QUALITATIVE ANALYSIS

OF A

DAMPED LINEAR DYNAMIC SYSTEM

By

Jane L. Hawkins

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

1985

ABSTRACT

QUALITATIVE ANALYSIS OF A DAMPED LINEAR DYNAMIC SYSTEM

By

Jane L. Hawkins

Frequently in system analysis a qualitative description of the system response can lead to useful insight into the solution. In the case of a damped linear dynamic system, knowledge of whether the system is overdamped, underdamped or a combination may be adequate, thus making the more expensive calculation of the modes unnecessary. This thesis describes a method and presents software (QUALADS) for determining the qualitative damping nature of a system, given a system description in terms of mass, stiffness and damping matrices.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

A primary question in the analysis of a damped linear dynamic system is whether or not the system can exhibit oscillatory behavior in free motion. This question can be answered quantitatively by calculating the eigenvalues of the differential equations of motion. The process is generally complex and burdensome for large systems. The question can also be answered qualitatively by determining the system damping type. Normally a qualitative analysis requires less time and effort than a quantitative analysis, making it a more attractive method for answering the question as stated above or for gaining preliminary information about a system's dynamics.

There are several techniques for making a qualitative analysis of a damped system. The method derived by Inman and Andry [1,2] proves to be one of the more computationally efficient and comprehensive techniques. It defines a matrix, termed the critical damping matrix, which, when compared with the system's damping matrix, generates the system's damping type. This thesis describes the software QUALADS (QUALitative Analysis of Damped Systems), which uses the technique of Inman and Andry to determine the damping type of a damped linear system described by a set of second order differential equations.

Chapter 2 describes the use of QUALADS and gives some simple examples. Chapter 3 presents an example run and users guide for QUALADS. Chapter 4 is a general discussion of quantitative versus qualitative analyses with a more detailed description of the Inman and Andry algorithm. Chapter 5 explains some of the methods of computation used by QUALADS and computation time for various segments. The final chapter provides a summary and suggestions for additional applications and extensions. The appendix contains the source code for QUALADS.

Chapter 2

QUALADS OVERVIEW

The QUALADS program performs a qualitative analysis of a damped linear dynamic system. The system must be asymptotically stable and defined by a set of second order differential equations. The inputs for QUALADS are the three coefficient matrices of the differential equations, commonly termed the mass, damping and stiffness matrices. The mass and stiffness matrices must be positive definite and symmetric and the damping matrix must be positive semi-definite and symmetric. The mass and stiffness matrices are used to compute a matrix called the critical damping matrix. By examining the matrix formed by the difference of the damping and critical damping matrices, QUALADS determines whether the system is overdamped, critically damped, underdamped or mixed damped. These damping types are the qualitative description which is the output from QUALADS. In the process of determining the damping type, QUALADS computes the undamped eigenvalues and eigenvectors as well as the critical damping matrix. If the user desires quantitative results then the damped eigenvalues may be computed as an additional option.

Consider the following 2 degree of freedom spring-mass-damper system.

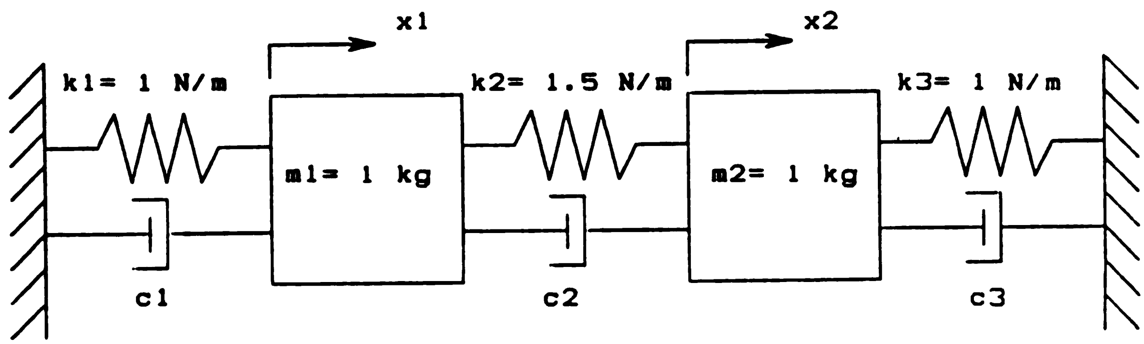


Figure 1. Two Degree of Freedom System

The mass, damping and stiffness matrices for this system are:

$$[M] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad [C] = \begin{bmatrix} c1+c2 & -c2 \\ -c2 & c2+c3 \end{bmatrix} \qquad [K] = \begin{bmatrix} 2.5 & -1.5 \\ -1.5 & 2.5 \end{bmatrix}$$

Depending on the values of c1, c2 and c3 the system will exhibit a particular damping type response.

2.1 Example 1 - Underdamped system

If c1=c2=c3=1 kg/sec, then the damping matrix would be:

$$[C] = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

After inputting the mass, stiffness and damping matrices, QUALADS would respond with

```
****************************
 The system is underdamped
****************************
```

As a check the damped eigenvalues can be computed and they are as follows.

$\lambda_{1,2} = -.5 \pm .866i$

$\lambda_{3,4} = -1.5 \pm 1.323i$

Since the eigenvalues are complex, both modes will be underdamped and thus the system will be underdamped, as predicted by QUALADS.

2.2 Example 2 - Critically damped system

If c1=c3=2 kg/sec and c2=1 kg/sec, then

$$[C] = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

and QUALADS outputs

```
**********************************
 The system is critically damped
**********************************
```

The damped eigenvalues for this system, as shown below, also indicate that the system is critically damped.

$\lambda_{1,2} = -1.0$

$\lambda_{3,4} = -2.0$

Since this system is critically damped, the damping matrix above must be equal to the critical damping matrix. In accord, QUALADS computes the critical damping matrix to be:

$$[C] = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

2.3 Example 3 - Mixed damped system

Letting c1=c3=2.5 kg/sec and c2=.5 kg/sec will form a damping matrix equal to:

$$[C] = \begin{bmatrix} 3 & -.5 \\ -.5 & 3 \end{bmatrix}$$

Using these damping conditions, QUALADS predicts

```
******************************
 The system has mixed damping
******************************
```

The damped eigenvalues are

$$\lambda 1 = -.5 \quad , \quad \lambda 2 = -2.0$$

$$\lambda 3,4 = -1.75 \pm .968i$$

Since one pair of eigenvalues is real and one pair is complex conjugate, one mode will be overdamped and one underdamped, respectively. This is mixed damping.

2.4 Example 4 - Overdamped system

Given c1=c3=3 kg/sec and c2=1 kg/sec, the damping matrix will be

$$[C] = \begin{bmatrix} 4 & -1 \\ -1 & 4 \end{bmatrix}$$

These damping values when inputted into QUALADS will produce:

```
**************************
 The system is overdamped
**************************
```

The damped eigenvalues are:

$$\lambda_1 = -.382 \quad , \quad \lambda_2 = -2.618$$

$$\lambda_3 = -1.0 \quad , \quad \lambda_4 = -4.0$$

which agrees with QUALADS conclusion.

Chapter 3

QUALADS SAMPLE RUN

This chapter provides instruction on how to use QUALADS. First an illustration of a QUALADS run will be presented. More detailed explanations of the subsections denoted by *3.2.n* appear at the end of the chapter, as well as general comments on QUALADS (section 3.2.0).

3.1 - Sample Run

Consider the two degree of freedom system with its four variants that is examined in chapter 2 (Figure 1).

$$[M] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad [C] = \begin{bmatrix} c1+c2 & -c2 \\ -c2 & c2+c3 \end{bmatrix} \quad [K] = \begin{bmatrix} 2.5 & -1.5 \\ -1.5 & 2.5 \end{bmatrix}$$

1) c1=c2=c3= 1 kg/sec

2) c1=c3= 2 kg/sec , c2= 1 kg/sec

3) c1=c3= 2.5 kg/sec , c2= .5 kg/sec

4) c1=c3= 3 kg/sec , c2= 1 kg/sec

The following QUALADS run determines the damping type and the damped eigenvalues for each of the four systems shown above. The underlined items are the information entered by the user.

```
OK, SEG #QUALADS
  Enter the size of the system.
2                                                          *3.2.1*

 Enter mass matrix M by rows

ROW  1
M( 1, 1)= 1                                                *3.2.2*
M( 1, 2)= 0

ROW  2
M( 2, 1)= 0
M( 2, 2)= 1


 Enter stiffness matrix K by rows

ROW  1
K( 1, 1)= 2.5
K( 1, 2)= -1.5

ROW  2
K( 2, 1)= -1.5
K( 2, 2)= 2.5


 Enter the damping matrix C by rows

ROW  1
C( 1, 1)= 2
C( 1, 2)= -1

ROW  2
C( 2, 1)= -1
C( 2, 2)= 2



***************************
 The system is underdamped
***************************

 Do you want to compute the damped eigenvalues?
Y                                                          *3.2.3*

 Would you like to open a file in which to place
      system matrices and output values?
Y                                                          *3.2.4*
 Enter filename
EXOUT
```

```
 The following information can be printed
      to your output file.
      1)Mass Matrix
      2)Stiffness Matrix
      3)Damping Matrix
      4)Critical Damping Matrix
      5)Undamped Natural Frequencies
      6)Modal Matrix
      7)Damped Eigenvalues

 Enter the option numbers desired in a single line
   separated by commas.

1,2,4,5,6,3,7                                              *3.2.5*


 Do you wish to run again with a different C matrix?
Y                                                          *3.2.6*


 Enter the damping matrix C by rows

ROW  1
C( 1, 1)= 3
C( 1, 2)= -1

ROW  2
C( 2, 1)= -1
C( 2, 2)= 3



*********************************
 The system is critically damped
*********************************

 Do you want to compute the damped eigenvalues?
Y
 The following information can be printed
      to your output file.
      1)Mass Matrix
      2)Stiffness Matrix
      3)Damping Matrix
      4)Critical Damping Matrix
      5)Undamped Natural Frequencies
      6)Modal Matrix
      7)Damped Eigenvalues

 Enter the option numbers desired in a single line
   separated by commas.

3,7
```

```
 Do you wish to run again with a different C matrix?
Y


 Enter the damping matrix C by rows

ROW  1
C( 1, 1)= 3
C( 1, 2)= -.5

ROW  2
C( 2, 1)= -.5
C( 2, 2)= 3



******************************
 The system has mixed damping
******************************

 Do you want to compute the damped eigenvalues?
Y
 The following information can be printed
      to your output file.
      1)Mass Matrix
      2)Stiffness Matrix
      3)Damping Matrix
      4)Critical Damping Matrix
      5)Undamped Natural Frequencies
      6)Modal Matrix
      7)Damped Eigenvalues

 Enter the option numbers desired in a single line
   separated by commas.

3,7


 Do you wish to run again with a different C matrix?
Y


 Enter the damping matrix C by rows

ROW  1
C( 1, 1)= 4
C( 1, 2)= -1

ROW  2
C( 2, 1)= -1
C( 2, 2)= 4
```

```
***************************
 The system is overdamped
***************************

 Do you want to compute the damped eigenvalues?
Y
 The following information can be printed
      to your output file.
      1)Mass Matrix
      2)Stiffness Matrix
      3)Damping Matrix
      4)Critical Damping Matrix
      5)Undamped Natural Frequencies
      6)Modal Matrix
      7)Damped Eigenvalues

 Enter the option numbers desired in a single line
   separated by commas.

 3,7


 Do you wish to run again with a different C matrix?
N


*** Output data file is EXOUT

OK,
```

In the QUALADS run above the user requested that an output file be opened and named EXOUT. EXOUT was supplied with the data requested by the user from the menu of options presented after each system solution. The file EXOUT would appear as follows.

```
********************
 Underdamped system
********************

 MASS MATRIX

    Column          Column
     ( 1)            ( 2)
 0.100000D+01  0.000000D+00
 0.000000D+00  0.100000D+01


 STIFFNESS MATRIX

    Column          Column
     ( 1)            ( 2)
 0.250000D+01 -0.150000D+01
-0.150000D+01  0.250000D+01


 CRITICAL DAMPING MATRIX

    Column          Column
     ( 1)            ( 2)
 0.300000D+01 -0.100000D+01
-0.100000D+01  0.300000D+01


 UNDAMPED NATURAL FREQUENCIES (rad/sec)

 1  0.10000D+01
 2  0.20000D+01

 EIGENVECTORS (MODAL MATRIX)

   Column   Column
    ( 1)     ( 2)
   0.7071  -0.7071
   0.7071   0.7071

 DAMPING MATRIX

    Column          Column
     ( 1)            ( 2)
 0.200000D+01 -0.100000D+01
-0.100000D+01  0.200000D+01


 DAMPED EIGENVALUES (Unpaired)
  1 -0.500000D+00+-0.866025D+00i
  2 -0.500000D+00+ 0.866025D+00i
  3 -0.150000D+01+ 0.132288D+01i
  4 -0.150000D+01+-0.132288D+01i
```

```
***************************
 Critically damped system
***************************

 DAMPING MATRIX

    Column        Column
     ( 1)          ( 2)
 0.300000D+01 -0.100000D+01
-0.100000D+01  0.300000D+01


 DAMPED EIGENVALUES (Unpaired)
  1 -0.100000D+01+ 0.000000D+00i
  2 -0.100000D+01+ 0.000000D+00i
  3 -0.200000D+01+ 0.000000D+00i
  4 -0.200000D+01+ 0.000000D+00i


****************************
 System with mixed damping
****************************

 DAMPING MATRIX

    Column        Column
     ( 1)          ( 2)
 0.300000D+01 -0.500000D+00
-0.500000D+00  0.300000D+01


 DAMPED EIGENVALUES (Unpaired)
  1 -0.175000D+01+-0.968246D+00i
  2 -0.175000D+01+ 0.968246D+00i
  3 -0.500000D+00+ 0.000000D+00i
  4 -0.200000D+01+ 0.000000D+00i


********************
 Overdamped system
********************

 DAMPING MATRIX

    Column        Column
     ( 1)          ( 2)
 0.400000D+01 -0.100000D+01
-0.100000D+01  0.400000D+01
```

```
DAMPED EIGENVALUES (Unpaired)
 1 -0.381966D+00+ 0.000000D+00i
 2 -0.100000D+01+ 0.000000D+00i
 3 -0.261803D+01+ 0.000000D+00i
 4 -0.400000D+01+ 0.000000D+00i
```

The command file for the QUALADS run shown would be as follows.

```
SEG #QUALADS
2
1
0
0
1
2.5
-1.5
-1.5
2.5
2
-1
-1
2
Y
Y
EXOUT
1,2,4,5,6,3,7
Y
3
-1
-1
3
Y
3,7
Y
3
-.5
-.5
3
Y
3,7
Y
4
-1
-1
4
Y
3,7
N
```

3.2 - Comments and Instructions for QUALADS

3.2.0 - General Information

System Restrictions - The QUALADS algorithm has the system restrictions shown below. QUALADS checks the input matrices for their required properties and informs the user if he is attempting to run an illegal system. The program will abort if an illegal M or K matrix is entered. If an illegal C matrix is attempted or the system is asymptotically unstable the user will be asked if he wishes to enter a new C matrix.

Restrictions

M matrix - symmetric, positive definite

K matrix - symmetric, positive definite

C matrix - symmetric, positive semi-definite

System - asymptotically stable

Precision - All numbers and operations are double precision.

Yes and No - All Yes and No responses can be answered by Y or y and N or n respectively.

Error Recovery - If an entry is not the proper data type or a valid option, the program response will be "Unrecognized Character" and "Re-enter _____".

3.2.1 - System Size

System size refers to the dimension of the square mass, stiffness and damping matrices. The maximum system size is 200.

3.2.2 - Matrix Entry

QUALADS asks for the values of the mass, stiffness, and damping matrices (in that order), element by element, working by rows.

3.2.3 - Damped Eigenvalue Option

After the damping type has been determined for a system the user has the option of computing the damped eigenvalues of the system. The damped eigenvalues are displayed in the users output file in a partially ordered fashion. The complex eigenvalues are listed first in conjugate pairs by order of magnitude, then the real eigenvalues in order of magnitude.

3.2.4 - Output File

The only information printed to the screen during a QUALADS run is the damping type. If the user wishes to examine the critical damping matrix, damped eigenvalues or other information available in the menu it has to be printed to an output file. The user is asked to enter a filename for the output file, which can be any legal filename. If the filename is already in existence QUALADS will ask if the user wishes to overwrite it.

3.2.5 - Output File Menu Options

QUALADS presents the following menu of data options that can be printed to the users designated output file.

```
1)Mass Matrix
2)Stiffness Matrix
3)Damping Matrix
4)Critical Damping Matrix
5)Undamped Natural Frequencies
6)Modal Matrix
7)Damped Eigenvalues

Enter the option numbers desired in a single line
  separated by commas.
```

As stated above the options that are desired to be printed are entered in a single line separated by commas. The options can be printed in any order and however many are desired. If no options are desired enter a blank line. The damping type is printed to the output file regardless of the users option choice. If the user has not chosen to compute the damped eigenvalues the menu will not contain line 7.

3.2.6 - Multiple Damping Matrix Option

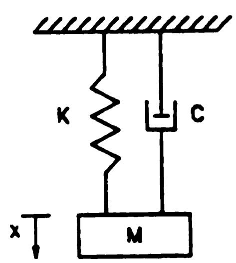
QUALADS has the ability to run more than one set of damping conditions for a single set of mass and stiffness conditions or matrices. After each new damping matrix is entered the damping type is determined and the user again has the option of computing the damped eigenvalues for this new system and printing to the output file. It should be noted that the only values that change for a new system are the damping matrix that was supplied by the user, the damped eigenvalues, if computed, and perhaps the damping type, depending on the system.

Chapter 4

QUALITATIVE VERSUS QUANTITATIVE ANALYSIS

4.1 - One Degree of Freedom System

In order to fully understand a qualitative analysis as compared to a quantitative analysis, a review of a simple one degree of freedom system will be helpful. The diagram below shows a spring-mass-damper system described by the differential equation noted beside it.



$$M\ddot{x} + C\dot{x} + Kx = 0$$

Figure 2. One Degree of Freedom System

The parameters of the system are the mass M, the spring stiffness K, and the damping constant C and are positive real constants. The system reaction to a disturbance or initial state depends on the solution or roots of the following equation called the characteristic equation.

These roots are called the damped eigenvalues of the system. They can produce three possible reactions since the two eigenvalues can be three kinds of numbers - real, real repeated or complex conjugates. Each case has a name which is called its damping 'type'. The eigenvalues and general solutions for the three cases or damping types are shown below. Note that determining the eigenvalues determines the general solution.

Damping Type	Eigenvalues General solution
Overdamping	real $\lambda_{1,2} = -\frac{C}{2M} \pm \frac{1}{2M}\sqrt{C^2 - 4KM}$ $x(t) = Ae^{\left(-\frac{C}{2M} + \frac{1}{2M}\sqrt{C^2 - 4KM}\right)t} + Be^{\left(-\frac{C}{2M} - \frac{1}{2M}\sqrt{C^2 - 4KM}\right)t}$
Critical Damping	real repeated $\lambda_{1,2} = -\frac{C}{2M}$ $x(t) = Ae^{-\frac{C}{2M}t} + Bte^{-\frac{C}{2M}t}$
Underdamping	complex $\lambda_{1,2} = -\frac{C}{2M} \pm \frac{1}{2M}\sqrt{4KM - C^2}\, i$ $x(t) = e^{-\frac{C}{2M}t}\left(A \sin \frac{\sqrt{4KM - C^2}}{2M} t + B \cos \frac{\sqrt{4KM - C^2}}{2M} t\right)$

As can be seen by the general solutions, the underdamped system has the only oscillatory response. Knowing which of the three damping types the system exhibits tells whether the system could oscillate or not. Arriving at this classification without calculating the eigenvalues would be a qualitative analysis of the system. The key value for determining the aforementioned is the critical damping constant Ccr, which is defined as follows:

$$Ccr = 2\sqrt{KM}$$

The relationship of the system's damping constant C to the critical damping constant Ccr determines the damping type as shown below.

$C > Ccr$	Overdamping
$C = Ccr$	Critical Damping
$C < Ccr$	Underdamping

Note that the qualitative method allows for simple and almost computation free determination of damping type if the damping constant C is changed.

For the one degree of freedom system the amount of work required in the determination of the eigenvalues for the quantitative analysis is just slightly more than the work required in comparing the critical damping to the damping constant for the qualitative analysis. The difference in workload increases when a multidegree of freedom system is examined.

4.2 - Multidegree of Freedom System

A multidegree of freedom system is described by a group of coupled second order differential equations which are expressed in matrix form below.

$$[M]\{\ddot{x}\} + [C]\{\dot{x}\} + [K]\{x\} = 0$$

The coefficient matrices are the mass matrix [M], the damping matrix [C] and the stiffness matrix [K], all of which have real elements. If the system has N degrees of freedom, then the square matrices [M], [C] and [K] are size N x N. The unknowns are the terms of the {x} vector where each x represents the relative motion of the system at some location.

For an N degree of freedom system there will be N pairs of eigenvalues. Each pair forms a normal mode which looks like the solution to a one degree of freedom system. Thus each mode can be defined as overdamped, critically damped or underdamped. The system's solution is a linear combination of these normal modes, where the magnitude of each mode's contribution depends on the initial conditions. If all of the normal modes are overdamped or nonoscillatory, then the system must be nonoscillatory; it is called an overdamped system. If all of the modes are critically damped, then the system is critically damped. If all of the modes are underdamped, then the system is underdamped (oscillatory). There is of course the possibility that the modes are a mixture of the three damping types. In this case the system is said to be mixed damped.

As in the one degree of freedom system, the principal step in a quantitative analysis of a multidegree of freedom system is to calculate the system's eigenvalues. Examining the eigenvalues will tell if the system can oscillate. The most comprehensive method to solve for the eigenvalues of a size N system is to convert the system into a size 2 x N system of first order state equations. The determination of the 2N eigenvalues requires lengthy matrix operations. Taking this into account, as well as the fact that system size has doubled, may make a quantitative analysis less attractive than a qualitative analysis in a design context.

4.3 - Qualitative Method by Inman and Andry

A qualitative analysis of a multidegree of freedom system should determine whether a system can oscillate without computing the solution, i.e. eigenvalues. A method which accomplishes this was derived by Inman and Andry [1,2]. Their algorithm defines and uses a matrix called the critical damping matrix, [Ccr].

$$[Ccr] = 2[M][U][w][U^T][M]$$

The matrix [U] is the undamped modal matrix normalized such that $[U^T][M][U] = [I]$ and the matrix $[\omega]$ is the diagonal matrix containing the undamped natural frequencies. Both matrices come from solving the eigenvalue problem of the undamped system $[M]\{\ddot{x}\} + [K]\{x\} = 0$. The algorithm is restricted to asymptotically stable systems with symmetric, positive definite mass and stiffness matrices and a symmetric, positive semi-definite damping matrix.

The critical damping matrix serves a function similar to that of the critical damping constant in the single degree of freedom system. The damping type of a system is ascertained by determining the definiteness of the matrix equal to the difference between the damping matrix [C] and the critical damping matrix [Ccr]. The relationships are as follows.

[C - Ccr]	System damping type
positive definite	overdamping
zero	critical damping
negative definite	underdamping
indefinite	mixed damping

Note again that changing the damping of the system does not affect the critical damping value. Thus to reexamine a system with a different damping matrix only requires determining the definiteness of the new [C - Ccr]. This makes the qualitative method more attractive than the quantitative, which would require that the analysis be reworked from the beginning.

Chapter 5

QUALADS COMPUTATIONAL INFORMATION

This chapter presents a brief description of the major methods of computation used in QUALADS. It then compares the computational load for the qualitative method of determining damping type with that for the quantitative procedure of determining the damped eigenvalues.

5.1 - Methods of Computation

Eigenvalue Problem - The undamped natural frequencies, modal matrix, and damped eigenvalues are all computed using the IMSL eigenvalue package EIGZF [3]. Given the mass and stiffness matrices, EIGZF computes the undamped eigenvalues and eigenvectors, which produce the natural frequencies and normalized modal matrix. For the damped eigenvalues the system equations are converted to state space equations [4]. The "mass" and "stiffness" matrices for state space are as follows.

$$"M" = \left[\begin{array}{c|c} 0 & M \\ \hline M & C \end{array}\right] \qquad "K" = \left[\begin{array}{c|c} -M & 0 \\ \hline 0 & K \end{array}\right]$$

Given the above matrices EIGZF computes the damped eigenvalues unpaired and unordered.

Definiteness - To determine the damping type QUALADS must determine if the [C - Ccr] matrix is positive definite, negative definite, indefinite or zero. The procedure used involves determining the sign of the determinants of the submatrices. Let [A] = [C -Ccr]. For the square matrix [A] that is size n, the square submatrix [Ak] is a size k matrix formed by the first k rows and first k columns of the [A] matrix. The following conditions determine the corresponding definite type [5].

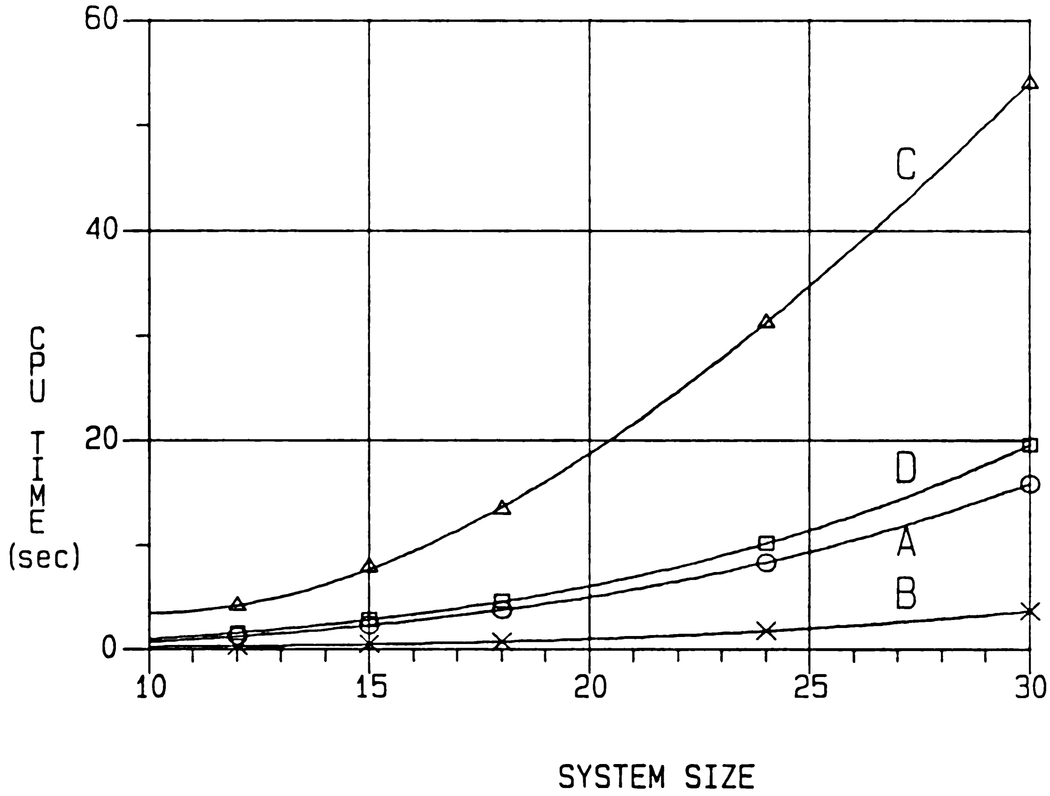
Condition	Type
Det([Ak]) > 0 for k=1,2,....,n	positive definite
Det([-Ak]) > 0 for k=1,2,....,n	negative definite

If [A] is neither positive definite nor negative definite nor zero, then it is indefinite.

Asymptotic Stability - If the damping matrix [C] of a system is positive semi-definite, then the system could be asymptotically unstable. QUALADS uses the criterion that a system is asymptotically unstable if any of the modal vectors of the undamped system lie in the null space of the damping matrix [C] [6]. This check is achieved by examining the matrix that is the product of the damping matrix [C] and the modal matrix [U]. If none of the columns of this product matrix are zero vectors then the system is stable.

5.2 - Computation Time

The major part of the computational load in determining the damping type occurs during the computation of the critical damping matrix [Ccr] as opposed to determining the definiteness of the [C - Ccr] matrix. If a set of several damping matrices is examined during one run then the computational load is considerably less after the initial system has been solved. This occurs because the critical damping matrix, which is determined by the mass and stiffness matrix, has already been computed. The computational load incurred by the quantitative analysis, which consists of computing the damped eigenvalues, is much greater than that of the qualitative analysis. Figure 3 shows the CPU (Central Processor Unit) time required to determine [Ccr], the definiteness (damping type) given [C] and [Ccr], and the damped eigenvalues. All the calculations were for overdamped systems, which produces the maximum computational effort in the definiteness routine. Although the data suggests that CPU time is a function of the system size squared for all three computations, the controlling coefficient of the definiteness algorithm is small relative to the coefficient of the eigenvalue algorithm. Therefore, as system size increases the computational savings afforded by the qualitative method become substantial.



A - Computation of critical damping matrix [Ccr]
B - Computation of definiteness of [C] - [Ccr] (damping type) given [Ccr]
C - Computation of damped eigenvalues
D - Computation of damping type given [M], [C], and [K] (i.e. A + B)

Figure 3. CPU Time Plot

These results were obtained on a Prime 750 virtual memory super mini-computer with a 6 Mb memory operating under PRIMOS Revision 19.2.

Chapter 6

SUMMARY

A qualitative analysis of damped linear dynamic systems was considered. This analysis consists of an algorithm derived by Inman and Andry that determines a matrix called the critical damping matrix, which is then used to generate the damping type of a system described by a set of linear second order differential equations. Software called QUALADS was written to implement this qualitative analysis method. In addition to finding the damping type, QUALADS can execute a quantitative analysis of a system by computing the damped eigenvalues.

6.1 - Conclusions

In the examination of a damped system the question of the system's ability to exhibit oscillatory behavior in free motion is generally an important question. The qualitative analysis described in this thesis offers an effective method for answering this question. As compared to a quantitative solution to the problem, the qualitative analysis requires less computational effort. The compute time and effort difference is even greater when several damping mechanisms for the same model are to be examined. There the qualitative solution determined by QUALADS may yield sufficient information about a system or provide sufficient preliminary data to guide a further investigation of the damping behavior.

6.2 - Extensions

An interesting extension to the qualitative analyses examined in this thesis is the inverse problem of designing a system that will not oscillate when perturbed. For an overdamped or nonoscillatory system, the analysis requires that the matrix [C - Ccr] be positive definite. This requirement would give rise to a set of nonlinear inequalities. An algorithm to solve these inequalities, along with additional user imposed constraints, could prove to be a very useful design tool.

LIST OF REFERENCES

LIST OF REFERENCES

1. Inman, D. J. and Andry, A. N., "Some Results on the Nature of Eigenvalues of Discrete Damped Linear Systems", Journal of Applied Mechanics, Dec. 1980.

2. Inman, D. J., "Nature of the Solutions of Damped Linear Dynamic Systems, Ph.D. Dissertation, Michigan State University, 1980.

3. "IMSL Library - User's Manual", IMSL Inc., Texas, 1984, Chapter E.

4. Meirovitch, L., "Analytical Methods in Vibrations", The Macmillan Co., N.Y., 1967, pg. 411,412.

5. Phillips, R., "Lecture Notes on Matrix Theory", Michigan State University, 1981.

6. Moran, T. J., "A Simple Alternative to the Routh-Hurwitz Criterion for Symmetric Systems", Journal of Applied Mechanics, Dec. 1970.

APPENDIX

The following appendix consists of the source code for QUALADS. The code is written in FORTRAN 77. The first page of code is the program MAIN which calls the main subroutines of the program. An explanation of the purpose of each subroutine is given before its call. A calling tree for the subroutines of QUALADS follows.

```
MAIN (QUALADS)
     DIMSIZ
     MATINP
     CHKM
          CHKSYM
          PDCHK
               DETER
               DIAGPR
     MATINP
     CHKK
          CHKSYM
          PDCHK
               DETER
               DIAGPR
     MATINP
     CHKC
          CHKSYM
          PDCHK
               DETER
               DIAGPR
          PSDCHK
               DETER
               DIAGPR
     EIGEN
          EIGZF (IMSL)
          SORT
          NORM
     CHKSTB
     COMP
     DEFNT
          DETER
          DIAGPR
     EIGQUE
     DMPEIG
          EIGZF (IMSL)
     OPNFIL
     OUTFIL
     RERUN
```

The subroutines shown in the calling tree appear in the code in the following order. The subroutines TRMTYP and PAGE perform screen paging for those terminals that require it and are called throughout QUALADS in several of its subroutines.

MAIN

DIMSIZ

CHKM

CHKK

CHKC

CHKSTB

EIGEN

SORT

NORM

DEFNT

EIGQUE

DMPEIG

OPNFIL

OUTFIL

RERUN

CHKSYM

PDCHK

PSDCHK

DETER

DIAGPR

TRMTYP

PAGE

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C********************************************************************C
C*                         PROGRAM MAIN                             *C
C*                                                                  *C
C*    MAIN calls the major subroutines and creates and              *C
C*            dimensions the arrays they will need.                 *C
C*                                                                  *C
C********************************************************************C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C*                                                                  *C
C*   *Variable List*                                                *C
C*    C - Damping matrix                                            *C
C*    CCR - Critical damping matrix                                 *C
C*    K -Stiffness matrix                                           *C
C*    LAMDA - Damped eigenvalues                                    *C
C*    M - Mass matrix                                               *C
C*    U - Modal matrix (undamped eigenvectors)                      *C
C*    W - Undamped natural frequencies                              *C
C*    WK** - General work matrices                                  *C
C*    Logicals - .TRUE. meaning                                     *C
C*            ABORT - M or K are illegal ( i.e. not symmetric       *C
C*                    or positive definite)                         *C
C*            CONT - User has opted to input another C matrix       *C
C*            DEVAL - User asked to compute damped eigenvalues      *C
C*            FIRST - Analysis is of first C matrix                 *C
C*            OK - C matrix is legal (i.e. symmetric and at         *C
C*                    least positive semi-definite)                 *C
C*            OPEN - User has opened a file for output values       *C
C*            PSD - C matrix is positive semi-definite              *C
C*            UCALC - The modal matrix U and the undamped           *C
C*                    eigenvalues have been computed                *C
C*                                                                  *C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      REAL*8 M(40000),C(40000),K(40000),U(40000),CCR(40000),W(200)
      COMPLEX*16 LAMDA(400)
      CHARACTER*6 TYPE
C                                                                    C
      REAL*8 WK33(40000),WK44(40000)
      REAL*8 WK11(160000),WK22(160000),WKL(320000),WK1(400)
      COMPLEX*16 WK1C(200),WK11C(160000)
      CHARACTER*32 FILEN
      LOGICAL FIRST,OK,OPEN,DEVAL,UCALC,ABORT,CONT,PSD
C                                                                    C
      COMMON/STUFF/M,C,K,U,CCR,W,WK33,WK44,WK11,WK22,WK1,WK1C
      COMMON/STUF2/WKL,WK11C
C                                                                    C
      UCALC=.FALSE.
      FIRST=.TRUE.
      ABORT=.FALSE.
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C********************************************************************C
C*    DIMSIZ asks the user for the system size (N), i.e. the        *C
C*       dimensions of the matrices.                                *C
C********************************************************************C
C                                                                    C
      CALL DIMSIZ(N)
```

```
      NN=N*N
      N2=2*N
      NN=2*N2*N2
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C*********************************************************************C
C*    MATINP asks the user to input the mass, stiffness, and         *C
C*       damping matrices (M,K,C).                                   *C
C*    CHKM and CHKK check the M and K matrices, respectively, for    *C
C*       symmetry and positive definiteness.                         *C
C*    CHKC checkc the C matrix for symmetry and positive semi-       *C
C*       definiteness.                                               *C
C*    CHKSTB checks to see if the system is asymptotically stable.  *C
C*       (Since this check requires the modal matrix U, EIGEN is     *C
C*       is called to compute it.  See EIGEN below.)                 *C
C*********************************************************************C
C
      CALL MATINP(M,N,'M')
      CALL CHKM(M,N,WKL,NN,ABORT)
      IF(ABORT)GO TO 999
C                                                                     C
      CALL MATINP(K,N,'K')
      CALL CHKK(K,N,WKL,NN,ABORT)
      IF(ABORT)GO TO 999
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
10    CONTINUE
      CALL MATINP(C,N,'C')
C                                                                     C
      CALL CHKC(C,N,WKL,NN,OK,PSD)
      IF(OK)THEN
        IF(PSD)THEN
          IF(.NOT.UCALC)THEN
            CALL EIGEN(M,K,W,U,N,NN2,WK1,WKL,WK1C,WK11,WK22,WK11C)
            UCALC=.TRUE.
          ENDIF
          CALL CHKSTB(C,U,WK11,N,OK)
        ENDIF
      ENDIF
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      IF(OK)THEN
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C*********************************************************************C
C*     EIGEN computes the undamped eigenvalues (natural              *C
C*        frequencies)(W) and eigen vectors (modal matrix)(U).       *C
C*        NN2 is the dimension of a workspace needed by the          *C
C*        eigenvalue package EIGZF (IMSLD).                          *C
C*********************************************************************C
C                                                                     C
      IF(.NOT.UCALC)THEN
        CALL EIGEN(M,K,W,U,N,NN2,WK1,WKL,WK1C,WK11,WK22,WK11C)
        UCALC=.TRUE.
```

```
      ENDIF
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C**********************************************************************C
C*    COMP computes the critical damping matrix (CCR).                *C
C**********************************************************************C
C                                                                      C
      IF(FIRST)THEN
        CALL COMP(U,V,CCR,M,N,WK11,WK22,WK33,WK44)
      ENDIF
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C**********************************************************************C
C*    DEFNT computes the definiteness (TYPE) of the matrix which      *C
C*       is the difference of the damping matrix (C) and the          *C
C*       critical damping matrix (CCR).                               *C
C*       NN is the dimension of a vector formed by the                *C
C*       aforementioned difference (C - CCR).                         *C
C*    The system description is printed out by DEFNT at this time,*C
C*       i.e. it is stated which type of system (overdamped,          *C
C*       critically damped, underdamped, or mixed) is present.        *C
C**********************************************************************C
C                                                                      C
      CALL DEFNT(C,CCR,TYPE,N,NN,WK11,WKL)
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C**********************************************************************C
C*    EIGQUE Asks the user if he unats to compute the damped          *C
C*          eigenvalues.  DEVAL is true if answer is yes.             *C
C*    DMPEIG computes them.                                           *C
C**********************************************************************C
C                                                                      C
C                                                                      C
      CALL EIGQUE(DEVAL)
      IF(DEVAL)THEN
        CALL DMPEIG(M,C,K,LAMDA,N,N2,NN2,WK1,WKL,WK11,WK22,WK11C)
      ENDIF
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C**********************************************************************C
C*   OUTFIL creates an output file with the users choice of           *C
C*       values, such as any of the system matrices.                  *C
C**********************************************************************C
C                                                                      C
      IF(FIRST)CALL OPNFIL(OPEN,FILEN)
      IF(OPEN)CALL OUTFIL(M,C,K,CCR,U,V,N,TYPE,LAMDA,N2,DEVAL)
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      FIRST=.FALSE.
      ENDIF
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C**********************************************************************C
C*   RERUN asks the user if he wishes to rerun the problem            *C
```

```
C*      using a new damping matrix, but maintaining the          *C
C*      same mass and stiffness matrices.                        *C
C*****************************************************************C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
      CALL RERUN(CONT)
      IF(CONT)GO TO 10
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                 C
      IF(OPEN)THEN
        CLOSE(12)
        WRITE(1,100)FILEN
100     FORMAT(/,'*** Output data file is ',A32,/)
      ENDIF
999   CALL EXIT
      END
C                                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                      Subroutine DIMSIZ                              C
C                                                                     C
C     DIMSIZ asks the user to input the dimension size N of the       C
C  system, i.e. matrix size or  number of degrees of freedom.         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  *Variable List*                                                    C
C   N - System size                                                   C
C   LOGICALS - .TRUE. meaning                                         C
C       ERROR - Unrecognized character                                C
C       REENT - Variable needs to be reenterd                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      SUBROUTINE DIMSIZ(N)
C                                                                     C
      CHARACTER*32 STRING
      LOGICAL ERROR, REENT
C                                                                     C
      CALL TRMTYP(NP)
      IF(NP.EQ.1)CALL PAGE(NL)
C                                                                     C
      REENT=.TRUE.
      WHILE(REENT)DO
        ERROR=.TRUE.
        PRINT*,' Enter the size of the system.'
        READ(*,'(A32)')STRING
        READ(STRING,100,ERR=10)N
        REENT=.FALSE.
        ERROR=.FALSE.
10      IF(ERROR)THEN
          PRINT*,'Unrecognized character'
          REENT=.TRUE.
        ENDIF
        IF(N.GT.200.AND..NOT.ERROR)THEN
          PRINT*,'Size limit is 200'
          REENT=.TRUE.
        ENDIF
      ENDWHILE
100   FORMAT(I3)
C                                                                     C
      RETURN
      END
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                    Subroutine MATINP                               C
C                                                                    C
C     MATINP asks the user to input the values of the mass,          C
C  stiffness and damping matrices, entry by entry and row by row.    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  *Variable List*                                                   C
C   A - System matrix (M,K, or C) to be entered                      C
C   N - System size                                                  C
C   MAT - Name of matrix to be entered (M,K, or C)                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      SUBROUTINE MATINP(A,N,MAT)
C                                                                    C
      REAL*8 A(N,N)
      CHARACTER*32 STRING
      CHARACTER*1 MAT
C                                                                    C
      CALL TRMTYP(NP)
C                                                                    C
      IF(NP.EQ.1)CALL PAGE(NL)
      PRINT*,' '
      PRINT*,' '
      IF(MAT.EQ.'M')THEN
         PRINT*,'Enter mass matrix M by rows'
      ELSEIF(MAT.EQ.'K')THEN
         PRINT*,'Enter stiffness matrix K by rows'
      ELSEIF(MAT.EQ.'C')THEN
         PRINT*,'Enter the damping matrix C by rows'
      ENDIF
C                                                                    C
      NL=3
      DO 20 I=1,N
      IF(NL.GE.30.AND.NP.EQ.1.AND.J.NE.1)CALL PAGE(NL)
      PRINT*,' '
      WRITE(1,60)I
60    FORMAT('ROW ',I2)
      NL=NL+2
      DO 30 J=1,N
      IF(NL.GE.30.AND.NP.EQ.1)CALL PAGE(NL)
      WRITE(1,40)MAT,I,J
40    FORMAT(A1,'(',I2,',',I2,')= ',$)
50    READ(*,'(A32)')STRING
      READ(STRING,*,ERR=100)A(I,J)
      NL=NL+2
      GO TO 30
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C     Error statment.                                                C
C                                                                    C
100   PRINT*,'Unrecognized entry.  Re-enter value.'
      NL=NL+2
      GO TO 50
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
30    CONTINUE
20    CONTINUE
```

```
C                                                                      C
      RETURN
      END
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                       Subroutine CHKM                                C
C                                                                      C
C       CHKM determines if the M matrix is legal, i.e. symmetric      C
C   and positive definite.                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  *Variable List*                                                     C
C   M - Mass matrix                                                    C
C   WK - Work matrix                                                   C
C   LOGICALS - .TRUE. meaning                                          C
C       ABORT - M illegal                                              C
C       OK - M is legal                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      SUBROUTINE CHKM(M,N,WK,NN,ABORT)
C                                                                      C
      REAL*8 M(N,N),WK(NN)
      LOGICAL OK,ABORT
C                                                                      C
      CALL CHKSYM(M,N,OK)
      IF(OK)THEN
        CALL PDCHK(M,N,WK,NN,OK)
        IF(.NOT.OK)THEN
          PRINT*,' M is not positive definite.  Program abort.'
        ENDIF
      ELSE
        PRINT*,' M is not symmetric. Program abort.'
      ENDIF
      IF(.NOT.OK)ABORT=.TRUE.
C                                                                      C
      RETURN
      END
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                     Subroutine CHKK                              C
C                                                                  C
C      CHKK determines if the K matrix is a legal sytem matrix,    C
C  i.e. symmetric and positive definite.                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C *Variable List*                                                  C
C  K - Stiffness matrix                                            C
C  WK - Work matrix                                                C
C  LOGICALS - .TRUE.  meaning                                      C
C       ABORT - K not legal                                        C
C       OK - K legal                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
      SUBROUTINE CHKK(K,N,WKL,NN,ABORT)
C                                                                  C
      REAL*8 K(N,N),WKL(NN)
      LOGICAL OK,ABORT
C                                                                  C
      CALL CHKSYM(K,N,OK)
      IF(OK)THEN
        CALL PDCHK(K,N,WKL,NN,OK)
        IF(.NOT.OK)THEN
          PRINT*,' K is not positive definite.  Program abort.'
        ENDIF
      ELSE
        PRINT*,' K is not symmetric. Program abort.'
      ENDIF
      IF(.NOT.OK)ABORT=.TRUE.
C                                                                  C
      RETURN
      END
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                      Subroutine CHKC                              C
C                                                                   C
C       CHKC determines if the C matrix is legal, meaning it        C
C   must be symmetric and either positive definite or positive      C
C   semi-definite.                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   *Varibale List*                                                 C
C    C - Damping matrix                                             C
C    WK - Work matrix                                               C
C    LOGICALS - .TRUE.  meaning                                     C
C       OK - C matrix is legal                                      C
C       PD - C matrix is positive definite                          C
C       PSD - Cmatrix is positive semi definite                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      SUBROUTINE CHKC(C,N,WK,NN,OK,PSD)
C                                                                   C
      REAL*8 C(N,N),WK(NN)
      LOGICAL OK, PSD, PD
C                                                                   C
      OK=.TRUE.
      PSD=.FALSE.
C                                                                   C
      CALL CHKSYM(C,N,OK)
      IF(OK)THEN
        CALL PDCHK(C,N,WK,NN,PD)
        IF(.NOT.PD)THEN
          CALL PSDCHK(C,N,WK,NN,PSD)
          IF(.NOT.PSD)THEN
            OK=.FALSE.
            PRINT*,' C is not positive semi-definite.'
            PRINT*,' Illegal system.'
          ENDIF
        ENDIF
      ELSE
        PRINT*,' C is not symmetric.  Illegal system.'
      ENDIF
C                                                                   C
      RETURN
      END
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                     Subroutine CHKSTB                            C
C                                                                  C
C       CHKSTB determines if the system is asymptotically stable.  C
C   This check is only necessary if the C matrix is positive       C
C   semi-definite.                                                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  *Variable List*                                                 C
C   C - Damping matrix                                             C
C   CU - Product of the C matrix and U matrix                      C
C   U - Modal matrix, undamped eigen values                        C
C   LOGICALS - .TRUE. meaning                                      C
C       STBL - System is asymptotically stable                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
      SUBROUTINE CHKSTB(C,U,CU,N,STBL)
C                                                                  C
      REAL*8 C(N,N),U(N,N),CU(N,N)
      LOGICAL STBL
C                                                                  C
      STBL=.TRUE.
C                                                                  C
      CALL DMMLT(CU,C,U,N,N,N)
      J=1
      WHILE(STBL.AND.J.LE.N)DO
        I=1
        WHILE(DABS(CU(I,J)).LT.1.E-6.AND.I.LE.N)DO
          I=I+1
        ENDWHILE
        IF(I.EQ.N+1)STBL=.FALSE.
        J=J+1
      ENDWHILE
C                                                                  C
      IF(.NOT.STBL)THEN
       PRINT*,' System is not asymptotically stable.'
       PRINT*,' Illegal system.'
      ENDIF
C                                                                  C
      RETURN
      END
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C                      SUBROUTINE EIGEN                            C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C      EIGEN computes the eigenvectors and natural frequencies of  C
C          the system given the mass and stiffness matrices.       C
C          All numbers are in double precision.                    C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C                                                                  C
      SUBROUTINE EIGEN(MMAIN,KMAIN,W,UREAL,N,NN,BETA,WK,ALFA
     &,MASS,STIF,UCOMP)
C                                                                  C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C                                                                  C
      REAL*8 MMAIN(N,N),KMAIN(N,N),W(N)
      REAL*8 BETA(N),WK(NN),MASS(N,N),STIF(N,N),UREAL(N,N)
      COMPLEX*16 ALFA(N),UCOMP(N,N)
C                                                                  C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C      Rename the mass and stiffness matrices for preservation.    C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C                                                                  C
      DO 110 I=1,N
      DO 100 J=1,N
      MASS(I,J)=MMAIN(I,J)
      STIF(I,J)=KMAIN(I,J)
100   CONTINUE
110   CONTINUE
C                                                                  C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C       Compute the eigenvectors and eigenvalues                   C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C                                                                  C
      CALL EIGZF(STIF,N,MASS,N,N,2,ALFA,BETA,UCOMP,N,WK,IER)
C                                                                  C
      DO 210 I=1,N
        DO 200 J=1,N
          UREAL(I,J)=REAL(UCOMP(I,J))
200     CONTINUE
210   CONTINUE
C                                                                  C
      DO 220 I=1,N
         ALFA(I)=ALFA(I)/BETA(I)
         W(I)=ALFA(I)
         IF (W(I) .GT. 1.D-6) THEN
```

```
          V(I)=DSQRT(V(I))
        ELSEIF(V(I).GT.-1.D-6.AND.V(I).LT.1.D-6)THEN
          V(I) = 0.0
        ELSE
          PRINT *,'THIS CAN NEVER HAPPEN -JANE HAWKINS'
          PRINT *, 'NEGATIVE ARG TO SQRT IN EIGEN'
        END IF
220   CONTINUE
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       SORT orders the modal matrix so that the first column         C
C          is associated with the first eigenvector, etc.             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      CALL SORT(V,UREAL,N)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   NORM normalizes the modal matrix U so that UT*M*U=I.             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      CALL NORM(UREAL,MMAIN,N,MASS,STIF,WK)
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C                                                                     C
      SUBROUTINE SORT(V,E,M)
      REAL*8 V(M),E(M,M)
      MM=M-1
         DO 40 I=1,MM
         DO 30 J=1,MM
            IF(V(J).LE.V(J+1)) GO TO 25
            N=J+1
            VTEMP=V(J)
            V(J)=V(N)
            V(N)=VTEMP
            DO 20 K=1,M
               ETEMP=E(K,J)
               E(K,J)=E(K,N)
               E(K,N)=ETEMP
20          CONTINUE
25       CONTINUE
30       CONTINUE
40    CONTINUE
      RETURN
      END
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      SUBROUTINE NORM(U,M,N,UT,UTM,UTMU)
C                                                                     C
      REAL*8 U(N,N),M(N,N)
      REAL*8 UT(N,N),UTM(N,N),UTMU(N,N)
C                                                                    C
      CALL DMTRN(UT,U,N)
      CALL DMMLT(UTM,UT,M,N,N,N)
      CALL DMMLT(UTMU,UTM,U,N,N,N)
      DO 10 J=1,N
      DO 20 I=1,N
      U(I,J)=U(I,J)/DSQRT(UTMU(J,J))
20    CONTINUE
10    CONTINUE
C                                                                    C
      RETURN
      END
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                        Subroutine COMP                             C
C                                                                    C
C   COMP computes the critical damping matrix, using the             C
C        formula: Ccr=2*M*UT*WM*U*M .                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      *Variable list*                                               C
C                                                                    C
C       CCR - Critical damping matrix.                               C
C       N - Size of matrices.                                        C
C       W - Vector of natural frequencies (ordered).                 C
C       M - Mass matrix.                                             C
C       UT - Transpose of modal matrix U.                            C
C       WM - Diagonal matrix of natural frequencies.                 C
C       U - Modal matrix (UT*M*U=I).                                 C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      SUBROUTINE COMP(U,W,CCR,M,N,UT,WM,C1,C2)
C                                                                    C
      REAL*8 U(N,N),W(N),M(N,N)
      REAL*8 UT(N,N),WM(N,N),CCR(N,N)
      REAL*8 C1(N,N),C2(N,N)
C                                                                    C
      DO 200 J=1,N
      DO 210 I=1,N
      WM(I,J)=0.
      WM(I,I)=W(I)
210   CONTINUE
200   CONTINUE
      CALL DMTRN(UT,U,N)
      CALL DMMLT(C1,M,U,N,N,N)
      CALL DMMLT(C2,C1,WM,N,N,N)
      CALL DMMLT(C1,C2,UT,N,N,N)
      CALL DMMLT(C2,C1,M,N,N,N)
      CALL DMSCL(CCR,C2,N,N,.2D01)
C                                                                    C
      RETURN
      END
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                     Subroutine DEFNT                              C
C                                                                   C
C      DEFNT determines the type of damping present in the          C
C          system.                                                  C
C     1)C-Ccr=0: critically damped                                  C
C     2)C-Ccr positive definite: overdamped                         C
C     3)C-Ccr negative definite: underdamped                        C
C     4)C-Ccr indefinite: mixed damping                             C
C     The definiteness of the matrix is determined by examining     C
C      the determinant's of the submatrices: sizes 1 X 1 through    C
C      N X N.                                                       C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
C     *Variable List*                                               C
C      C - Damping matrix.                                          C
C      CCR - Critical damping matrix.                               C
C      DIF - Matrix that is the difference of C - CCR               C
C      TYPE - Type of definiteness.                                 C
C      N - Size of matrices.                                        C
C      DET - Determinant of submatrices.                            C
C      MS - Size of submatrix.                                      C
C      DPR - Product of diagonal terms of submatrix.                C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      SUBROUTINE DEFNT(C,CCR,TYPE,N,NN,DIF,TM)
C                                                                    C
      REAL*8 C(N,N),CCR(N,N),DIF(N,N),TM(NN),DET,DPR
      CHARACTER*6 TYPE
      LOGICAL QUITP,QUITN
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      PRINT*,' '
C                                                                   C
C ---Compute C - Ccr.
C
      CALL DMSUB(DIF,C,CCR,N,N)
C                                                                    C
      TYPE='INDEF'
      QUITP=.FALSE.
      QUITN=.FALSE.
C                                                                    C
C ---Check to see if C - Ccr = 0., thus critically damped.
C                                                                    C
      NC=0
      DO 10 I=1,N
      DO 20 J=1,N
      IF(DABS(DIF(I,J)).LE..0001)NC=NC+1
20    CONTINUE
10    CONTINUE
      IF(NC.EQ.N**2)THEN
        TYPE='ZERO'
        QUITP=.TRUE.
        QUITN=.TRUE.
      ENDIF
C
```

```
C --- Determine if C - Ccr is positive definite, negative definite
C --- or indefinite.
C
      MS=1
      WHILE(MS.LE.N.AND..NOT.QUITP)DO
        CALL DETER(DET,DIF,TM,MS,N,NN,1.DO)
        CALL DIAGPR(DIF,N,MS,DPR)
        IF(DET/DPR.LT.1.E-12)QUITP=.TRUE.
        MS=MS+1
      ENDWHILE
      IF(.NOT.QUITP)TYPE='POSDEF'
      MS=1
      WHILE(MS.LE.N.AND.QUITP.AND..NOT.QUITN)DO
        CALL DETER(DET,DIF,TM,MS,N,NN,-1.DO)
        CALL DIAGPR(DIF,N,MS,DPR)
        IF(DET/DPR.LT.1.E-12)QUITN=.TRUE.
        MS=MS+1
      ENDWHILE
      IF(.NOT.QUITN.AND.QUITP)TYPE='NEGDEF'
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   Print out the determined type                                     C
C                                                                     C
      CALL TRMTYP(NP)
      IF(NP.EQ.1)CALL PAGE(NL)
C                                                                     C
      IF(TYPE.EQ.'POSDEF')WRITE(1,100)
100    FORMAT(//,26('*'),/,' The system is overdamped',/,26('*'))
      IF(TYPE.EQ.'NEGDEF')WRITE(1,110)
110    FORMAT(//,27('*'),/,' The system is underdamped',/,27('*'))
      IF(TYPE.EQ.'INDEF')WRITE(1,120)
120    FORMAT(//,30('*'),/,' The system has mixed damping',/,
     & 30('*'))
      IF(TYPE.EQ.'ZERO')WRITE(1,130)
130    FORMAT(//,33('*'),/,' The system is critically damped',/,
     & 33('*'))
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      RETURN
      END
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                         Subroutine EIGQUE                           C
C                                                                     C
C       EIGQUE asks the user if he wishes to compute the damped      C
C    eigenvalues.                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      SUBROUTINE EIGQUE(DEVAL)
C                                                                     C
C                                                                     C
      CHARACTER*1 ANS(3)
      LOGICAL DEVAL, UNREC
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      UNREC=.TRUE.
      PRINT*,' '
      PRINT*,'Do you want to compute the damped eigenvalues?'
      WHILE(UNREC)DO
        READ(1,'(3A1)')ANS
        CALL UPCASE(ANS)
        IF(ANS(1).EQ.'Y')THEN
          DEVAL=.TRUE.
          UNREC=.FALSE.
        ELSEIF(ANS(1).EQ.'N')THEN
          DEVAL=.FALSE.
          UNREC=.FALSE.
        ELSE
          PRINT*,'Unrecognized character. Enter yes or no.'
          UNREC=.TRUE.
        ENDIF
      ENDWHILE
C                                                                     C
      RETURN
      END
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                      Subroutine DMPEIG                              C
C                                                                     C
C       The subroutine DMPEIG calculates the damped eigenvalues.      C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C      *Variable List*                                                C
C       N - System or matrix size                                     C
C       N2 - Two times the size N                                     C
C       NN - Workspace size of 2*N2*N2                                C
C       M - Mass matrix as inputted                                   C
C       K - Stiffness matrix as inputted                              C
C       C - Damping matrix as inputted                                C
C       LAMDA - Damped eigenvalues vector                             C
C       BETA - Output of EIGZF used to determine eigenvalues          C
C       WK - Workspace needed by EIGZF                                C
C       DM - Damped mass matrix, see below                            C
C       DK - Damped stiffness matrix, see below                       C
C                                                                     C
C                   _         _                  _         _          C
C                  |     |     |                |     |     |         C
C                  |  O  |  M  |                | -M  |  O  |         C
C          DM  =   |-----|-----|        DK  =   |-----|-----|         C
C                  |  M  |  C  |                |  O  |  K  |         C
C                  |_    |    _|                |_    |    _|         C
C                                                                     C
C                                                                     C
C                                                                     C
C       UCOMP - Damped eigenvectors (not used)                        C
C       EIGZF - Library program that computes eigenvalues             C
C               and eigen vectors                                     C
C                                                                     C
C                                                                     C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      SUBROUTINE DMPEIG(M,C,K,LAMDA,N,N2,NN,BETA,WK,DM,DK,UCOMP)
C                                                                     C
C                                                                     C
      REAL*8 M(N,N),C(N,N),K(N,N),BETA(N2),WK(NN)
      REAL*8 DM(N2,N2),DK(N2,N2)
      COMPLEX*16 LAMDA(N2),UCOMP(N2,N2),LAMTMP
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C     The following loops construct the 2N by 2N mass and             C
C        stiffness matrices, DM and DK as shown above.                C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      DO 20 I=1,N
      DO 10 J=1,N
      II=N+I
      JJ=N+J
      DM(I,J)=0.
      DM(I,JJ)=M(I,J)
      DM(II,J)=M(I,J)
      DM(II,JJ)=C(I,J)
```

```
      DK(I,J)=-M(I,J)
      DK(I,JJ)=0.
      DK(II,J)=0.
      DK(II,JJ)=K(I,J)
10     CONTINUE
20     CONTINUE
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C    Send the DM and DK matrices to EIGZF which computes the          C
C       values LAMDA and BETA which define the eigenvalues.           C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      CALL EIGZF(DK,N2,DM,N2,N2,2,LAMDA,BETA,UCOMP,N2,WK,IER)
C                                                                     C
      DO 40 I =1,N2
      IF(DABS(BETA(I)).LT.1.D-10)THEN
        LAMDA(I)=0
      ELSE
        LAMDA(I)=LAMDA(I)/BETA(I)
      ENDIF
40     CONTINUE
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C        Separate the real from the complex eigenvalues,              C
C            the real going to the work array WK and the              C
C            complex reforming the vector LAMDA                       C
C        NR is the number of real eigenvalues                         C
C        NC is the number of complex eigenvalues                      C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      NR=0
      NC=0
      DO 100 I=1,N2
      IF(DABS(DIMAG(LAMDA(I))).LT.(1.E-30*DREAL(LAMDA(I))))THEN
        NR=NR+1
        WK(NR)=DREAL(LAMDA(I))
      ELSE
        NC=NC+1
        LAMDA(NC)=LAMDA(I)
      ENDIF
100     CONTINUE
      IF(NR.EQ.0)GO TO 140
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C          Order the real values from smallest to largest             C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
      DO 120 II=1,NR-1
      DO 110 I=1,NR-1
      IF(WK(I).LE.WK(I+1))GO TO 110
      WKTMP=WK(I)
      WK(I)=WK(I+1)
      WK(I+1)=WKTMP
```

```
110     CONTINUE
120     CONTINUE
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C        If all the values are real put them back into the           C
C             LAMDA vector and quit                                  C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      IF(NC.EQ.O)THEN
        DO 130 I=1,N2
        LAMDA(I)=DCMPLX(WK(I))
130     CONTINUE
      ENDIF
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C           Order the complex values from smallest magnitude to      C
C               largest by complex pair                              C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
140     DO 160 II=1,NC-1
      DO 150 I=1,NC-1
      IF(CDABS(LAMDA(I)).LE.CDABS(LAMDA(I+1)))GO TO 150
      LAMTMP=LAMDA(I)
      LAMDA(I)=LAMDA(I+1)
      LAMDA(I+1)=LAMTMP
150    CONTINUE
160    CONTINUE
C                                                                    C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C        If all the values are complex then quit                     C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      IF(NC.EQ.N2)GO TO 180
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C         Place the real values back into the LAMDA vector after     C
C             the complex, but rewrite them as complex numbers       C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      DO 170 I=NC+1,N2
      II=I-NC
      LAMDA(I)=DCMPLX(WK(II))
170    CONTINUE
180    CONTINUE
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C         Since EIGZF outputs the eigenvalues as positive values,    C
C              change them to the more common negative.              C
C                                                                    C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      DO 200 I=1,N2
      LAMDA(I)=-LAMDA(I)
200   CONTINUE
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C                                                                    C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
      END
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                        Subroutine OPNFIL                          C
C                                                                   C
C    OPNFIL open a file for output values.                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
C    *Variable List*                                                C
C     LOGICALS - .TRUE. meanings                                    C
C       OPEN - file opened for output values                        C
C       EX - filename already exists                                C
C       REENT - need to reenter filename                            C
C       UNREC - unrecognized character (i.e. not yes or no)         C
C       LEGAL - legal filename                                      C
C     FILEN - filename                                              C
C     ANS1,ANS2 - yes or no answers                                 C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      SUBROUTINE OPNFIL(OPEN,FILEN)
C                                                                   C
      LOGICAL OPEN,EX,REENT,UNREC,LEGAL
      CHARACTER*1 ANS1(3),ANS2(3)
      CHARACTER*32 FILEN
C                                                                   C
      REENT=.TRUE.
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
      PRINT*,' '
      PRINT*,'Would you like to open a file in which to place'
      PRINT*,'       system matrices and output values?'
5     READ(1,'(3A1)')ANS1
      CALL UPCASE(ANS1)
      IF(ANS1(1).EQ.'Y')THEN
       WHILE(REENT)DO
        PRINT*,'Enter filename'
        READ(1,'(A32)')FILEN
C                                                                   C
        LEGAL=.TRUE.
        IF(FILEN(1:1).GT.'0'.AND.FILEN(1:1).LT.'9')THEN
          PRINT*,'Illegal filename'
          LEGAL=.FALSE.
        ENDIF
C                                                                   C
        UNREC=.TRUE.
        IF(LEGAL)THEN
         INQUIRE(FILE=FILEN,EXIST=EX)
         IF(EX)THEN
          PRINT*,'File already exists.  Do you want to overwrite it?'
          WHILE(UNREC)DO
           READ(1,'(3A1)')ANS2
           CALL UPCASE(ANS2)
           IF(ANS2(1).EQ.'Y')THEN
            REENT=.FALSE.
            UNREC=.FALSE.
           ELSEIF(ANS2(1).EQ.'N')THEN
            REENT=.TRUE.
            UNREC=.FALSE.
           ELSE
```

```
            PRINT*,'Unrecognized character.  Enter yes or no.'
            UNREC=.TRUE.
           ENDIF
          ENDWHILE
         ELSE
          REENT=.FALSE.
         ENDIF
        ENDIF
       ENDWHILE
       OPEN(12,FILE=FILEN)
       OPEN=.TRUE.
      ELSEIF(ANS1(1).EQ.'N')THEN
       OPEN=.FALSE.
      ELSE
       PRINT*,'Unrecognized character.  Enter yes or no.'
       GO TO 5
      ENDIF
C                                                                     C
C                                                                     C
      RETURN
      END
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C                        Subroutine OUTFIL                        C
C                                                                 C
C      OUTFIL presents the user with the menu of options that     C
C           can be printed to the output file.                    C
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C                                                                 C
      SUBROUTINE OUTFIL(M,C,K,CCR,U,W,N,TYPE,LAMDA,N2,DEVAL)
C                                                                 C
C                                                                 C
      REAL*8 M(N,N),C(N,N),K(N,N),CCR(N,N),U(N,N),W(N)
      COMPLEX*16 LAMDA(N2)
      CHARACTER*1 RESP(16)
      CHARACTER*6 TYPE
      INTEGER COL(8)
      CHARACTER*32 TEXT
      LOGICAL DEVAL,CONT
C                                                                 C
      CONT=.TRUE.
C                                                                 C
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C                                                                 C
      WRITE(12,'(//)')
      IF(TYPE.EQ.'ZERO')THEN
       WRITE(12,10)
       WRITE(12,*) 'Critically damped system'
       WRITE(12,10)
      ELSEIF(TYPE.EQ.'NEGDEF')THEN
       WRITE(12,20)
       WRITE(12,*) 'Underdamped system'
       WRITE(12,20)
      ELSEIF(TYPE.EQ.'INDEF')THEN
       WRITE(12,30)
       WRITE(12,*) 'System with mixed damping'
       WRITE(12,30)
      ELSEIF(TYPE.EQ.'POSDEF')THEN
       WRITE(12,40)
       WRITE(12,*) 'Overdamped system'
       WRITE(12,40)
      ENDIF
10    FORMAT(26('*'))
20    FORMAT(20('*'))
30    FORMAT(27('*'))
40    FORMAT(19('*'))
C                                                                 C
      PRINT*,'The following information can be printed'
      PRINT*,'     to your output file.'
      PRINT*,'     1)Mass Matrix'
      PRINT*,'     2)Stiffness Matrix'
      PRINT*,'     3)Damping Matrix'
      PRINT*,'     4)Critical Damping Matrix'
      PRINT*,'     5)Undamped Natural Frequencies'
      PRINT*,'     6)Modal Matrix'
      IF(DEVAL)PRINT*,'     7)Damped Eigenvalues'
      PRINT*,' '
      PRINT*,'Enter the option numbers desired in a single line'
      PRINT*,'  separated by commas.  '
      PRINT*,' '
C                                                                 C
```

```
      READ(1,'(16A1)')(RESP(II),II=1,16)
      II=1
      WHILE(CONT)DO
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   Write Mass matrix                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF(RESP(II).EQ.'1')THEN
      WRITE(12,'( )')
       WRITE(12,*) 'MASS MATRIX'
      NI=(N/5)*5
      IF(NI.EQ.N)NBLK=N/5
      IF(NI.NE.N)NBLK=N/5+1
      DO 100 IB=1,NBLK
      IF(N.LE.(5*IB))THEN
        NCOL=N
        NUM=5-(5*IB-N)
      ELSEIF(N.GT.(5*IB))THEN
        NCOL=5*IB
        NUM=5
      ENDIF
      DO 110 J=5*IB-4,NCOL
      JJ=J-5*(IB-1)
      COL(JJ)=J
110   CONTINUE
      IF(IB.EQ.1)THEN
        WRITE(TEXT,900)NUM
        WRITE(12,TEXT)
        WRITE(TEXT,910)NUM
        WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
      ELSE
        WRITE(TEXT,920)NUM
        WRITE(12,TEXT)
        WRITE(TEXT,930)NUM
        WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
      ENDIF
      DO 120 I=1,N
      IF(IB.EQ.1)THEN
        WRITE(12,940)(M(I,J),J=1,NCOL)
      ELSE
        WRITE(12,950)(M(I,J),J=5*IB-4,NCOL)
      ENDIF
120   CONTINUE
      WRITE(12,'( )')
100   CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  Write Stiffness matrix                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSEIF(RESP(II).EQ.'2')THEN
      WRITE(12,'( )')
      WRITE(12,*) 'STIFFNESS MATRIX'
      NI=(N/5)*5
      IF(NI.EQ.N)NBLK=N/5
      IF(NI.NE.N)NBLK=N/5+1
      DO 200 IB=1,NBLK
      IF(N.LE.(5*IB))THEN
        NCOL=N
        NUM=5-(5*IB-N)
      ELSEIF(N.GT.(5*IB))THEN
        NCOL=5*IB
```

```
          NUM=5
        ENDIF
        DO 210 J=5*IB-4,NCOL
        JJ=J-5*(IB-1)
        COL(JJ)=J
210     CONTINUE
        IF(IB.EQ.1)THEN
          WRITE(TEXT,900)NUM
          WRITE(12,TEXT)
          WRITE(TEXT,910)NUM
          WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
        ELSE
          WRITE(TEXT,920)NUM
          WRITE(12,TEXT)
          WRITE(TEXT,930)NUM
          WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
        ENDIF
        DO 220 I=1,N
        IF(IB.EQ.1)THEN
          WRITE(12,940)(K(I,J),J=1,NCOL)
        ELSE
          WRITE(12,950)(K(I,J),J=5*IB-4,NCOL)
        ENDIF
220     CONTINUE
        WRITE(12,'( )')
200     CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C    Write Damping matrix                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        ELSEIF(RESP(II).EQ.'3')THEN
         WRITE(12,'( )')
         WRITE(12,*) 'DAMPING MATRIX'
        NI=(N/5)*5
        IF(NI.EQ.N)NBLK=N/5
        IF(NI.NE.N)NBLK=N/5+1
        DO 300 IB=1,NBLK
        IF(N.LE.(5*IB))THEN
          NCOL=N
          NUM=5-(5*IB-N)
        ELSEIF(N.GT.(5*IB))THEN
          NCOL=5*IB
          NUM=5
        ENDIF
        DO 310 J=5*IB-4,NCOL
        JJ=J-5*(IB-1)
        COL(JJ)=J
310     CONTINUE
        IF(IB.EQ.1)THEN
          WRITE(TEXT,900)NUM
          WRITE(12,TEXT)
          WRITE(TEXT,910)NUM
          WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
        ELSE
          WRITE(TEXT,920)NUM
          WRITE(12,TEXT)
          WRITE(TEXT,930)NUM
          WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
        ENDIF
        DO 320 I=1,N
```

```
      IF(IB.EQ.1)THEN
        WRITE(12,940)(C(I,J),J=1,NCOL)
      ELSE
        WRITE(12,950)(C(I,J),J=5*IB-4,NCOL)
      ENDIF
320   CONTINUE
      WRITE(12,'( )')
300   CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   Write Critical damping matrix                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSEIF(RESP(II).EQ.'4')THEN
      WRITE(12,'( )')
      WRITE(12,*) 'CRITICAL DAMPING MATRIX'
      NI=(N/5)*5
      IF(NI.EQ.N)NBLK=N/5
      IF(NI.NE.N)NBLK=N/5+1
      DO 400 IB=1,NBLK
      IF(N.LE.(5*IB))THEN
        NCOL=N
        NUM=5-(5*IB-N)
      ELSEIF(N.GT.(5*IB))THEN
        NCOL=5*IB
        NUM=5
      ENDIF
      DO 410 J=5*IB-4,NCOL
      JJ=J-5*(IB-1)
      COL(JJ)=J
410   CONTINUE
      IF(IB.EQ.1)THEN
        WRITE(TEXT,900)NUM
        WRITE(12,TEXT)
        WRITE(TEXT,910)NUM
        WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
      ELSE
        WRITE(TEXT,920)NUM
        WRITE(12,TEXT)
        WRITE(TEXT,930)NUM
        WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
      ENDIF
      DO 420 I=1,N
      IF(IB.EQ.1)THEN
        WRITE(12,940)(CCR(I,J),J=1,NCOL)
      ELSE
        WRITE(12,950)(CCR(I,J),J=5*IB-4,NCOL)
      ENDIF
420   CONTINUE
      WRITE(12,'( )')
400   CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   Write undamped natural frequencies                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSEIF(RESP(II).EQ.'5')THEN
       WRITE(12,'( )')
       WRITE(12,*) 'UNDAMPED NATURAL FREQUENCIES (rad/sec)'
       WRITE(12,*) ' '
       DO 500 I=1,N
       WRITE(12,510)I,W(I)
510    FORMAT(I2,1X,E13.6)
```

```
500      CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   Write Modal matrix                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSEIF(RESP(II).EQ.'6')THEN
       WRITE(12,'( )')
      WRITE(12,*) 'EIGENVECTORS (MODAL MATRIX)'
      NN=(N/8)*8
      IF(NN.EQ.N)NBLK=N/8
      IF(NN.NE.N)NBLK=N/8+1
      DO 600 IJ=1,NBLK
      IF(N.LE.(8*IJ))THEN
       NCOL=N
       NUM=8-(8*IJ-N)
      ELSEIF(N.GT.(8*IJ))THEN
       NCOL=8*IJ
       NUM=8
      ENDIF
      WRITE(TEXT,650)NUM
      WRITE(12,TEXT)
      DO 610 J=8*IJ-7,NCOL
      JJ=J-8*(IJ-1)
      COL(JJ)=J
610     CONTINUE
      WRITE(TEXT,660)NUM
      WRITE(12,TEXT)(COL(JJ),JJ=1,NUM)
       DO 620 I=1,N
       WRITE(12,670)(U(I,J),J=8*IJ-7,NCOL)
620         CONTINUE
600       CONTINUE
650      FORMAT('(/,',I1,'(3X,6HColumn))')
660     FORMAT('(4X,',I1,'(1H(,I2,1H),5X))')
670      FORMAT(8F9.4)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   Write Damped Eigenvalues                                            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSEIF(RESP(II).EQ.'7')THEN
       IF(DEVAL)THEN
        WRITE(12,'( )')
        WRITE(12,*) 'DAMPED EIGENVALUES (Unpaired) '
        DO 700 I=1,N2
        WRITE(12,710)I,LAMDA(I)
710      FORMAT(I3,1X,E13.6,'+',E13.6,'i')
700      CONTINUE
       ELSE
        PRINT*,'**You did not chose to compute damped eigenvalues**'
       ENDIF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  Read delineators                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSEIF(RESP(II).EQ.' '.OR.RESP(II).EQ.',')THEN
       II=II-1
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  Unrecognized character                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ELSE
       PRINT*,'Unrecognized character'
       CONT=.FALSE.
C                                                                       C
```

```
      ENDIF
C                                                                    C
      II=II+2
      IF(II.GE.16)CONT=.FALSE.
      ENDWHILE
C                                                                    C
      WRITE(12,'( )')
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
900   FORMAT('(/,',I1,'(4X,6HColumn,4X))')
910   FORMAT('(',I1,'(5X,1H(,I2,1H),5X))')
920   FORMAT('(/,3X,',I1,'(4X,6HColumn,4X))')
930   FORMAT('(3X,',I1,'(5X,1H(,I2,1H),5X))')
940   FORMAT(5(E13.6,1X))
950   FORMAT(2X,5(1X,E13.6))
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      RETURN
       END
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                          Subroutine RERUN                          C
C                                                                    C
C        RERUN asks the user if he wishes to continue and look at    C
C  a new problem using a different damping matrix.                   C
C  CONT is true if the answer is yes.                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      SUBROUTINE RERUN(CONT)
C                                                                    C
      CHARACTER*1 ANS(3)
      LOGICAL CONT
C                                                                    C
      PRINT*,' '
      PRINT*,' '
      PRINT*,'Do you wish to run again with a different C matrix?'
      READ(1,'(3A1)')ANS
      CALL UPCASE(ANS)
      WHILE(ANS(1).NE.'Y'.AND.ANS(1).NE.'N')DO
        PRINT*,'Unrecognized character. Answer yes or no.'
        READ(1,'(3A1)')ANS
        CALL UPCASE(ANS)
      ENDWHILE
      IF(ANS(1).EQ.'Y')CONT=.TRUE.
      IF(ANS(1).EQ.'N')CONT=.FALSE.
C                                                                    C
      RETURN
      END
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                          Subroutine CHKSYM                         C
C                                                                    C
C   CHKSYM determines if the matrix MAT is symmetric.                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
      SUBROUTINE CHKSYM(MAT,N,SYM)
C                                                                    C
      REAL*8 MAT(N,N)
      LOGICAL SYM
C                                                                    C
      SYM=.TRUE.
C                                                                    C
      I=1
      WHILE(SYM.AND.I.LE.N)DO
       J=I+1
       WHILE(SYM.AND.J.LE.N)DO
        IF(DABS(MAT(I,J)-MAT(J,I)).GT.1.E-30)SYM=.FALSE.
        J=J+1
       ENDWHILE
       I=I+1
      ENDWHILE
C                                                                    C
      RETURN
      END
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                          Subroutine PDCHK                            C
C                                                                      C
C   PDCHK determines if a matrix MAT is positive definite.             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      SUBROUTINE PDCHK(MAT,N,TM,NN,PD)
C                                                                      C
      REAL*8 MAT(N,N),TM(NN),DPR,DET
      LOGICAL PD
C                                                                      C
      PD=.TRUE.
C                                                                      C
      MS=1
      WHILE(MS.LE.N.AND.PD)DO
        CALL DETER(DET,MAT,TM,MS,N,NN,1,DO)
        CALL DIAGPR(MAT,N,MS,DPR)
        IF(DET/DPR.LT.1.E-12)PD=.FALSE.
        MS=MS+1
      ENDWHILE
C                                                                      C
      RETURN
      END
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                          Subroutine PSDCHK                           C
C                                                                      C
C  PSDCHK determines if a matrix MAT is positive semi-definite.        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      SUBROUTINE PSDCHK(MAT,N,TM,NN,PSD)
C                                                                      C
      REAL*8 MAT(N,N),TM(NN),DPR,DET
      LOGICAL PSD
C                                                                      C
      PSD=.TRUE.
C                                                                      C
      MS=1
      WHILE(MS.LE.N.AND.PSD)DO
        CALL DETER(DET,MAT,TM,MS,N,NN,1,DO)
        CALL DIAGPR(MAT,N,MS,DPR)
        IF(DET/DPR.LT.-1.E-12)PSD=.FALSE.
        MS=MS+1
      ENDWHILE
C                                                                      C
      RETURN
      END
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                          Subroutine DETER                            C
C                                                                      C
C   DETER computes the determinant of a matrix MAT.                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      SUBROUTINE DETER(DET,MAT,TM,MS,N,NN,C)
      REAL*8 MAT(N,N),TM(NN),DET,C
C                                                                      C
      KK=1
      DO 10 J=1,MS
      DO 20 I=1,MS
      TM(KK)=C*MAT(I,J)
      KK=KK+1
20     CONTINUE
10     CONTINUE
      CALL DMDET(DET,TM,MS,I1,I2,I3,I4,IERR)
C                                                                      C
      RETURN
      END
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                          Subroutine DIAGPR                           C
C                                                                      C
C        DIAGPR computes the product of the nonzero diagonal of        C
C   the matrix MAT.                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
      SUBROUTINE DIAGPR(MAT,N,MS,DPR)
      REAL*8 MAT(N,N),DPR
C                                                                      C
      DPR=1.D0
      DO 10 I=1,MS
      IF(DABS(MAT(I,I)).GT.1.E-12)THEN
        DPR=DPR*MAT(I,I)
      ENDIF
10    CONTINUE
      DPR=DABS(DPR)
C                                                                      C
      RETURN
      END
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                        Subroutine TRMTYP                         C
C                                                                  C
C     TRMTYP determines if the terminal being used is one that     C
C         requires paging.  NP=1 if it Needs Paging.               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
      SUBROUTINE TRMTYP(NP)
C                                                                  C
      INTEGER IT(5)
      DATA IT/4014,4052,4006,4010,4114/
C                                                                  C
      CALL TERM$$(ITERM,ITYPE,IBAUD,IER)
      NP=0
      DO 10 J=1,5
      IF(ITYPE.EQ.IT(J))NP=1
10    CONTINUE
C                                                                  C
      RETURN
      END
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                        Subroutine PAGE                           C
C                                                                  C
C     PAGE pages the screen and resets the line counter NL         C
C          to zero.                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
      SUBROUTINE PAGE(NL)
C                                                                  C
      CALL NEWPAG
      CALL SLEEP$(2000)
      NL=0
C                                                                  C
      RETURN
      END
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```