful. MAP has been put in the Bellcomm Applications Program Library (#0218).

DISTRIBUTION

COMPLETE MEMORANDUM TO

CORRESPONDENCE FILES:

OFFICIAL FILE COPY
plus one white copy for each additional case referenced

TECHNICAL LIBRARY (4)

F. A. Brewer
G. A. Briggs
E. L. Bush (4)
M. L. Carothers
D. E. Cassidy
P. R. Dowling
B. A. Gropper
R. W. Grutzner
M. C. Harris
S. N. Hou
S. Kaufman
M. L. Kratage
P. F. Long
R. K. McFarland
R. E. McGaughy
H. Murrey
D. J. Osias
S. L. Penn
C. S. Rall
T. J. Rudd
J. J. Schoch
R. D. Sharma
E. N. Shipley
H. E. Stephens
F. F. Tomblin

COVER SHEET ONLY TO

Memorandum to without Appendix 2

Bellcomm

G. R. Andersen
G. M. Anderson
D. O. Baechler
A. P. Boysen
W. C. Brubaker
C. L. Davis
J. P. Downs
D. R. Hagner
H. A. Helm
W. W. Hough
A. N. Kontaratos
J. Kranton
D. P. Ling
H. S. London
D. Macchia
E. D. Marion
J. Z. Menard
L. D. Nelson
J. M. Nervik
G. T. Orrok
J. A. Schelke
P. G. Smith
R. V. Sperry
V. Thuraisamy
J. Strand
W. B. Thompson
J. W. Timko
A. R. Vernon
J. E. Volonte
R. L. Wagner
M. P. Wilson

Department 1024 File

BA-146 (8-64)

BELLCOMM, INC.
955 L'ENFANT PLAZA NORTH, S.W. WASHINGTON, D.C. 20024

SUBJECT: Modal Analysis Computer Program Package

DATE: January 7, 1971

FROM: D. L. Mather

TECHNICAL MEMORANDUM

INTRODUCTION

The modal analysis computer program package (MAP) was developed at the request of S. N. Hou for Department 2031 and arose from Apollo structural studies.[1] The fundamental purpose of MAP is to find some or all of the frequencies λ (in cps) and some or all of the (orthogonal) mode shapes y of any linear discrete system which is governed by a generalized eigenvalue equation of the form

$$Ky = \lambda My$$

where (1) M and K are real symmetric matrices, (2) M is positive definite, (3) M is not ill-conditioned with respect to inversion, and (4) M and K are possibly, but not necessarily, large matrices. MAP also provides a user with (1) some flexibility in what mathematical methods are used, (2) the capability of easily implementing different methods, and (3) a great deal of flexibility in what information can be saved as output and how it can be saved.

Method

The generalized algebraic eigenproblem is reduced to an ordinary algebraic eigenproblem by the method of Cholesky[2, 3] (i.e., by the square root method). This technique is used to compute a lower triangular matrix L such that

$$M = LL^T.$$

(If M is positive definite, such a matrix not only exists but is also real.)[4] The original equation can now be written as

$$Ky = \lambda LL^T y.$$

Hence

$$(L^{-1}KL^{-T})\ (L^T y) = \lambda (L^T y),^* \text{ or}$$

$$Px = \lambda x,$$

where $P = L^{-1}KL^{-T}$ and $x = L^T y$.

---

*The notation L^{-T} will be used in this memo rather than the cumbersome $(L^{-1})^T$.

Note that $P^T = P$. Thus the equation $Px = \lambda x$ represents a real symmetric algebraic eigenproblem. The first step in solving this problem is to reduce the (dense) real symmetric matrix P to a real symmetric tridiagonal matrix by the method of Householder.[5, 6] The eigenvalues of the tridiagonal matrix (which are the same as those of the original problem) are found by a Sturm sequence bisection method.[7] The eigenvectors of the tridiagonal matrix can be computed either by inverse iteration (or back substitution)[8] or a combination of inverse iteration and Gram-Schmidt orthogonalization.[9, 10] The eigenvectors of the tridiagonal matrix are then transformed into those of P, i.e, into x. Then, if the user wants them, the vectors x are transformed into the vectors y, and also the units of the eigenvalues are changed to cps. These last two operations are done by computing $y = L^{-T}x$ and $\sqrt{\lambda}/2\pi$, respectively.

The crucial questions in deciding which of the two methods for computing the eigenvectors of the tridiagonal matrix to use are whether the user wants orthogonal vectors and whether there is a multiple eigenvalue. Inverse iteration will produce orthogonal vectors if all the eigenvalues are distinct, but will not, in general, produce orthogonal vectors for a multiple eigenvalue. However, inverse iteration combined with Gram-Schmidt orthogonalization will, in general, yield orthogonal vectors for a repeated eigenvalue. The latter program requires 6n more locations for arrays, where n is the dimension of the matrices M and K, and will, as a rule, require more time than the former.

Nearly all of these methods are known for their accuracy, speed, and ability to take advantage of symmetry. It should, perhaps, be pointed out that the bound on the absolute computational error is the same for all the eigenvalues.[11] That is, if the eigenvalue of largest magnitude is in error by $\Delta\lambda$ then the eigenvalues of smallest magnitude may be in error by the same amount. Thus the relative computational error will, in general, be greater for the eigenvalues of smaller magnitude. More specifically, if the ratio of the eigenvalue of largest magnitude to that of the smallest magnitude is quite large (say, 10^6 or more) then the eigenvalue of smallest magnitude will not, in general, have a low relative error. J. H. Wilkinson illustrates how this happens for a Sturm sequence method.[12]

If only a few of the n eigenvalues are computed then, of course, the obvious way of learning of a wide separation in the eigenvalues is not available. In such a case, there are two indicators of a large separation. One is the trace Tr(P) of the matrix P.*

---

*By definition, the trace Tr(S) of an arbitrary matrix S is the sum of all the main diagonal elements of S.

Since the sum of all n eigenvalues equals Tr(P), it follows that at least one eigenvalue must be as large as Tr(P)/n. Hence, a Tr(P) (or, to be strictly accurate, a Tr(P)/n) that is large compared to the eigenvalue of smallest magnitude indicates the existence of a large eigenvalue, and hence a wide separation in the eigenvalues. Another indicator is the fact that if P denotes a real symmetric matrix which has only positive eigenvalues then

$$\frac{\max \mid p_{ii} \mid}{\min \mid p_{ii} \mid} \leq \frac{\max \mid \lambda_i \mid}{\min \mid \lambda_i \mid}$$

where i ranges from 1 to n.* MAP computes and prints Tr(P), the left hand side of the inequality, the ratio of the largest $|\lambda_i|$ computed to the smallest nonzero $|\lambda_i|$ computed, and the sum of all the computed eigenvalues.

Check Cases

Since MAP was intended for large matrices, illustrative examples involving large matrices whose exact eigenvalues are known would be most appropriate. However, there seems to be a lack of large test matrices for the generalized eigenproblem. Consequently, the first two of the four check cases involve 20x20 and 25x25 matrices. These particular problems were picked since very precise answers for them have been published.

The third example is perhaps the most interesting one. It is that of a vertical simple beam (an idealized form of the Saturn V missile and Apollo spacecraft structure). This example may give a potential user of MAP the most insight into how much precision MAP can be expected to give in the answers to a practical problem. The reason for this is that the error in the results is a function of the discretization error (introduced when the continuous beam was modeled as a series of mass points) and computational error in generating the input matrices M and K as well as the computational error that is introduced by MAP itself. In addition to the preceding reason this example was chosen as its analytic solution is known,[13] it is a source of arbitrarily large matrices, its M and K matrices have nice numerical properties, and the origin of this programming task was a structural problem.

In the last example M = I (the identity matrix) and K is a matrix whose dimension can be made arbitrarily large and whose exact eigenvalues are known.

* Communicated to the author in conversation by S. L. Levie.

Several runs of each of the last two examples were made in which the size of the input matrices was increased in order to illustrate the growth of computational error. The computed results of the runs for all four examples are summarized in the following tables. The computer output of the runs in Examples 3 and 4 are available in the Computer Library.

Example 1.[14] K and M are 20x20 band matrices of full band width 7.*

$k_{ii} = 51-i$ $\quad m_{ii} = 41-i$, for $i = 1,2,\ldots, 20$

$k_{ij} = 1$ $\quad m_{ij} = 1$, for $0<|i-j|\leq 3$.

Published Eigenvalues	Eigenvalues Computed by MAP
1.2362 2996 622	1.2362 300
1.2543 8078 474	1.2543 807
1.2619 2368 457	1.2619 237
1.2694 3952 847	1.2694 395
1.2773 9754 724	1.2773 975
1.2856 3483 441	1.2856 348
1.2940 9698 102	1.2940 970
1.3030 1061 009	1.3030 106
1.3125 0454 161	1.3125 046
1.3226 0009 164	1.3226 000
1.3333 9423 801	1.3333 943
1.3450 0343 860	1.3450 035
1.3575 7195 730	1.3575 720
1.3713 1462 185	1.3713 147
1.3866 8413 225	1.3866 841
1.4034 7245 976	1.4034 725
1.4222 3523 837	1.4222 352
1.4475 1739 434	1.4475 174
1.4704 2713 163	1.4704 271
1.4952 1305 093	1.4952 130

Examples 2.[15] K and M are 25x25 band matrices of full band width 7 . M is the identity matrix and K is the matrix defined

*By definition, a matrix S is said to have full band width 2k+1 if k is the smallest positive integer with the property that $s_{ij} = 0$ for all i and j such that $|i-j|>k$.

by

$$K = \begin{pmatrix} X & -I & 0 & 0 & 0 \\ -I & X & -I & 0 & 0 \\ 0 & -I & X & -I & 0 \\ 0 & 0 & -I & X & -I \\ 0 & 0 & 0 & -I & X \end{pmatrix}, \text{ where}$$

$$X = \begin{pmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 4 \end{pmatrix}$$

Only the eigenvalues between 0.19 and 0.35 were published.

Published Eigenvalues	Eigenvalues Computed by MAP
	.1339 7460
	.1485 4314
	.1485 4316
	.1666 6668
	.1744 5764
	.1744 5765
.2000 0000 0000	.2000 0000
.2000 0000 0000	.2000 0002
.2113 2486 5405	.2113 2487
.2113 2486 5405	.2113 2489
.2500 0000 0000	.2499 9998
.2500 0000 0000	.2500 0000
.2500 0000 0000	.2500 0000
.2500 0000 0000	.2500 0000
.2500 0000 0000	.2500 0002
.3060 0230 9436	.3060 0230
.3060 0230 9436	.3060 0230
.3333 3333 3333	.3333 3334
.3333 3333 3333	.3333 3334
	.4409 2697
	.4409 2697
	.5000 0000
	.7886 7508
	.7886 7508
	.1866 0254

EXAMPLE 3.* This example is that of a vertical simple beam which is of uniform density, is unattached at both ends (i.e. free-free), lies in a vertical plane, and whose motion is restricted to lateral displacements and rotations. MAP tacitly assumes that the beam is discrete, i.e., the mass of the beam is assumed to be concentrated in a finite number of points. One way to think of this is to divide the beam into, say j equal segments, and regard the mass of each segment as being distributed equally at the two endpoints of the segment. Hence a beam that has been divided into j segments will be equivalent to j+1 mass points. It can be shown that since the beam can move in only two directions, each of these j+1 mass points increases the dimensions of the input matrices M and K by two. Thus by dividing the beam into j segments, one will generate M and K matrices of dimension 2(j+1).

The eigenvalues and vectors of a discrete beam differ from those of a continuous beam (with the sole exception that in both cases the smallest eigenvalue is a zero eigenvalue of multiplicity two). As the number of segments tends to infinity, however, they do approach (in the absence of computational error) the eigenvalues and eigenvectors of the corresponding continuous beam. Unlike discretization error, computational error increases as n gets larger. Hence, as n increases, one might expect to see the computed eigenvalues approach those of the continuous beam for a while, due to the reduction of discretization error, and then to see them not getting any closer and even begin losing digits of accuracy due to a substantial growth of computational error. The following table (Table 1) summarizes the performance of MAP in the computation of the eigenvalues and vectors (or modes shapes) of a free-free beam. Table 2 gives some information pertinent to using MAP to solve a generalized eigenproblem - the core requirement and total charges for a given size of matrix may be of special interest. The third eigenvalue, i.e., the first nonzero eigenvalue seems to point out most clearly the reduction of discretization error followed by the eventual dominance of the computational error. The exact value of the i^{th} eigenvalue (to six places) for the continuous beam is given at the head of the column containing values of the i^{th} computed eigenvalue.

---

* The mass and stiffness matrices M and K, respectively, of the the simple beam were generated using a program written by S. N. Hou.

TABLE 1

The Values of the Seven Smallest Computed Eigenvalues as a Function of the Dimension n of the Input Matrices M and K

	Zero Eigenvalues		Nonzero Eigenvalues				
n	0	0	1.54589	4.26131	8.35597	13.8094	20.6288
10	.0048	.0098	1.3028	3.3013	5.9773	44.221	50.468
20	.0110	.0299	1.4842	3.9766	7.5876	12.223	17.803
30	.0274	.0481	1.5166	4.1178	7.9476	12.924	18.982
40	.0398	.0828	1.5281	4.1665	8.0756	13.183	19.429
50	.0593	.1264	1.5341	4.1889	8.1344	13.304	19.640
60	.0922	.1861	1.5397	4.2014	8.1663	13.369	19.755
70	.1432	.2266	1.5456	4.2103	8.1860	13.408	19.824
80	.2062	.2643	1.5518	4.2144	8.1981	13.433	19.868
90	.2603	.3475	1.5581	4.2215	8.2074	13.451	19.899
100	.2508	.4699	1.5759	4.2311	8.2184	13.466	19.922
120	.2253	.4746	1.5726	4.2315	8.2238	13.481	19.950
140	.3050	.5728	1.6025	4.2426	8.2323	13.493	19.967
160	.3122	.6974	1.6297	4.2491	8.2384	13.501	19.980
180	.3643	.8518	1.6823	4.2602	8.2480	13.509	19.990
200	.3983	.9670	1.7081	4.2691	8.2535	13.514	20.000
220	.4868	1.0724	1.7560	4.2955	8.2644	13.521	20.007
240	.5768	1.2724	1.8144	4.3066	8.2801	13.530	20.014

TABLE 2

Information about the Computer Runs as a Function of the Dimension n of the Input Matrices M and K

Main features of the runs:

(1) M matrix diagonal
(2) n eigenvalues computed if n<50 and fifty eigenvalues computed otherwise
(3) three eigenvectors computed (and transformed into the vectors y)
(4) the only output was the printing of the eigenvalues and vectors
(5) one compilation (except for n=10 and n=20 in which cases there were three)

n	Core Requirement	CPU Time	Wall-Clock Time	Core-Seconds	I/O Count	Charge
10	31.7K	0:13	1:14	361	1,125	10
20	31.9K	0:12	1:18	363	1,136	10
30	32.3K	0:08	0:50	222	711	6
40	32.8K	0:12	0:56	260	764	7
50	33.4K	0:15	0:54	291	818	8
60	34.1K	0:20	1:09	353	890	10
70	34.9K	0:26	1:04	417	971	11
80	35.8K	0:34	1:13	494	1,067	13
90	36.8K	0:44	3:39	585	1,173	16
100	37.9K	0:57	7:45	692	1,290	19
120	40.4K	1:26	12:39	1,100	1,562	28
140	43.3K	2:07	3:14	1,367	1,884	35
160	46.6K	3:00	3:57	1,786	2,257	45
180	50.3K	4:39	7:43	3,174	2,676	78
200	54.4K	6:43	9:22	4,201	3,152	103
220	58.9K	9:51	12:59	6,488	3,677	154
240	63.7K	9:13	11:31	7,459	4,237	170

EXAMPLE 4.[16]

$$\begin{pmatrix} n & n-1 & n-2 & \cdots & 2 & 1 \\ n-1 & n-1 & n-2 & \cdots & 2 & 1 \\ n-2 & n-2 & n-2 & \cdots & 2 & 1 \\ & & & \vdots & & \\ 2 & 2 & 2 & \cdots & 2 & 1 \\ 1 & 1 & 1 & \cdots & 2 & 1 \end{pmatrix}$$

The exact eigenvalues are given by

$$\lambda_i = 0.5\left(1-\cos\frac{(2i-1)\pi}{2n+1}\right)^{-1}, \text{ for } i = 1,2,\ldots,n.$$

The relative errors in Table 3 were computed from the formula

$$\left| \frac{\text{computed eigenvalue} - \text{exact eigenvalue}}{\text{exact eigenvalue}} \right|$$

Table 4 gives some information pertinent to using MAP to solve an ordinary eigenproblem - the core requirement and total charges for a given size of matrix may be of special interest.

TABLE 3

Maximum Relative Error as a Function of The Dimension n

n	Minimum Error ($\times 10^6$)	Maximum Error ($\times 10^6$)
20	.008370	1.220
40	.01109	18.72
60	.05949	5.476
80	.08430	7.687
100	.1042	8.472
120	.02919	10.91
140	.04019	14.07
160	.08919	12.75
180	.01490	16.68
200	.1490	19.29
220	.1042	16.26
240	.5004	20.29

TABLE 4

Information about The Computer Runs as a Function of The Dimension n of The Input Matrices M and K

Main features of the runs:

(1) Cholesky decomposition skipped (i.e., an ordinary real symmetric algebraic eigenproblem is being solved).
(2) n eigenvalues computed if n<50 and 50 eigenvalues computed otherwise
(3) three eigenvectors computed (and not transformed into vectors y)
(4) the only output was the printing of the eigenvalues and vectors
(5) one compilation (except for n = 20 and 40 in which cases there were four)

n	Core Requirements	CPU Time	Wall-Clock Time	Core Seconds	IO Count	Charge
20	32.6K	0:14	12:53	361	1,007	10
40	33.5K	0:17	7:56	341	901	9
60	44.8K	0:15	1:29	260	720	7
80	36.5K	0:21	4:45	315	761	9
100	38.6K	0:34	4:03	393	835	11
120	41.1K	0:44	2:48	530	935	14
140	43.9K	1:01	2:32	614	1,043	16
160	47.2K	1:24	8:44	1,033	1,167	26
180	50.9K	1:54	9:15	1,288	1,309	32
200	55.0K	2:27	3:18	1,645	1,466	41
220	59.5K	3:11	4:45	2,317	1,639	56
240	64.4K	4:04	5:07	3,195	1,828	74

Usage

While MAP has been used to solve problems as small as 5x5, potential users of MAP who have such small problems may find it inconvenient to have to store the matrix M in the manner required by MAP - the lower triangular part must be stored in a one-dimensional array by columns. Such users may wish to call the subroutines RESTOR that acts as an interface between programs which store a whole matrix in a two-dimensional array and programs which store only the lower triangular part of a matrix in a one-dimensional array by columns. Instructions for using RESTOR are given in the introductory commentary. of RESTOR which is listed in the first appendix.

The introductory commentary to the MAP subroutine itself is a user's guide to the MAP program package. (See the first appendix). MAP has been put in the Bellcomm Applications Program Library (#0218). The relocatable elements of the entire program package are on SYS*BLIB as MAP/BC, SGEIG/BC, SIREIG/BC, etc.

The author would appreciate being informed of any bugs in MAP, errors of accuracy or omission in the introductory commentaries, and anything awkward about using MAP.

Program Description

The MAP program package was written with the idea in mind that it should be able to handle matrices on the order of 300x300. Thus growth of computational error and storage were the primary concerns. The sponsor, S. N. Hou, dealt with the former problem by choosing mathematical methods which are numerically stable and which use a minimum of arithmetic operations. The mathematical proofs of the numerical stability of these methods, however, assume that all accumulations are done exactly or as nearly so as possible.[17, 18] All accumulations in MAP are therefore done in double precision to approximate this condition. The latter problem was handled mainly by taking advantage of the symmetry in the matrices M, K and P and of the fact that L and L^{-1} are lower triangular matrices - only the lower triangular part of a matrix is kept in core. (I would like to emphasize that the programs assume these matrices are symmetric - no check for symmetry is made). Hence the storage requirement for a matrix is reduced from n^2 to $n(n+1)/2$, i.e., by a factor of about two for large n. Nevertheless, for a 300x300 matrix, this entails allocating an array dimensioned at least 45,150. Clearly in a 65K core environment there can be only one such array. With the sole exception of the matrix product $P = L^{-1}KL^{-T}$, the mathematical methods lend themselves to this approach. The computation of P, however, makes considerable use of external storage areas. The storage problem is also handled by computing the eigenvectors and storing them on a mass storage device one at a time so that only a one-dimensional

array dimensioned n is needed to deal with them. The storage problem can also be alleviated some by overlaying the MAP program package. Bearing this in mind, the program was segmented as much as possible. At present, there are twenty subroutines in the package. Table 5 lists them and gives a very brief description of each, and Figure 1 shows how they are related.

In order that each subroutine in the package can be easily understood and changed by someone other than the author, there is a great deal of commentary in each of the twenty programs comprising the MAP package. The commentary is of two kinds: introductory and intercode. The former explains (among other things) the purpose, method, and assumptions of the program and defines the input and output for it. A flowchart can be constructed from the latter.

While the idea that MAP should be able to handle matrices on the order of 300x300 was the main consideration that determined the final structure of the MAP program package, it was not the only one. A second factor was the recognition that the mathematical methods are of general interest. This led to making each mathematical method a separate subroutine with commentary directed toward the general user (rather than a structural engineer), led to writing the control subroutines SGEIG and SIREIG, and influenced the argument lists of L and LINVRS.

The relationships between all the subroutines in the MAP program package are illustrated in Figure 1. The subroutines MAP, INPUT, and OUTPUT are intended for structural engineers (though they can be used by others provided the terminology of structural problems is not too inconvenient). SGEIG and all the subroutines below it are general purpose mathematical programs that solve a variety of problems in linear algebra which involve real symmetric matrices.

SGEIG and SIREIG differ from the other general purpose routines in that they are control programs for the solution of a generalized real symmetric positive definite algebraic eigenproblem and an ordinary real symmetric algebraic eigenproblem, respectively. The main purposes of, say, SIREIG are to coordinate the argument lists of the routines which solve the ordinary eigenproblem for the user, and potentially to enable the user to have at his disposal via one CALL statement all the useful eigenvalue and eigenvector programs at Bellcomm which solve a (possibly) large real symmetric eigenproblem. These two routines are also written in a way that simplifies the implementation of new methods. In order to incorporate a new eigenvalue or eigenvector method into SIREIG, for instance, all that is needed is to include the appropriate CALL statement (or statements) in SIREIG, add one component to a computed GO TO statement in SIREIG,

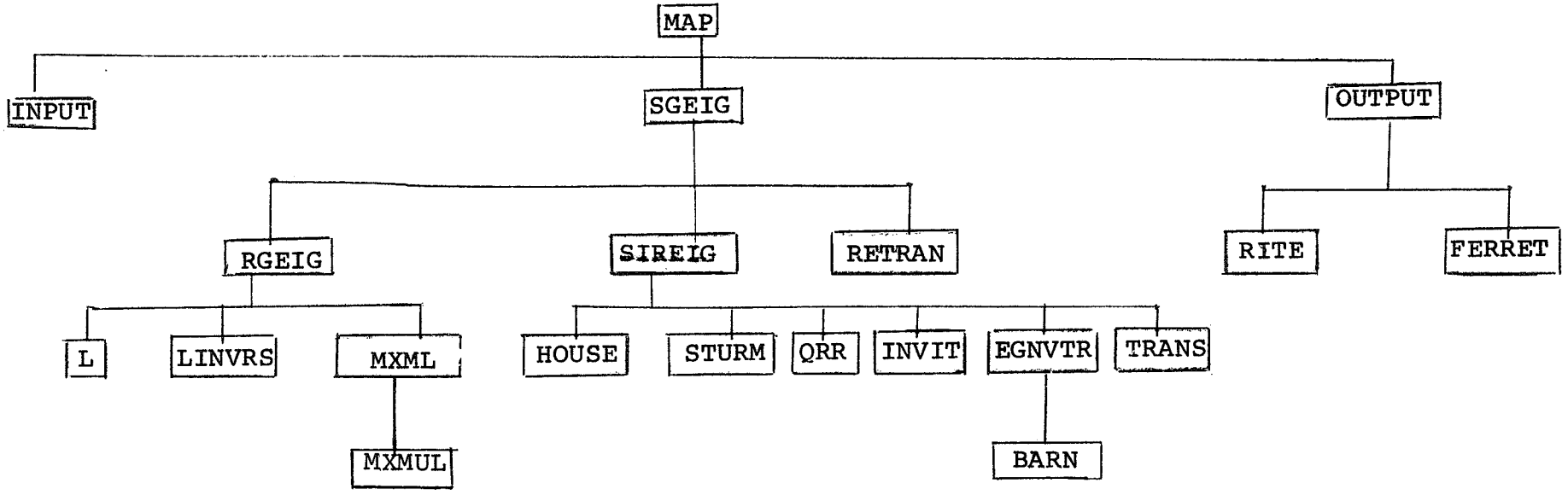
TABLE 5

Subroutines Comprising the MAP Program Package

BARN	N calls to BARN will generate n random numbers.
EGNVTR	Computes orthogonal eigenvectors by a combination of inverse iteration and Gram-Schmidt orthogonalization.
FERRET	Punches cards and does some storing onto tape.
HOUSE	Householder tridiagonalization.
INPUT	Reads the input matrices.
INVIT	Computes the eigenvectors x by back substitution and inverse iteration.
L	Computes the lower triangular matrix L (and can compute the determinant of M).
LINVRS	Computes the inverse of the matrix L.
MAP	The main subroutine for computing frequencies and modes.
MXML	Control subroutine for the computation of P.
MXMUL	Computes the product of two large matrices.
OUTPUT	A control subroutine for all the outputting.
QRR	Computes only eigenvalues by a QR method.
RETRAN	Retransforms the vectors x of P into the vectors y, i.e. multiplies a vector on the left by an upper triangular matrix.
RGEIG	Reduces the generalized algebraic eigenproblem to an ordinary one by the method of Cholesky.
RITE	Does most of the actual printing and storing onto tape.
SGEIG	Solves the linear real symmetric generalized algebraic eigenproblem.
SIREIG	Solves the real symmetric algebraic eigenproblem.
STURM	Computes eigenvalues by a Sturm sequence bisection method.
TRANS	Transforms vectors of the tridiagonal matrix computed by EGNVTR into those of P.

FIGURE 1

Structure of The MAP Program Package



and let one argument in the SIREIG argument list take on a new value. At present SIREIG has two eigenvalue methods and two eigenvector methods. Also at present SGEIG has only way of solving the generalized problem - reduction to an ordinary eigenproblem by the method of Cholesky. It frequently happens in structural problems that the input matrices M and K are strongly banded. It would be desirable to be able to take advantage of that property in order to reduce the number of arithmetic operations and to reduce the core requirement of MAP. A method which takes this approach could be implemented and easily incorporated into MAP (or, to be strictly accurate, into SGEIG).[19]

The subroutine L can be used to compute the determinant of a real symmetric positive definite matrix and to determine whether a given real symmetric matrix is positive definite. The programs L and LINVRS can also be the basis for a program which inverts real symmetric positive definite matrices or solves a system of linear algebraic equations which has a real symmetric positive definite matrix of coefficients.* The use of the subroutines L and LINVRS (i.e. of Cholesky decomposition) can be recommended over Gauss-Jordan elimination and Gaussian elimination with partial pivoting programs as the bound for the computational error for Cholesky decomposition is as low as can be reasonably expected of any method,[20] they take advantage of symmetry, and they require no row interchanges. L can also take advantage of any band form in the matrix M.

It was the intention of the author in writing MAP that this program should be as useful in solving a 5x5 problem as it is in solving a 300x300 problem. In order to realize this goal, the core requirement of MAP had to depend on the size of the input matrices. All arrays whose dimensions are functions of the dimensions of the input matrices M and K are therefore passed through the argument list of MAP, so that they are dimensioned by the user rather than by MAP.

Thus far nothing has been said about what the maximum dimension of the M and K matrices which the MAP program package can handle is, but rather only indicated that it should be about 300. Actually, there is nothing in the MAP program package, as such, which restricts the dimension of the input matrices. Not only does passing all arrays through the argument list of MAP make the package practical for small problems, but it also means that, in this sense, MAP can work with arbitrarily large matrices. There are, however, three factors which serve to limit the dimension of the input matrices. They are (1) computational error increases with the dimension, (2) the requirement that the whole lower triangular part of one matrix (along with everything else) must fit in core, and (3) the possible existence of a maximum running time, charge, or whatever at a computer installation. The answer to the first depends on how much precision is wanted and

---

*Double precision versions DL and DLINVR of L and LINVRS, respectively, have been made by Mrs. S. B. Watson and have been put in the Bellcomm Applications Program Library (BAPL)

on whether the user has (and wishes to use) a routine for refining the values of the eigenvalues. The answer to the second, of course, depends on how much core is available. With a 65K core and an overlay for MAP, the maximum dimension is slightly in excess of 300; without an overlay, the maximum dimension is between 240 and 260. The current version of the software at Bellcomm imposes a maximum charge per run of 2047. This restriction does not limit the size of the matrices which MAP can handle.

Conclusions

One of the main concerns about a modal analysis program for large matrices was the possibility of computational error becoming so large that the computed values of the eigenvalues and vectors would be rendered meaningless. The results of the simple beam runs of Example 3 confirmed that computational error is substantial when using large matrices. In the largest case run, that is, the 240x240 case, there were one and sometimes two digits of accuracy in the nonzero eigenvalues. The two zero eigenvalues, however, had computed values of 0.5768 and 1.2724. In this particular case, the computed answers still had practical value. It should be pointed out that the input matrices M and K for the simple beam runs are probably very good approximations. If a user of MAP made a run using input matrices which were not as accurate as the corresponding ones in the simple beam runs, there is reason for him to expect the precision in his answers to be any better, and perhaps not even as good, as those in the corresponding simple beam run. Whenever computed answers result from many operations involving large matrices, therefore, it would seem wise to pay more than passing attention to the question of how much, if any, precision remains in the answers.

Donald L. Mather

1015-DLM-sms

D. L. Mather

References

1. Hou, S. N., Large Degree-of-Freedom Modal Analysis Program for Structural Dynamics, Bellcomm Memorandum for File, B70 05043, May 19, 1970.

2. Faddeeva, V. N., Computational Methods of Linear Algebra, Dover, 1959, pp.81-5, (Note that there is an error in the theory on p. 81 - an arbitrary real symmetric matrix M cannot necessarily be factored into the form $M = LL^T$. Consider, for instance, the matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. The property of positive definitness is also needed. See the next reference.)

3. Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarendon Press, 1965, pp. 229-30.

4. Wilkinson, J. H., ibid, pp. 229-30.

5. Noble, Ben, Applied Linear Algebra, Prentice-Hall, 1969, pp. 328-32. (This reference includes a geometric interpretation of Householder's method.)

6. Wilkinson, J. H., "Householder's Method for the Solution of the Algebraic Eigenproblem," Computer J., 1960 (3), pp. 23-7.

7. Wilkinson, J. H., 1965, ibid, pp. 299-307.

8. Wilkinson, J. H., "The Calculation of the Eigenvectors of Codiagonal Matrices, Computer J., 1958 (2), pp. 90-6.

9. Thuraisamy, V., The Complete Eigenvalue Problem for General Real Matrices, Bellcomm Technical Memorandum, TM-70-1033-7, June 16, 1970.

10. Singers, R. R., EGNVEC - Eigenvector Subroutine, Bellcomm Memorandum for File, B70 08009, August 5, 1970.

11. Vandergraft, J. S., Small Eigenvalues and Ill-Conditioned Matrices, Bellcomm Memorandum for File, B70 06038, June 11, 1970, pp. 3-4.

12. Wilkinson, J. H., 1965, ibid, p. 307.

13. Young, Dana and Felgar, Robert P., Jr., Tables of Characteristic Functions Representing Normal Modes of Vibrations of a Beam, University of Texas, #4913, July 1, 1949. (Note: this is the so-called "Texas Tables.")

References (Continued)

14. Peters, G. and Wilkinson, J. H., "Eigenvalues of Ax = λBx with Band Symmetric A and B," Computer J., 1969 (4), p. 400.

15. Peters, G. and Wilkinson, J. H., ibid, p. 401.

16. Gregory, Robert T. and Karney, David L., A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, 1969, p. 74.

17. Wilkinson, J. H., 1965, ibid, pp. 231-2, 236 and 247.

18. Ortega, J. M., "An Error Analysis of Householder's Method for the Symmetric Eigenvalue Problem," Numer. Math., 1963 (5), pp. 211-25.

19. Peters, G. and Wilkinson, J. H. ibid, pp. 398-404.

20. Wilkinson, J. H. 1965, ibid, pp. 231-4.

APPENDIX 1

The Users' Guide to MAP and a Listing of RESTOR

```
C  TITLE      MAP - A MODAL ANALYSIS PROGRAM
C
C
C  PROGRAMMER D. L. MATHER
C
C  SPONSOR     S. N. HOU
C
C  DATE        SEPTEMBER 1970
C
C  PURPOSE     TO FIND SOME OR ALL OF THE FREQUENCIES J (IN CPS) AND SOME
C              OR ALL OF THE MODE SHAPES Y OF ANY LINEAR DISCRETE SYSTEM
C              WHICH IS GOVERNED BY A GENERALIZED EIGENVALUE EQUATION OF
C              THE FORM
C
C                                   KY = JMY,
C
C              WHERE (1) M AND K ARE REAL AND SYMMETRIC MATRICES, (2) M
C              IS POSITIVE DEFINITE, (3) M IS NOT ILL-CONDITIONED WITH
C              RESPECT TO INVERSION, AND (4) M AND K ARE POSSIBLY, BUT NOT
C              NECESSARILY, LARGE MATRICES
C
C  METHOD      (1) THE GENERALIZED ALGEBRAIC EIGENPROBLEM IS REDUCED TO
C              AN ORDINARY ALGEBRAIC EIGENPROBLEM BY THE METHOD OF
C              CHOLESKY (I.E. THE SQUARE ROOT METHOD). THIS TECHNIQUE
C              IS USED TO COMPUTE A LOWER TRIANGULAR MATRIX L SUCH THAT
C
C                                   M = L*(L TRANSPOSE)
C
C              (IF M IS POSITIVE DEFINITE, SUCH A MATRIX NOT ONLY EXISTS
C              BUT IS ALSO REAL.) THE INITIAL EQUATION CAN NOW BE
C              WRITTEN AS
C
C                                   KY = JL*(L TRANSPOSE)Y
C
C              THEN (LINVRS*K*(LINVRS TRANSPOSE)) * ((L TRANSPOSE)*Y) =
C                                   J(LINVRS*L) * ((L TRANSPOSE)*Y), OR
C
C                                   PX = JX,
C
C              FOR P = LINVRS*K*(LINVRS TRANSPOSE) AND X = (L TRANSPOSE)*Y
C              NOTE THAT THE EIGENVALUES OF P ARE THE SAME AS THOSE IN THE
C              ORIGINAL PROBLEM.
C              (2) THE MATRIX P IS REDUCED TO A REAL SYMMETRIC TRIDIAGONAL
C              MATRIX BY THE METHOD OF HOUSEHOLDER
C              (3) THE EIGENVALUES OF THE TRIDIAGONAL MATRIX (WHICH ARE THE
C              SAME AS THOSE OF P) CAN BE COMPUTED IN ANY OF THE
C              FOLLOWING WAYS
C                  (A) A STURM SEQUENCE BISECTION METHOD TO COMPUTE SOME OR
C                      ALL OF THE EIGENVALUES STARTING WITH THE SMALLEST ONE
C                  (B) A QR TRANSFORMATION METHOD TO COMPUTE ALL (AND ONLY)
C                      THE EIGENVALUES
C              (4) THE EIGENVECTORS OF THE TRIDIAGONAL MATRICES CAN BE COMPUTED
C              IN ONE OF THE FOLLOWING WAYS (NOTE: THE COMPUTATION OF THE
C              VECTORS CAN BE SKIPPED IF ONLY EIGENVALUES ARE WANTED)
C              (A) AN INVERSE ITERATION METHOD TO COMPUTE THE VECTORS
```

```
C                ASSOCIATED WITH SOME OR ALL OF THE COMPUTED EIGENVALUES
C                BEGINNING WITH THE SMALLEST EIGENVALUE. (NOTE: WHILE
C                THIS METHOD COMPUTES ORTHOGONAL VECTORS WHEN ALL THE
C                EIGENVALUES ARE DISTINCT, IT WILL NOT COMPUTE EVEN
C                LINEARLY INDEPENDENT VECTORS FOR A MULTIPLE ROOT.)  THE
C                NUMBER OF ITERATIONS IS DETERMINED BY THE USER.  ONE
C                ITERATION IS EQUIVALENT TO THE METHOD OF BACK
C                SUBSTITUTION.  AS A RULE, THE MORE ACCURATE THAT THE
C                EIGENVALUES ARE THE FEWER ITERATIONS THAT WILL BE
C                NEEDED.  ONE TO THREE ITERATIONS ARE, IN GENERAL, ARE
C                SUFFICIENT.
C           (4)  A COMBINATION OF INVERSE ITERATION (USING TWO
C                ITERATIONS) AND GRAM-SCHMIDT ORTHOGONALIZATION TO
C                COMPUTE ORTHOGONAL VECTORS FOR SOME OR ALL OF THE
C                COMPUTED EIGENVALUES STARTING WITH THE SMALLEST
C                EIGENVALUE.  (NOTE: THIS METHOD WILL, IN GENERAL, TAKE
C                MORE TIME AND REQUIRE 6N MORE LOCATIONS FOR ARRAYS THAN
C                THE PRECEDING METHOD.)
C           (5)  THE VECTORS OF THE TRIDIAGONAL MATRIX ARE TRANSFORMED INTO
C                THOSE OF P (NOTE: THIS TRANSFORMATION IS SKIPPED IF NO
C                VECTORS ARE COMPUTED AND CAN BE SKIPPED IF IT IS THE VECTORS
C                OF THE TRIDIAGONAL MATRIX THAT ARE WANTED)
C           (6)  THE VECTORS X OF P CAN BE TRANSFORMED INTO THE VECTORS Y
C                (NOTE: THIS TRANSFORMATION IS SKIPPED IF NO VECTORS ARE
C                COMPUTED OR THE PRECEDING TRANSFORMATION IS SKIPPED, AND
C                CAN BE SKIPPED IF IT IS THE VECTORS OF P THAT ARE WANTED)
C
C NOTES     (1)  WHEN THERE ARE SEVERAL MAGNITUDES OF DIFFERENCE BETWEEN
C                THE EIGENVALUES OF SMALLEST AND LARGEST MAGNITUDES,
C                THE SMALLEST EIGENVALUES WILL NOT, IN GENERAL, HAVE A LOW
C                RELATIVE ERROR (THIS FACT CAN BE EASILY ILLUSTRATED FOR
C                A STURM SEQUENCE METHOD - SEE WILKINSON, P. 307)
C           (2)  IF ONLY A FEW OF THE EIGENVALUES ARE WANTED, A STURM
C                SEQUENCE BISECTION AND INVERSE ITERATION COMBINATION WILL,
C                IN GENERAL, BE FASTER THAN A QR TRANSFORMATION METHOD OR ANY
C                COMBINATION OF METHODS WHICH INCLUDES A QR TRANSFORMATION
C           (3)  IF THE CALLING PROGRAM HAS STORED THE INPUT MATRIX IN A TWO
C                DIMENSIONAL ARRAY, THE USER MAY FIND IT CONVENIENT TO CALL
C                THE SUBROUTINE RESTOR THAT ACTS AS AN INTERFACE BETWEEN
C                PROGRAMS WHICH STORE A MATRIX IN A TWO DIMENSIONAL ARRAY AND
C                PROGRAMS WHICH STORE THE LOWER TRIANGULAR PART OF A MATRIX
C                BY COLUMNS IN A ONE DIMENSIONAL ARRAY BEFORE CALLING MAP
C
C ASSUMPTIONS OF THE PROGRAM (IN ADDITION TO THOSE OF THE METHOD)
C
C           (1)  SCRATCH OUT-OF-CORE STORAGE IS NEEDED IN SOME PARTS
C                OF THE PROGRAM (NOTE THIS ASSUMPTION DOES NOT IMPLY
C                THAT THE PROGRAM CANNOT, OR EVEN SHOULD NOT, BE USED
C                TO ANALYZE SMALL PROBLEMS, BUT RATHER THAT MORE
C                OUT-OF-CORE STORAGE WILL BE USED THAN THE STORAGE
C                REQUIREMENTS OF SMALL MATRICES JUSTIFY)
C           (2)  THE LOWER TRIANGULAR PART OF ONE NXN MATRIX FITS IN CORE
C
C SUBROUTINE STATEMENT
C
C           SUBROUTINE MAP(N,S,SS,III,ITITLE,METHOD,ICHECK,IMK,IUNIT,
C          .                   OP,OT,OPCH,OPLOT,FREQ,TABLE,C,$)
```

```
C
C  INPUT       *** BY THE ARGUMENT LIST ***
C
C               N          THE NUMBER OF DEGREES OF FREEDOM
C               S(.)       AN ARRAY TO HOLD THE LOWER TRIANGULAR PART OF
C                          SOME OF THE MATRICES AND MUST THEREFORE BE
C                          DIMENSIONED AT LEAST N*(N+1)/2.  THE LOWER
C                          TRIANGULAR PART OF THE MASS MATRIX CAN BE STORED
C                          IN THIS ARRAY BY COLUMNS AND PASSED THROUGH THE
C                          ARGUMENT LIST (IN WHICH CASE IMK(1) MUST BE
C                          NEGATIVE)
C               SS(.)      A WORKING ARRAY DIMENSIONED AT LEAST 9N IF
C                          METHOD(5)=2, AT LEAST 7N IF METHOD(4)=2, AND AT
C                          LEAST 5N OTHERWISE
C               III(.)     THIS IS A WORKING ARRAY WHICH IS USED IF
C                          EITHER OR BOTH OF THE MASS AND STIFFNESS
C                          ARE DEFINED BY THEIR NONZERO ELEMENTS AND SHOULD BE
C                          DIMENSIONED AT LEAST 2N.  IT IS ALSO A WORKING ARRAY IF
C                          METHOD(5)=2 IN WHICH CASE IT NEEDS TO BE DIMENSIONED AT
C                          LEAST 2N.  OTHERWISE, III IS A DUMMY INPUT AND NEED NOT
C                          BE DIMENSIONED AT ALL.
C               ITITLE(.)  AN ARRAY DIMENSIONED 40, CONTAINING SIX HOLLERITH
C                          CHARACTERS (INCLUDING BLANKS) PER COMPONENT FOR
C                          A TOTAL OF 240 CHARACTERS.  THE FIRST 120 CHARACTERS
C                          WILL BE PRINTED ON THE FIRST LINE OF PRINT AND
C                          THE REMAINING 120 WILL BE PRINTED ON THE SECOND
C                          LINE OF PRINT.  THE USER CAN THEREFORE USE THIS
C                          ARRAY TO IDENTIFY A RUN.  IF HE DOES NOT WISH TO
C                          USE THIS FEATURE, HE NEED NOT DIMENSION ITITLE
C                          AND INSTEAD SET AN (UNDIMENSIONED) DUMMY INTEGER
C                          VARIABLE EQUAL TO THE HOLLERITH CHARACTER * AND
C                          PASS THAT THROUGH THIS ARGUMENT.
C               METHOD(.)  AN ARRAY DIMENSIONED 10 WHICH DETERMINES WHICH
C                          METHODS WILL BE USED AND SOME RELATED INFORMATION
C               METHOD(1)  DETERMINES WHAT METHOD WILL BE USED TO REDUCE
C                          THE GENERALIZED ALGEBRAIC EIGENPROBLEM TO AN
C                          ORDINARY ALGEBRAIC EIGENPROBLEM
C                          = -1  AN ORDINARY ALGEBRAIC EIGENPROBLEM IS BEING
C                                SOLVED (I.E. SKIP THE CHOLESKY DECOMPOSITION OR
C                                ANY OTHER GENERALIZED ALGEBRAIC EIGENPROBLEM
C                                METHOD)
C                          = 1   THE CHOLESKY METHOD (I.E. THE SQUARE ROOT METHOD)
C                          = 2   (NOT IMPLEMENTED YET)
C               METHOD(2)  A REDUCTION IN THE NUMBER OF ARITHMETIC OPERATIONS
C                          IN THE COMPUTATION OF THE LOWER TRIANGULAR MATRIX L
C                          CAN BE REALIZED IF THE MASS MATRIX IS A BAND MATRIX.
C                          THIS INPUT IS EITHER THE DIMENSION OF M OR BAND WIDTH
C                          OF M AND THEREFORE CAN RANGE FROM 1 FOR A DIAGONAL
C                          MATRIX TO N FOR A FULL ONE (FOR THE PURPOSE OF THIS
C                          PROGRAM, BAND WIDTH = (FULL BAND WIDTH - 1)/2 + 1)
C               METHOD(3)  A DUMMY COMPONENT IF METHOD(1) IS NOT ONE
C                          = 0   THE DETERMINANT OF M WILL NOT BE COMPUTED
C                          = 1   IT WILL (NOTE: IF THE VALUE OF THE DETERMINANT
C                                IS EXTREMELY SMALL OR EXTREMELY LARGE THE COMPUTER
C                                SYSTEM WILL PRINT A CHARACTERISTIC UNDERFLOW OR
C                                OVERFLOW MESSAGE, RESPECTIVELY)
C               METHOD(4)  DETERMINES WHICH METHOD EIGENVALUE METHOD WILL BE USED
```

```
C                    = 1  A STURM SEQUENCE BISECTION METHOD TO COMPUTED SOME
C                         OR ALL OF THE EIGENVALUES IN ASCENDING ORDER
C                    = 2  A QR TRANSFORMATION METHOD TO COMPUTE ALL (AND
C                         ONLY) THE EIGENVALUES
C          METHOD(5) DETERMINES WHICH METHOD WILL BE USED TO COMPUTE VECTORS
C                    OF THE TRIDIAGONAL MATRIX (NOTE: THIS IS A DUMMY
C                    INPUT IF METHOD(7)=0)
C                    = 1  AN INVERSE ITERATION METHOD TO COMPUTE VECTORS FOR
C                         SOME OR ALL OF THE COMPUTED EIGENVALUES STARTING
C                         WITH THE VECTOR ASSOCIATED WITH THE SMALLEST
C                         EIGENVALUE
C                    = 2  A COMBINATION OF INVERSE ITERATION AND GRAM-
C                         SCHMIDT ORTHOGONALIZATION TO COMPUTE VECTORS FOR
C                         SOME OR ALL OF THE COMPUTED EIGENVALUES STARTING
C                         WITH THE VECTOR ASSOCIATED WITH THE SMALLEST
C                         EIGENVALUE
C          METHOD(6) THIS IS A DUMMY INPUT IF METHOD(4)=2 AND OTHERWISE IS
C                    THE NUMBER OF EIGENVALUES TO BE COMPUTED STARTING WITH
C                    THE SMALLEST ONE (NOTE: AN EIGENVALUE OF MULTIPLICITY M
C                    IS COUNTED AS M EIGENVALUES)
C          METHOD(7) THIS IS THE NUMBER OF VECTORS OF THE TRIDIAGONAL MATRIX
C                    TO BE COMPUTED (NOTE: IF NO VECTORS ARE WANTED SET THIS
C                    INPUT EQUAL TO 0)
C          METHOD(8) THIS IS A DUMMY INPUT IF METHOD(5)=2 AND
C                    OTHERWISE IS THE NUMBER OF ITERATIONS THAT WILL BE USED
C                    IN THE INVERSE ITERATION METHOD (SEE THE ''METHOD''
C                    SECTION ABOVE FOR SUGGESTED CRITERIA FOR DETERMINING
C                    THE VALUE OF THIS INPUT)
C          METHOD(9) THIS IS A DUMMY INPUT IF METHOD(7)=0
C                    = 0  DO NOT TRANSFORM THE VECTORS OF THE TRIDIAGONAL
C                         MATRIX INTO THOSE OF P
C                    = 1  DO
C          METHOD(10)THIS IS A DUMMY INPUT IF METHOD(7)=0 OR METHOD(9)=1
C                    AND OTHERWISE THIS IS THE NUMBER OF VECTORS OF P THAT
C                    SHOULD BE TRANSFORMED INTO THE VECTORS Y OF THE
C                    ORIGINAL PROBLEM (NOTE: THIS CAN BE ZERO)
C          ICHECK(.) AN ARRAY DIMENSIONED AT LEAST 4 TO DETERMINE WHAT, IF
C                    ANY, AUXILIARY INFORMATION SHOULD BE PRINTED
C          ICHECK(1) = 0  THE ARGUMENT LISTS OF MAP, INPUT, SGEIG, OUTPUT,
C                         RGEIG, SIREIG, AND RETRAN WILL NOT BE PRINTED
C                    = 1  ONLY THE ARGUMENT LIST OF MAP WILL BE PRINTED
C                    = 2  ONLY THE ARGUMENT LISTS OF MAP, INPUT, SGEIG,
C                         AND OUTPUT WILL BE PRINTED
C                    = 3  ALL SEVEN ARGUMENT LISTS WILL BE PRINTED
C          ICHECK(2) = 0  THE MAIN DIAGONAL AND OFF-DIAGONAL OF THE
C                         TRIDIAGONAL MATRIX WILL NOT BE PRINTED
C                    = 1  THEY WILL
C          ICHECK(3) THIS IS A DUMMY INPUT IF METHOD(5) IS NOT ONE
C                    = 1  EACH ITERATE OF EACH OF THE VECTORS OF THE
C                         TRIDIAGONAL MATRIX WILL BE PRINTED
C                    = 2  EACH NORMALIZED VECTOR OF THE TRIDIAGONAL MATRIX
C                         WILL BE PRINTED
C                    = 3  BOTH OF THE ABOVE WILL BE PRINTED
C                    = 0  NONE OF THE ABOVE WILL BE PRINTED
C          ICHECK(4) THE NUMBER OF VECTORS OF THE TRIDIAGONAL MATRIX TO BE
C                    PRINTED BEGINNING WITH THE VECTOR ASSOCIATED WITH THE
C                    SMALLEST EIGENVALUE (NOTE: THIS CAN BE ZERO)
```

```
C          IMK(.)    AN ARRAY DIMENSIONED 2 TO DETERMINE THE FORMAT
C                    WITH WHICH THE INPUT MATRICES WILL BE READ
C          IMK(1)    FORMAT FOR THE MASS MATRIX (IF THE MASS MATRIX IS
C                    BEING PASSED THROUGH THE ARGUMENT LIST SET THIS
C                    COMPONENT EQUAL TO A NEGATIVE INTEGER - BEWARE NOT TO
C                    PASS THE MASS MATRIX THRU THE ARGUMENT LIST WHEN
C                    IMK(2)=1 AS THE ARRAY S IS USED AS A WORKING AREA THEN)
C                    = 1  NASTRAN FORMAT (ALL THE NONZERO ELEMENTS BY ROWS
C                         AND THEIR NASTRAN ROW AND COLUMN NUMBERS)
C                    = 2  FIRST RECORD OF IUNIT(1) CONSISTS OF TWO BINARY
C                         INTEGERS - A NUMBER TO IDENTIFY THE DATA
C                         FOR THE USER FOLLOWED BY THE NUMBER OF
C                         DEGREES OF FREEDOM. THE SECOND RECORD
C                         CONSISTS OF THE LOWER TRIANGULAR PART OF
C                         THE MASS MATRIX STORED IN BINARY BY COLUMNS.
C          IMK(2)    FORMAT FOR THE STIFFNESS MATRIX (MAKE THIS VARIABLE
C                    NEGATIVE IF THE STIFFNESS MATRIX HAS BEEN STORED ON
C                    IUNIT(6) IN THE PROPER WAY ALREADY)
C                    = 1  NASTRAN FORMAT
C                    = 2  THE WHOLE STIFFNESS MATRIX STORED IN BINARY
C                         BY ROWS BEGINNING WITH THE FIRST ROW ON THE
C                         FIRST RECORD (THIS IS PERHAPS THE BEST FORM
C                         AS IT IS THE FORM REQUIRED BY SGEIG)
C                    = 3  SAME AS FOR (2) EXCEPT THAT THE MASS AND
C                         STIFFNESS MATRICES ARE ON THE SAME MASS
C                         STORAGE UNIT AND IMK(1) = 2 (SO THAT THE
C                         FIRST ROW OF THE STIFFNESS MATRIX IS ON THE
C                         THIRD RECORD RATHER THAN ON THE FIRST)
C          IUNIT(.)  AN INTEGER ARRAY DIMENSIONED 15 TO DEFINE THE
C                    NUMBERS OF THE MASS STORAGE UNITS (OR AREAS)
C                    WHICH CONTAIN THE FOLLOWING DATA (NOTE: THE
C                    BELLCOMM CONVENTION FOR NUMBERS DENOTING MASS
C                    STORAGE UNITS IS THAT ANY INTEGER FROM 1 TO 40
C                    INCLUSIVE MAY BE USED EXCEPT 5, 6, 7, AND 30)
C          IUNIT(1)  THE MASS MATRIX IN ANY UNITS
C          IUNIT(2)  STIFFNESS MATRIX IN THE SAME UNITS AS THE MASS MATRIX
C          IUNIT(3)  A WORKING AREA USED BY NASTRN AND MXML
C          IUNIT(4)  A SECOND WORKING AREA FOR NASTRN.  THIS IS THEREFORE
C                    A DUMMY COMPONENT IF NASTRAN DATA ARE NOT USED
C          IUNIT(5)  THE LOWER TRIANGULAR PART OF THE MASS MATRIX IS
C                    STORED BY COLUMNS ON THE FIRST AND ONLY RECORD
C                    OF THIS UNIT IN BINARY IF THE MASS MATRIX NEEDS TO
C                    BE SAVED FOR OUTPUT BY MAP. IF IT IS NOT NEEDED FOR
C                    OUTPUT BUT THE USER WANTS IT SAVED ANYWAY, MAKE
C                    THIS INPUT POSITIVE
C          IUNIT(6)  THE WHOLE STIFFNESS MATRIX IN BINARY BY ROWS
C                    BEGINNING WITH THE FIRST ROW ON THE FIRST RECORD
C                    BY THE SUBROUTINE INPUT OR BY THE USER FOR MATMUL AND
C                    OUTPUT.  IN THE LATTER CASE MAKE IMK(2) NEGATIVE.
C          IUNIT(7)  THE LOWER TRIANGULAR PART OF THE MATRIX L IN
C                    BINARY BY COLUMNS ON THE FIRST (AND ONLY) RECORD
C                    ONTO CARDS, OR STORING ON TAPE).  THE USER CAN PREVENT
C                    IT FROM BEING SAVED BY MAKING THIS UNIT NUMBER NEGATIVE
C          IUNIT(8)  THE LOWER TRIANGULAR PART OF L INVERSE BY COLUMNS
C                    IN BINARY ON THE FIRST RECORD BY RGEIG FOR REIRAN.
C          IUNIT(9)  THE WHOLE L INVERSE IN BINARY ONE ROW PER RECORD
C                    BEGINNING WITH THE FIRST ROW ON THE FIRST RECORD
```

```
C                 BY RGEIG FOR MXML AND OUTPUT
C       IUNIT(10) THE WHOLE L INVERSE TRANSPOSE IN BINARY ONE ROW PER
C                 RECORD BEGINNING WITH THE FIRST ROW ON THE FIRST RECORD
C                 BY RGEIG FOR OUTPUT.  THE USER CAN PREVENT IT FROM
C                 BEING SAVED BY MAKING THE UNIT NUMBER NEGATIVE.
C       IUNIT(11) THE LOWER TRIANGULAR PART OF THE MATRIX P BY COLUMNS
C                 IN BINARY ON THE FIRST RECORD BY RGEIG FOR OUTPUT
C       IUNIT(12) EIGENVECTORS OF P IN BINARY ONE PER RECORD BEGINNING
C                 WITH THE EIGENVECTOR WHICH IS ASSOCIATED WITH THE
C                 SMALLEST EIGENVALUE ON THE FIRST RECORD BY INVIT
C                 AND TRANS FOR RETRAN AND OUTPUT UNLESS BOTH METHOD(5)=1
C                 AND METHOD(7)=0
C       IUNIT(13) THE VECTORS OF THE TRIDIAGONAL MATRIX IN BINARY ONE PER
C                 RECORD BEGINNING WITH THE VECTOR ASSOCIATED WITH THE
C                 SMALLEST EIGENVALUE ON THE FIRST RECORD IF METHOD(5)=1
C                 AND METHOD(9)=0, OR METHOD(5)=2 FOR TRANS OR PRINTING
C                 THEM (ICHECK(4)=1)
C       IUNIT(14) A WORKING AREA FOR THE QR EIGENVALUE ROUTINE IF
C                 METHOD(4)=2 AND A DUMMY COMPONENT OTHERWISE
C       IUNIT(15) THE METHOD(10) MODE SHAPES IN BINARY ONE PER RECORD
C                 BEGINNING WIH THE MODE SHAPE WHICH IS ASSOCIATED WITH
C                 THE SMALLEST FREQUENCY ON THE FIRST RECORD BY RETRAN
C                 FOR OUTPUT
C       OP(.)     AN ARRAY DIMENSIONED 6 TO DETERMINE WHICH, IF
C                 ANY, OF THE MATRICES USED IN THIS PROGRAM PACKAGE
C                 AND WHICH OF THE RESULTS OF THE EIGENPROBLEMS
C                 WILL BE PRINTED
C       OP(1)     DETERMINES WHICH, IF ANY, OF THE MATRICES M, K,
C                 AND P WILL BE PRINTED
C                 = 0  NONE
C                 = 1  K
C                 = 2  K AND M
C                 = 3  K, M AND P
C                 = 4  M
C                 = 5  M AND P
C                 = 6  P
C       OP(2)     THIS IS A DUMMY INPUT IF NASTRAN DATA WERE NOT USED
C                 = 0  THE NASTRAN NUMBERS WILL NOT BE PRINTED
C                 = 1  THEY WILL
C       OP(3)     DETERMINES WHICH, IF ANY, OF THE MATRICES L,
C                 L INVERSE AND L INVERSE TRANSPOSE WILL BE PRINTED
C                 = 0  NONE
C                 = 1  L
C                 = 2  L AND L INVERSE
C                 = 3  ALL THREE
C                 = 4  L INVERSE
C                 = 5  L INVERSE AND L INVERSE TRANSPOSE
C                 = 6  L INVERSE TRANSPOSE
C       OP(4)     DETERMINES WHICH, IF ANY, OF THE RESULTS OF THE
C                 EIGENPROBLEMS WILL BE PRINTED
C                 = 0  NONE
C                 = 1  FREQUENCIES IN (RAD/SEC)**2 AND EIGENVECTORS OF P
C                 = 2  FREQUENCIES IN (RAD/SEC)**2 AND MODE SHAPES
C                 = 3  FREQUENCIES IN CPS AND THE VECTORS OF P
C                 = 4  FREQUENCIES IN CPS AND THE MODE SHAPES
C                 = 5  ONLY THE FREQUENCIES IN (RAD/SEC)**2
C                 = 6  ONLY THE FREQUENCIES IN CPS
```

```
C          OP(5)     DETERMINES WHAT PART OF EACH OF THE MATRICES DEFINED
C                    BY OP(1) AND OP(3) WILL BE PRINTED.  (THIS IS THEREFORE
C                    A DUMMY COMPONENT IF OP(1) = OP(3) =0)
C                    = 0  THE WHOLE MATRIX WILL BE PRINTED
C                    = 1  ONLY THE LOWER TRIANGULAR PART OF THE MATRIX (OR
C                         THE UPPER TRIANGULAR PART IN THE CASE OF L
C                         INVERSE TRANSPOSE)
C          OP(6)     DETERMINES THE FORMAT WITH WHICH THE RESULTS OF
C                    THE EIGENPROBLEMS WILL BE PRINTED (THIS IS THEREFORE A
C                    DUMMY COMPONENT IF OP(4) = 0)
C                    = 1  EACH FREQUENCY AND ITS MODE SHAPE, AND THEN A
C                         LIST OF ALL THE FREQUENCIES COMPUTED
C                    = 2  (NOT IMPLEMENTED YET)
C          OT(.)     AN ARRAY DIMENSIONED 6 TO DETERMINE WHICH, IF
C                    ANY, MATRICES AND WHICH OF THE RESULTS, IF ANY,
C                    OF THE EIGENPROBLEMS WILL BE STORED ON TAPE
C          OT(1)     DETERMINES WHICH, IF ANY, OF THE MATRICES M, K,
C                    AND P WILL BE PUT ON TAPE
C                    = 0  NONE
C                    = 1  K
C                    = 2  K AND M
C                    = 3  K, M AND P
C                    = 4  M
C                    = 5  M AND P
C                    = 6  P
C          OT(2)     THIS IS A DUMMY INPUT IF NASTRAN DATA WERE NOT USED
C                    = 0  THE NASTRAN NUMBERS WILL NOT BE STORED ON TAPE
C                    = 1  THEY WILL
C          OT(3)     DETERMINES WHICH, IF ANY, OF THE MATRICES L,
C                    L INVERSE AND L INVERSE TRANSPOSE WILL BE PRINTED
C                    = 0  NONE
C                    = 1  L
C                    = 2  L AND L INVERSE
C                    = 3  ALL THREE
C                    = 4  L INVERSE
C                    = 5  L INVERSE AND L INVERSE TRANSPOSE
C                    = 6  L INVERSE TRANSPOSE
C          OT(4)     DETERMINES WHICH, IF ANY, OF THE RESULTS OF THE
C                    EIGENPROBLEMS WILL BE PUT ON TAPE
C                    = 0  NONE
C                    = 1  FREQUENCIES IN (RAD/SEC)**2 AND EIGENVECTORS OF P
C                    = 2  FREQUENCIES IN (RAD/SEC)**2 AND MODE SHAPES Y
C                    = 3  FREQUENCIES IN CPS AND EIGENVECTORS X OF P
C                    = 4  FREQUENCIES IN CPS AND MODE SHAPES Y
C                    = 5  ONLY THE FREQUENCIES IN (RAD/SEC)**2
C                    = 6  ONLY THE FREQUENCIES IN CPS
C          OT(5)     DETERMINES HOW THE MATRICES WILL BE STORED
C                    = 1  THE WHOLE MATRIX IN BINARY ONE ROW PER RECORD
C                    = 2  THE LOWER TRIANGULAR PART BY COLUMNS (OR THE
C                         BY ROWS IN THE CASE OF L INVERSE TRANSPOSE) IN
C                         BINARY ON ONE RECORD
C                    = 3  ALL AND ONLY THE NONZERO ELEMENTS AND THEIR
C                         MATRIX ROW AND COLUMN NUMBERS IN BINARY BY ROWS
C                         WITH THREE ELEMENTS PER RECORD.  THE WRITE
C                         STATEMENT IS OF THE FORM
C
C                         WRITE(IOT6) (IROW(I),JCOL(I),ELT(I),I=1,3)
```

```
C
C          OT(6)      THE NUMBER OF THE UNIT TO WHICH THE OUTPUT TAPE IS
C                     ASSIGNED
C          OPCH(.)    AN INTEGER ARRAY DIMENSIONED 3 TO DETERMINE WHAT
C                     INFORMATION WILL BE PUNCHED ONTO CARDS
C          OPCH(1)    DETERMINES WHICH, IF ANY, MATRICES WILL BE PUNCHED
C                     ONTO CARDS (ONLY THE NONZERO ELEMENTS WILL BE PUNCHED)
C                     = 0  NONE
C                     = 1  K
C                     = 2  K AND M
C                     = 3  K, M AND P
C                     = 4  M
C                     = 5  M AND P
C                     = 6  P
C          OPCH(2)    DETERMINES WHICH, IF ANY, OF THE MATRICES L,
C                     L INVERSE, AND L INVERSE TRANSPOSE WILL BE PUNCHED
C                     = 0  NONE
C                     = 1  L
C                     = 2  L AND LINVERSE
C                     = 3  ALL THREE
C                     = 4  L INVERSE
C                     = 5  L INVERSE AND L INVERSE TRANSPOSE
C                     = 6  L INVERSE TRANSPOSE
C          OPCH(3)    DETERMINES WHICH, IF ANY, OF THE RESULTS OF THE
C                     EIGENPROBLEMS WILL BE PUNCHED
C                     = 0  NONE
C                     = 1  FREQUENCIES IN (RAD/SEC)**2 AND EIGENVECTORS OF P
C                     = 2  FREQUENCIES IN (RAD/SEC)**2 AND MODE SHAPES Y
C                     = 3  FREQUENCIES IN CPS AND EIGENVECTORS X OF P
C                     = 4  FREQUENCIES IN CPS AND MODE SHAPES Y
C                     = 5  ONLY THE FREQUENCIES IN (RAD/SEC)**2
C                     = 6  ONLY THE FREQUENCIES IN CPS
C          OPLOT(.)   (NOT IMPLEMENTED YET)
C
C          *** BY MASS STORAGE DEVICE OR CARDS ***
C
C          THE STIFFNESS MATRIX, POSSIBLY THE MASS MATRIX, AND WHATEVER
C          INFORMATION, IF ANY, IS REQUIRED BY IMK(1) AND IMK(2)
C
C  OUTPUT  *** BY THE ARGUMENT LIST ***
C
C          SS(.)      THE MAIN DIAGONAL OF THE TRIDIAGONAL MATRIX IS IN
C                     SS(1),SS(2),...,SS(N) AND THE OFF-DIAGONAL IS IN
C                     SS(N+2),SS(N+3),...,SS(2N)
C          FREQ(.)    AN ARRAY TO HOLD THE FREQUENCIES IN CPS IN ASCENDING
C                     ORDER AND MUST THEREFORE BE DIMENSIONED AT LEAST N
C                     (EVEN IF FEWER THAN N FREQUENCIES WERE COMPUTED)
C          TABLE(.)   ALL THE NASTRAN NUMBERS IN ASCENDING ORDER IF
C                     AND ONLY IF NASTRAN DATA WERE USED (I.E. IF
C                     IMK(1) = 1 AND IMK(2) = 1)
C          C(.)       AN ARRAY DIMENSIONED 5 TO HOLD CERTAIN INDICATORS
C                     OF THE ACCURACY OF THE RESULTS
C          C(1)       THE DETERMINANT OF M IF METHOD(3)=1 AND A DUMMY
C                     COMPONENT OTHERWISE
C          C(2)       THE TRACE OF P IN (RAD/SEC)**2 (NOTE: C(2),...,C(5) ARE
C                     COMPUTED BY THE SUBROUTINE SIREIG WHICH DENOTES THE
C                     MATRIX P WITH THE LETTER S)
```

```
C       C(3)    MAX(ABS(P(I,I)))/MIN(ABS(P(I,I))), PROVIDED THAT THE
C               DENOMINATOR IS NOT ZERO
C       C(4)    THE SUM OF ALL THE COMPUTED FREQUENCIES IN (RAD/SEC)**2
C       C(5)    MAX(ABS(J(I)))/MIN(ABS(J(I))), PROVIDED THAT THE
C               DENOMINATOR IS NOT ZERO, WHERE I RANGES FROM 1 TO THE
C               NUMBER OF FREQUENCIES COMPUTED
C       $I      CONTROL WILL BE TRANSFERRED FROM THIS SUBROUTINE
C               TO THAT STATEMENT IN THE CALLING PROGRAM NUMBERED
C               I IF ANY OF THE FOLLOWING ERRORS IS DETECTED:
C                 1. AN ILLEGAL VALUE FOR ONE OF THE INPUT ARGUMENTS
C                    IN THE MAP SUBROUTINE STATEMENT
C                 2. THE DIMENSIONS OF M AND K ARE DIFFERENT
C                    (THIS CHECK IS MADE WHEN THE MATRICES ARE
C                    DEFINED BY THEIR NONZERO ELEMENTS TO INSURE
C                    THAT EACH ROW HAS AT LEAST ONE NONZERO ELEMENT
C                 3. AN IMAGINARY NUMBER APPEARS IN THE MATRIX L
C
C       *** BY THE PRINTED WORD ***
C
C         1. INFORMATION IDENTIFYING THIS RUN IF ITITLE(1) (OR A
C            DUMMY INTEGER VARIABLE IN ITS PLACE) IS NOT EQUAL TO '*'
C         2. THE ARGUMENT LIST OF MAP IF ICHECK(1) = 1,2,3, THE
C            ARGUMENT LISTS OF MAP, INPUT, SGEIG, AND OUTPUT IF
C            ICHECK(1) = 2,3, AND THE ARGUMENT LISTS OF ALL THESE PLUS
C            THOSE OF RGEIG, SIREIG, AND RETRAN IF ICHECK(1) = 3
C         3. THE TRACE OF P IN (RAD/SEC)**2
C         4. THE RATIO OF THE MAIN DIAGONAL ELEMENT OF P OF GREATEST
C            MAGNITUDE TO THAT OF SMALLEST MAGNITUDE (PROVIDED THAT THE
C            LATTER IS NOT ZERO)
C         5. THE SUM OF ALL THE COMPUTED EIGENVALUES IN (RAD/SEC)**2
C         6. THE RATIO OF THE COMPUTED FREQUENCY OF LARGEST MAGNITUDE TO
C            THAT OF THE SMALLEST MAGNITUDE, PROVIDED THE LATTER IS NOT
C            ZERO (IN WHICH CASE A MESSAGE IS PRINTED)
C         8. THE MAIN DIAGONAL AND THE OFF-DIAGONAL OF THE REAL
C            SYMMETRIC TRIDIAGONAL MATRIX IN (RAD/SEC)**2 IF ICHECK(2)=1
C         8. ERROR MESSAGES WHEN THE FAILURE EXIT (I.E. THE ARGUMENT
C            $) IS USED
C         9. THE INFORMATION DETERMINED BY THE VALUES OF THE COMPONENTS
C            OF THE INPUT ARGUMENT OP AND ICHECK(2-4)
C
C       *** BY TAPE ***
C
C       THE INFORMATION DETERMINED BY THE VALUES OF THE COMPONENTS
C       OF THE INPUT ARGUMENT OT
C
C       *** BY CARDS ***
C
C       THE INFORMATION DETERMINED BY THE VALUES OF THE COMPONENTS
C       OF THE INPUT ARGUMENT OPCH
C
C       *** BY PLOTS ***
C
C       (NOT IMPLEMENTED YET)
C
C       *** BY SCRATCH MASS STORAGE ***
C
C       SEE THE DEFINITIONS OF IUNIT(5)-IUNIT(15)
```

```
C
C  SUBROUTINES NEEDED TO USE MAP
C
C        INPUT     READS THE MASS AND STIFFNESS MATRICES AND PUTS THEM
C                  INTO THE FORM REQUIRED BY SGEIG IF THEY ARE NOT ALREADY
C        SGEIG     SOLVES A GENERALIZED ALGEBRAIC EIGENPROBLEM
C        OUTPUT    DOES NEARLY ALL THE PRINTING, AND ALL THE PUNCHING
C                  ONTO CARDS AND STORING ON TAPE
C        NASTRN    COLLECTS ALL THE NASTRAN NUMBERS AND PUTS THEM
C                  INTO A TABLE IN ASCENDING ORDER
C        RGEIG     REDUCES A GENERALIZED ALGEBRAIC EIGENPROBLEM TO AN
C                  ORDINARY ONE
C        SIREIG    SOLVES THE REAL SYMMETRIC ALGEBRAIC EIGENPROBLEM
C                  IN ANY OF THREE WAYS
C        RETRAN    RETRANSFORMS THE EIGENVECTORS OF P INTO MODE SHAPES
C        NTRAN     UNIVAC ROUTINE USED BY NASTRN FOR STORING AND
C                  RETRIEVING INFORMATION ON MASS STORAGE UNITS
C        L         COMPUTES THE MATRIX L
C        LINVRS    COMPUTES L INVERSE
C        MXML      COMPUTES THE MATRIX P USING MXMUL
C        MXMUL     COMPUTES THE PRODUCT OF CERTAIN TWO MATRICES
C        HOUSE     TRIDIAGONALIZES THE MATRIX P
C        STURM     COMPUTES SOME (OR ALL) OF THE FREQUENCIES
C        INVIT     COMPUTES SOME (OR ALL) OF THE EIGENVECTORS
C        EGNVIR    COMPUTES VECTORS BY INVERSE ITERATION AND GRAM-SCHMIDT
C                  ORTHOGONALIZATION
C        RITE      DOES THE ACTUAL PRINTING AND SOME STORING ON TAPE
C                  OF MATRICES FOR THE SUBROUTINE OUTPUT
C        FERRET    SEARCHES A MATRIX FOR ITS NONZERO ELEMENTS
C        BARN      A RANDOM NUMBER GENERATOR
C
C  ****************** INTRODUCTORY FORTRAN STATEMENTS ********************
C
C  SUBROUTINE STATEMENT
C
      SUBROUTINE MAP(N,S,SS,III,ITITLE,METHOD,ICHECK,IMK,IUNIT,
     .               OP,OT,OPCH,OPLOT,FREQ,TABLE,C,$)
C
C  SPECIFICATION STATEMENTS
C
      REAL S(1),SS(1),FREQ(1),C(1)
      INTEGER III(1),METHOD(1),IMK(1),IUNIT(1),OP(1),OT(1),
     .OPCH(1),OPLOT(1),TABLE(1),UNIT(8),ITITLE(1),IFORM(6)
     .,ICHECK(1)
      DATA NOQID/'*'/
C
C  ********************** SOME PRELIMINARIES ****************************
C
C  PRINT THE RUN ID
C
      IF(ITITLE(1).EQ.NOQID) GOTO 10
      WRITE(6,1001) (ITITLE(I),I=1,40)
C
C  PRINT THE ARGUMENT LIST OF MAP
C
   10 IF(ICHECK(1).EQ.0) GOTO 20
      WRITE(6,1002) N,(METHOD(I),I=1,10),(ICHECK(I),I=1,4),
```

```
     .     (IMK(I),I=1,2),(IUNIT(I),I=1,15),(OP(I),I=1,6),(OT(I),I=1
     .     ,6),(OPCH(I),I=1,3),ITITLE(1)
           IF(ICHECK(1).NE.0) ICHECK(1)=ICHECK(1)-1
C
C  MISCELLANEOUS
C
 20        NARGS=17  @ NUMBER OF ARGUMENTS IN THE SUBROUTINE STATEMENT
C
C  CHECK THE REASONABLENESS OF SOME OF THE INPUT ARGUMENTS
C
           IF(N.GT.0) GOTO 22  @ N
           WRITE(6,2003)
           WRITE(6,2004) N
           RETURN NARGS
C
 22        IF(-1.LE.METHOD(1).AND.METHOD(1).LE.1) GOTO 23   @ METHOD AND N
           WRITE(6,2003)
           WRITE(6,2005) METHOD(1)
           RETURN NARGS
 23        IF(1.LE.METHOD(2).AND.METHOD(2).LE.N) GOTO 24
           WRITE(6,2003)
           WRITE(6,2006) N,METHOD(2)
           METHOD(2)=N
           WRITE(6,2007) METHOD(2)
 24        IF(1.LE.METHOD(4).AND.METHOD(4).LE.2) GOTO 25
           WRITE(6,2003)
           WRITE(6,2008) METHOD(4)
           RETURN NARGS
C
 25        IF(METHOD(4).NE.1) GOTO 26   @ METHOD(1),METHOD(6),N
           IF(1.LE.METHOD(6).AND.METHOD(6).LE.N) GOTO 26
           WRITE(6,2003)
           WRITE(6,2004) METHOD(1),METHOD(6),N
           RETURN NARGS
C
 26        IF(0.LE.METHOD(7).AND.METHOD(7).LE.N) GOTO 28
           WRITE(6,2003)
           WRITE(6,2010) N,METHOD(7)
           RETURN NARGS
C
 28        IF(IUNIT(5).GT.0) GOTO 30   @ DOES M NEED TO BE SAVED?
           IF(2.LE.OP(1).AND.OP(1).LT.6) GOTO 29
           IF(2.LE.OT(1).AND.OT(1).LT.6) GOTO 29
           IF(OPCH(1).LT.2.OR.OPCH(1).EQ.6) GOTO 31
 29        WRITE(6,2012) IUNIT(5),OP(1),OT(1),OPCH(1)
           RETURN NARGS
C
 30        MSAVE=1
           IF(IMK(2).GT.0) MSAVE=MSAVE+1
 31        IF(IUNIT(7).GT.0) GOTO 35   @ DOES L NEED TO BE SAVED?
           IF(1.LE.OP(3).AND.OP(3).LE.3) GOTO 32
           IF(1.LE.OT(3).AND.OT(3).LE.3) GOTO 32
           IF(OPCH(2).EQ.0.OR.3.LT.OPCH(2)) GOTO 35
 32        WRITE(6,2013) IUNIT(7),OP(3),OT(3),OPCH(2)
           RETURN NARGS
C
C  ********** PREPARE THE INPUT MATRICES FOR SGEIG AND OUTPUT **********
```

```
C
 35           IF(MSAVE.EQ.0) GOTO 50
 40           CALL INPUT(N,S,SS,IMK,IUNIT,ICHECK,ICASE,TABLE,$70)
C
C  ************ SOLVE THE GENERALIZED ALGEBRAIC EIGENPROBLEM ************
C
 50           IU=IUNIT(5)
              IUNIT(5)=IUNIT(3)
              CALL SGEIG(N,S,SS,III,FREQ,METHOD,ICHECK,IUNIT(5),C,$70)
              IUNIT(5)=IU
C
C  ************************ RECORD THE RESULTS ************************
C
C
C  CHANGE (RAD/SEC)**2 TO CPS
C
              TWOPI=6.2831852
              NFREQ=N
              IF(METHOD(4).EQ.1) NFREQ=METHOD(6)
              DO 60 I=1,NFREQ
              SS(2*N+I)=FREQ(I)
              IF(FREQ(I).GE.0.) GOTO 60
              WRITE(6,1003) I,FREQ(I)
 60           FREQ(I)=SQRT(ABS(FREQ(I)))/TWOPI
C
C  NOW RECORD WITHOUT DELAY
C
              IF(METHOD(1).NE.-1.AND.METHOD(3).GT.0) WRITE(6,2014) C(1)
              WRITE(6,1004) (C(I),I=2,3)
              WRITE(6,1005) NFREQ,C(4),C(5)
              IF(ICHECK(2).EQ.1) WRITE(6,1006) (I,SS(I),I=1,N)
              IF(ICHECK(2).EQ.1) WRITE(6,1007) (I,SS(2*N+I),I=2,N)
              IFORM(1)=1
              IFORM(2)=2
              IFORM(3)=2
              IFORM(4)=4
              IFORM(5)=1
              IFORM(6)=1
              UNIT(1)=IUNIT(6)
              UNIT(2)=IUNIT(5)
              UNIT(3)=IUNIT(7)
              UNIT(4)=IUNIT(9)
              UNIT(5)=IUNIT(10)
              UNIT(6)=IUNIT(11)
              UNIT(7)=IUNIT(12)
              UNIT(8)=IUNIT(15)
              IF(IMK(1).NE.1.AND.IMK(2).NE.1) TABLE(1)=0
              CALL OUTPUT(N,S,SS(3*N+1),ITITLE,NFREQ,METHOD(7),IFORM,
             .OP,OPCH,OT,OPLOT,UNIT,ICHECK,TABLE,SS(2*N+1),FREQ)
C
              RETURN
 70           RETURN NARGS
C
C  ************************ FORMAT STATEMENTS ************************
C
 1001         FORMAT('1',20A6/'0',20A6//)
 1002         FORMAT('0MAP ARGUMENT LIST: N =',I6,6X,'METHOD =',I2,I6,
```

```
     .     3I2,2I6,2I2,I6,6X,'ICHECK =',4I2,6X,'IMK =',2I6/' ',19X,
     .     'IUNIT =',15I5/' ',19X,'OP =',6I2,4X,'OT =',5I2,I6,4X,
     .     'OPCH =',3I2,4X,'ITITLE(1) =',A6)
 2003      FORMAT('0INPUT ERROR IN MAP:')
 2004      FORMAT('+',21X,'N =',I6)
 2005      FORMAT('+',21X,'METHOD(1) =',I6)
 2006      FORMAT('+',21X,'METHOD(2) =',I6)
 2007      FORMAT('0METHOD(2) WAS CHANGED TO N:  METHOD(2) =',I6)
 2008      FORMAT('+',21X,'METHOD(4) =',I6)
 2009      FORMAT('+',21X,'METHOD(1) =',I6,5X,'METHOD(6) =',I6,
     .     5X,'N =',I6)
 2010      FORMAT('+',21X,'N =',I6,5X,'METHOD(7) =',I6)
 2012      FORMAT('0INPUT ERROR IN MAP - MASS MATRIX NEEDS TO BE ',
     .     'STORED OUT-OF-CORE, BUT ITS UNIT NUMBER IS NOT POSITIVE'
     .     /'0IUNIT(5) =',I6,10X,'OP(1) =',I2,10X,'OT(1) =',I2,10X,
     .     'OPCH(1) =',I2)
 2013      FORMAT('0INPUT ERROR IN MAP:  L MATRIX NEEDS TO BE ',
     .     'STORED OUT-OF-CORE, BUT ITS UNIT NUMBER IS NOT POSITIVE'
     .     /'0IUNIT(7) =',I6,5X,'OP(3) =',I2,5X,'OT(3) =',I2,5X,
     .     'OPCH(2) =',I2)
 2014      FORMAT('0DETERMINANT OF M MATRIX =',1PE15.7)
 1003      FORMAT('0A FREQUENCY IN (RAD/SEC)**2 IS NEGATIVE:  I =',
     .     I4,2X,1PE14.7)
 1004      FORMAT('0TRACE OF P =',1PE15.7,5X,'MAX(ABS(P(I,I)))/',
     .     'MIN(ABS(P(I,I))) =',1PE15.7)
 1005      FORMAT('0THE',I6,' SMALLEST FREQUENCIES WERE COMPUTED'/
     .     '0SUM OF COMPUTED FREQUENCIES IN (RAD/SEC)**2 =',
     .     1PE14.7/'0MAX(ABS(J(I)))/MIN(ABS(J(I))) =',1PE14.7,
     .     ' WHERE I RUNS FROM 1 TO THE NUMBER OF EIGENVALUES ',
     .     'COMPUTED')
 1006      FORMAT(/'0',50X,'TRIDIAGONAL MATRIX IN (RAD/SEC)**2'/
     .     '0MAIN DIAGONAL:'/(' ',6(I6,1PE14.7)))
 1007      FORMAT('0OFF DIAGONAL:'/(' ',6(I6,1PE14.7)))
           END
```

RESTOR

```
C  TITLE        RESTOR - RESTORE A MATRIX
C
C
C  PROGRAMMER   D. L. MATHER
C
C  SPONSOR      S. N. HOU
C
C  DATE         SEPTEMBER 1970
C
C  PURPOSE      TO STORE THE LOWER TRIANGULAR PART OF A MATRIX WHICH IS
C               STORED IN A TWO DIMENSIONAL ARRAY BY COLUMNS AS THOUGH
C               IT WERE STORED IN A ONE DIMENSIONAL ARRAY
C
C  SUBROUTINE STATEMENT
C
C               SUBROUTINE RESTOR(MAXN,N,A)
C
C  INPUT        BY THE ARGUMENT LIST
C
C               MAXN  THE ARRAY A IS ASSUMED TO BE DIMENSIONED MAXN,MAXN
C                     IN THE CALLING PROGRAM
C               N     THE ACTUAL DIMENSION OF THE MATRIX
C               A     THE ARRAY CONTAINING THE MATRIX
C
C  OUTPUT       BY THE ARGUMENT LIST
C
C               A     THE ARRAY CONTAINING THE LOWER TRIANGULAR PART OF
C                     THE INPUT MATRIX BY COLUMNS STORED AS THOUGH THE
C                     ARRAY WERE A ONE DIMENSIONAL ONE
C
C  SUBPROGRAMS USED
C
C               NONE
C
C  SUBROUTINE STATEMENT
C
C               SUBROUTINE RESTOR(MAXN,N,A)
C
C  SPECIFICATION STATEMENTS
C
C               DIMENSION A(1)
C
C  CHANGE THE WAY THE MATRIX IS STORED
C
               DO 10 J=2,N
               JM1=J-1
               JSKIP=JM1*N-J*JM1/2
               ISKIP=JM1*MAXN
               DO 10 I=J,N
 10            A(I+JSKIP)=A(I+ISKIP)
               RETURN
               END
```