Fortran programs for a 3-D four-noded curved beam element: ARCHCODE version 1.0

Jing Li

Appendix to the thesis

A geometrically exact curved beam theory and its finite element formulation/implementation

submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Rakesh K. Kapania, Chair
Eric R. Johnson
Raymond H. Plaut

December 28, 2000
Blacksburg, Virginia

Keywords: Curved beam, geometrically exact nonlinear, finite rotations, finite element.

©Copyright 2000, Jing Li

Fortran programs for a 3-D four-noded curved beam element:

ARCHCODE Version 1.0

Jing Li
Rakesh K. Kapania

Aerospace and Ocean Engineering
Virginia Polytechnic Institute and State University

Table of Contents

1. Introduction

The ARCHCODE is a special purpose Fortran program using a 3-D four-noded curved beam element (and 1-DOF hinge model) based on a geometrically exact curved beam theory with rigid cross-section assumption.

The beam element has the following features:

(i) can be used for 3-D natural curved/twisted slender and thick beams;
(ii) has finite strains and finite displacements/rotations;
(iii) can handle classical coupled cross-section material constants with an initial curvature correction term; and
(iv) uses iso-parametric Lagrangian interpolation with uniform reduced integration.

However, the ARCHCODE Version 1.0 handles only slender beams and cross-section material constants without coupling.

The ARCHCODE Version 1.0 is used to simulate the static responses, such as, nodal displacements, buckling and post-buckling responses, as well as search for the lowest nonlinear and linearized bifurcation (branching) point.

The loads considered now are point loads and distributed loads (such as, self-weight, snow load, and pressure load, as given by load density per arc-length of the beam).

2. Lists of ARCHCODE version 1.0 Fortran source files and data files

Make sure that **all** files are put into the same subdirectory or folder.

The ARCHCODE Version 1.0 includes the following Fortran 77 source files:
analy0.f
analy1.f
analy2.f
analy3.f
analy4.f
analy5.f
analy6.f
eigenrgrgg.f

The ARCHCODE Version 1.0 has the following data files as opened in **analy0.f** as follows:

```
OPEN (UNIT=5,FILE='archcode.in', STATUS='OLD')
OPEN (UNIT=6,FILE='archcode.out', STATUS='UNKNOWN')
OPEN (UNIT=8,FILE='archu', STATUS='UNKNOWN ')
OPEN (UNIT=9,FILE='archv', STATUS='UNKNOWN ')
OPEN (UNIT=11,FILE='archw', STATUS='UNKNOWN ')
OPEN (UNIT=13,FILE='archrx', STATUS='UNKNOWN ')
OPEN (UNIT=14,FILE='archry', STATUS='UNKNOWN ')
OPEN (UNIT=15,FILE='archrz', STATUS='UNKNOWN ')
OPEN (UNIT=10,FILE='bifurcation.out', status = 'UNKNOWN ')
OPEN (UNIT=12,FILE='archdet', status = 'UNKNOWN ')
OPEN (UNIT=18,FILE='archshape', STATUS='UNKNOWN')
```

archcode.in:
The main one input data file.

archcode.out:
The main output data file.

archu, archv, archw, archrx, archry, archrz:
Output data files, storing the translation displacements in X, Y, Z directions, and rotation displacement (in rad) around X, Y, Z axes, respectively, for a specified node as given interactively when the ARCHCODE is run. Those data are given in TECPLOT format .

bifurcation.out:
Output data file, storing the information during the process of searching bifurcation point or the eigen-problem solution information when linearized bifurcation point is evaluated.

archdet:
Output data file, storing the relative determinants of the tangential stiffness matrix via loading factors during the process of searching bifurcation point. The data are given in TECPLOT format.

archshape:
Output data file, storing the nodal coordinates of the undeformed and deformed curved beam/arch/frame for each load step. The data are given in TECPLOT format.

3. How to compile and run the ARCHCODE version 1.0?

Here the instructions for compiling and runing the ARCHCODE version 1.0 are given for unix workstation (IRIX 6.2).

To compile, type:

f77 -col120 analy0.f analy1.f analy2.f analy3.f analy4.f analy5.f analy6.f eigenrgrgg.f

then press ENTER or RETURN key.

Rename the execution file **a.out** to **archrun**:

mv a.out archrun

then press ENTER or RETURN key.

When the input data file **archcode.in** has been prepared, the ARCHCODE version 1.0 can be run by simply typing:

archrun

then press ENTER or RETURN key.

The ARCHCODE version 1.0 will ask you to enter some optional input information:

Choose the problem to be solved:

1 - for load control (with bifurcation analysis)
2 - for arclength control (with bifurcation analysis)
3 - for linearized bifurcation analysis

For load control option, for example, type:

1

and then press ENTER or RETURN key. You will be asked to give further information:

Need to locate bifurcation point ??
if yes, enter 2; if no, enter 1:

Type

2

to locate bifurcation point if applicable, then press ENTER or RETURN key; or type

1

if you don't want to locate bifurcation point, then press ENTER or RETURN key.

And so on.

4. The format of the input file: archcode.in

Note that free field format is used in **archcode.in**. Data fields are separated by commas or blanks.

In the following tables, the first column is the comment, which does not appear in the file, the data fields start from the second column.

Data set 1: General information of the problem

1: Title

Format	Title
Example	Post-buckling analysis of a clamped-hinged circular arch

Title: any string of ascii codes; the total string length is less than 80

2: The scale of the finite element mesh

Format	NTNODE	NELM
Example	19	6

NTNODE: integer, the total number of nodes
NELM: integer, the total number of elements

3: The number of 1 DOF hinges

Format	NHINGE1
Example	0

NHINGE1: integer, the total number of 1 DOF hinges

Data set 2: Nodal coordinates

Format	X	Y	Z
Example	95.3717	-30.0706	0.0000

X, Y, Z: real, the X, Y, Z coordinates in the global coordinate system. Each line consists of the coordinates of one node. The total number of record lines for the coordinates of the nodes is NTNODE, in the order of the nodes numbered, from 1 to NTNODE.

Data set 3: Element connection matrix: MELM

Format	Node 1	Node 2	Node 3	Node 4
Example	1	2	3	4

Node I: integer, the corresponding global ID number of I-*th* node in the beam element; I = 1, 2, 3, 4;
Each line is for one element. The total number of record lines for the element connection matrix is NELM, in the order of the elements numbered, from 1 to NELM.

Data set 4: 1 DOF hinge connection matrix: MHING1

Format	P_node	Q_node
Example	19	20

P _node: integer, the corresponding global ID number of Master node of the hinge;
Q_node: integer, the corresponding global ID number of Slave node of the hinge;
Each line is for one hinge. The total number of record lines for the hinge connection matrix is NHINGE1, in the order of the hinges numbered, from 1 to NHINGE1. If there is no 1 DOF hinge at all, i.e., NHINGE1 = 0, there will no need to fill in this data set.

Note that the 1 DOF hinge is that two beam members are connected at a spatial point , say, S, which is on the center line of the both beam members; the two beam members can only have a relative rotation around an axis, say H, through the spatial point S; the orientation of the axis H is fixed relative to the two beam members. When a finite element model is created, two nodes are created through that spatial point, and one node, say P, is fixed at one of the beam members, and the other, say Q, is fixed at the other beam member.

Data set 5: 1 DOF hinge direction matrix: HING1

Format	Hx	Hy	Hz	K
Example	0.	0.	1.	10.00

Hx, Hy, Hz: real, the X, Y, Z components of the direction vector H for the 1 DOF hinge.
K: real, the torsional spring constant of the 1 DOF hinge.
Each line is for one hinge. The total number of record lines for the hinge direction matrix is NHINGE1, in the order of the hinges numbered, from 1 to NHINGE1. If there is no 1 DOF hinge at all, i.e., NHINGE1 = 0, there will no need to fill in this data set.

Data set 6: Material constant matrices: CN, CM

The material constants of a 4-noded beam element at the first node:

Format	EA	GAn	GAb	GJt	EIn	EIb
Example	1.E8	1.E8	1.E8	1.E8	1.E7	1.E6

EA: real, the extensional stiffness associated with normal force
GAn: real, the shearing stiffness associated with shearing force along the normal direction
GAb: real, the shearing stiffness associated with shearing force along the bi-normal direction
GJt: real, the torsional stiffness associated with torque around the tangent direction
EIn: real, the bending stiffness associated with bending moment around the normal direction

EIb: real, the bending stiffness associated with bending moment around the bi-normal direction

The same format for the second, third, and fourth node. Each line is for one node.
Four lines are for four nodes in one element.

The above are repeated NELM times, consisting of 4*NELM record lines, in the order of the elements numbered.

Data set 7: Prescribed zero displacements

1: The total number of prescribed zero displacements:

Format	NPVBC
Example	62

NPVBC: integer, the total number of prescribed zero displacements

2: Information for prescribed zero displacements

Format	INOD	IDOF
Example	1	1

INOD, IDOF: integer, they indicate that the IDOF *-th* component of the INOD *-th* node has prescribed zero displacement.
IDOF = 1, translation component in X direction
2, translation component in Y direction
3, translation component in Z direction
4, rotational component around X direction
5, rotational component around Y direction
6, rotational component around Z direction
(Note: right-handed screw rule is used.)
Each line is for one zero displacement component. The total number of record lines for the zero displacement components is NPVBC, in any order that is convenient.

Data set 8: Prescribed non-zero point loads

1: The total number of prescribed non-zero point loads:

Format	NSVBC
Example	62

NSVBC: integer, the total number of prescribed non-zero point load components

2: The data of prescribed non-zero point loads

Format	INOD	IDOF	ALOD
Example	10	2	-1000.

INOD, IDOF: integer, they indicate that the IDOF *-th* component of the INOD *-th* node has prescribed non-zero point load
IDOF = 1, translation component in X direction

2, translation component in Y direction
3, translation component in Z direction
4, rotational component around X direction
5, rotational component around Y direction
6, rotational component around Z direction
(Note: right-handed screw rule is used.)

ALOD: real, point load value

Note: No. of the component = 1: Px
= 2: Py
= 3: Pz
= 4: Mx
= 5: My
= 6: Mz

Otherwise, wrong data for point loads!

Each line is for one non-zero point load component. The total number of record lines for the non-zero point load components is NSVBC, in any order that is convenient.
If there is no non-zero point load at all, i.e., NSVBC = 0, there will no need to fill in this data set.

Data set 9: Non-zero weight density

1: The total number of non-zero weight elements

Format	NSVBCG
Example	6

NSVBCG: integer, the total number of nonzero weight elements.

2: The weight density of a 4-noded beam element

2.1 The ID number of the beam element

Format	IDELEM
Example	4

IDELEM: The ID number of the beam element

2.2 The weight density of a 4-noded beam element for the first node:

Format	X-comp	Y-comp	Z-comp
Example	0.	-100.	0.

The same format for the second, third, and fourth node. Each line is for one node.
Four lines are for four nodes in one element.

2.1 and 2.2 are repeated NSVBCG times, consisting of 5*NSVBCG record lines, in any order that is convenient.

If there is no non-zero weight elements at all, i.e., NSVBCG = 0, there will no need to fill in this data set.

Data set 10: Non-zero direction fixed distributed load density

1: The total number of non-zero direction fixed distributed load elements

Format	NSVBCN
Example	6

NSVBCN: integer, the total number of nonzero direction fixed distributed load density elements

2: The load density of a 4-noded beam element
2.1 The ID number of the beam element

Format	IDELEM
Example	4

IDELEM: The ID number of the beam element

2.2 The load density of a 4-noded beam element for the first node:

Format	X-comp	Y-comp	Z-comp
Example	0.	-100.	0.

The same format for the second, third, and fourth node.
Four lines are for four nodes in one element.

2.1 and 2.2 are repeated NSVBCN times, consisting of 5*NSVBCN record lines, in any order that is convenient.

If there is no non-zero weight elements at all, i.e., NSVBCN = 0, there will no need to fill in this data set.

Data set 11: Non-zero pressure load density

1: The total number of nonzero pressure load elements

Format	NSVBCP
Example	6

NSVBCP: integer, the total number of nonzero pressure load density elements

2: The load density of a 4-noded beam element

2.1 The ID number of the beam element

Format	IDELEM
Example	4

IDELEM: The ID number of the beam element

2.2 The load density of a 4-noded beam element for the first node:

Format	X-comp	Y-comp	Z-comp
Example	0.	-100.	0.

The same format for the second, third, and fourth node.
Four lines are for four nodes in one element.

2.1 and 2.2 are repeated NSVBCP times, consisting of 5*NSVBCP record lines, in any order that is convenient.

If there is no non-zero weight elements at all, i.e., NSVBCP = 0, there will no need to fill in this data set.

5. Some sample archcode.in files for the examples in the thesis

Note that in all the examples for **archcode.in** file, the comment lines starting with **Data set** do not appear in the real **archcode.in** file.

Example 1 Unrolling and re-rolling of a circular cantilever beam

Data set 1: General information of the problem
Unrolling and re-rolling of a circular cantilever beam
13 4
0
Data set 2: Nodal coordinates
0.0000000000000000E+00 1.591549430918954 0.0000000000000000E+00
0.7957747154594766 1.378322238554480 0.0000000000000000E+00
1.378322238554480 0.7957747154594770 0.0000000000000000E+00
1.591549430918954 9.7454295812984397E-17 0.0000000000000000E+00
1.378322238554480 -0.7957747154594764 0.0000000000000000E+00
0.7957747154594773 -1.378322238554480 0.0000000000000000E+00
1.9490859162596879E-16 -1.591549430918954 0.0000000000000000E+00
-0.7957747154594764 -1.378322238554480 0.0000000000000000E+00
-1.378322238554480 -0.7957747154594774 0.0000000000000000E+00
-1.591549430918954 -2.9236288743895316E-16 0.0000000000000000E+00
-1.378322238554481 0.7957747154594756 0.0000000000000000E+00
-0.7957747154594774 1.378322238554480 0.0000000000000000E+00
-3.8981718325193759E-16 1.591549430918954 0.0000000000000000E+00
Data set 3: Element connection matrix: MELM
1 2 3 4
4 5 6 7
7 8 9 10
10 11 12 13
Data set 4: 1 DOF hinge connection matrix: MHING1
Data set 5: 1 DOF hinge direction matrix: HING1
Data set 6: Material constant matrices: CN, CM
1200.000000000000 500.0000000000000 500.0000000000000
200.0000000000000 100.0000000000000 100.0000000000000
1200.000000000000 500.0000000000000 500.0000000000000
200.0000000000000 100.0000000000000 100.0000000000000
1200.000000000000 500.0000000000000 500.0000000000000
200.0000000000000 100.0000000000000 100.0000000000000
1200.000000000000 500.0000000000000 500.0000000000000
200.0000000000000 100.0000000000000 100.0000000000000
1200.000000000000 500.0000000000000 500.0000000000000
200.0000000000000 100.0000000000000 100.0000000000000
1200.000000000000 500.0000000000000 500.0000000000000
200.0000000000000 100.0000000000000 100.0000000000000

1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000
1200.000000000000	500.0000000000000	500.0000000000000
200.0000000000000	100.0000000000000	100.0000000000000

Data set 7: Prescribed zero displacements

6
1 1
1 2
1 3
1 4
1 5
1 6

Data set 8: Prescribed non-zero point loads

1
13 6 62.83185307179586

Data set 9: Non-zero weight density

0

Data set 10: Non-zero direction fixed distributed load density

0

Data set 11: Non-zero pressure load density

0

Example 2 Post-buckling analysis of a clamped-hinged circular arch

Data set 1: General information of the problem

Post-buckling analysis of a clamped-hinged circular arch

19 6

0

Data set 2: Nodal coordinates

95.37169507482270	-30.07057995042731	0.0000000000000000E+00
99.53027957931658	-9.681087070317911	0.0000000000000000E+00
99.37895171394428	11.12762131982996	0.0000000000000000E+00
94.92426435730339	31.45447561516136	0.0000000000000000E+00
86.35911671778987	50.41927170956703	0.0000000000000000E+00
74.05440131090047	67.20078605555223	0.0000000000000000E+00
58.54294337699407	81.07233671702122	0.0000000000000000E+00
40.49642820326738	91.43325053161793	0.0000000000000000E+00
20.69631545515058	97.83487377505476	0.0000000000000000E+00
2.8327694488239898E-14	100.0000000000000	0.0000000000000000E+00
-20.69631545515052	97.83487377505476	0.0000000000000000E+00
-40.49642820326733	91.43325053161794	0.0000000000000000E+00
-58.54294337699400	81.07233671702126	0.0000000000000000E+00
-74.05440131090042	67.20078605555229	0.0000000000000000E+00
-86.35911671778985	50.41927170956707	0.0000000000000000E+00
-94.92426435730337	31.45447561516141	0.0000000000000000E+00
-99.37895171394426	11.12762131983003	0.0000000000000000E+00
-99.53027957931660	-9.681087070317876	0.0000000000000000E+00
-95.37169507482271	-30.07057995042727	0.0000000000000000E+00

Data set 3: Element connection matrix: MELM

1	2	3	4
4	5	6	7
7	8	9	10
10	11	12	13
13	14	15	16
16	17	18	19

Data set 4: 1 DOF hinge connection matrix: MHING1

Data set 5: 1 DOF hinge direction matrix: HING1

Data set 6: Material constant matrices: CN, CM

100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000

100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000
100000000.0000000	100000000.0000000	100000000.0000000
100000000.0000000	10000000.00000000	1000000.000000000

Data set 7: Prescribed zero displacements

62

1 1
1 2
1 3
1 4
1 5

19 1
19 2
19 3
19 4
19 5
19 6
2 3
2 4
2 5
3 3
3 4
3 5
4 3
4 4
4 5
5 3
5 4
5 5
6 3
6 4
6 5
7 3
7 4
7 5
8 3
8 4
8 5
9 3
9 4
9 5
10 3
10 4
10 5
11 3
11 4
11 5
12 3
12 4
12 5
13 3
13 4
13 5
14 3
14 4
14 5
15 3

15 4
15 5
16 3
16 4
16 5
17 3
17 4
17 5
18 3
18 4
18 5

Data set 8: Prescribed non-zero point loads

1
10 2 -1000.000000000000

Data set 9: Non-zero weight density

0

Data set 10: Non-zero direction fixed distributed load density

0

Data set 11: Non-zero pressure load density

0

Example 3 45-degree bend

Data set 1: General information of the problem

45 degree bend

4 1

0

Data set 2: Nodal coordinates

100.0000000000000 0.0000000000000000E+00 0.0000000000000000E+00
96.59258262890683 25.88190451025207 0.0000000000000000E+00
86.60254037844388 49.99999999999999 0.0000000000000000E+00
70.71067811865476 70.71067811865474 0.0000000000000000E+00

Data set 3: Element connection matrix: MELM

1 2 3 4

Data set 4: 1 DOF hinge connection matrix: MHING1

Data set 5: 1 DOF hinge direction matrix: HING1

Data set 6: Material constant matrices: CN, CM

10000000.00000000 4166666.666666667 4166666.666666667
703000.0000000000 833333.3333333333 833333.3333333333
10000000.00000000 4166666.666666667 4166666.666666667
703000.0000000000 833333.3333333333 833333.3333333333
10000000.00000000 4166666.666666667 4166666.666666667
703000.0000000000 833333.3333333333 833333.3333333333
10000000.00000000 4166666.666666667 4166666.666666667
703000.0000000000 833333.3333333333 833333.3333333333

Data set 7: Prescribed zero displacements

6

1 1
1 2
1 3
1 4
1 5
1 6

Data set 8: Prescribed non-zero point loads

1

4 3 600.0000000000000

Data set 9: Non-zero weight density

0

Data set 10: Non-zero direction fixed distributed load density

0

Data set 11: Non-zero pressure load density

0

Example 4 Cantilever beam subjected to conservative and non-conservative distributed loads (the data are for conservative load case)

Data set 1: General information of the problem

Cantilever beam subjected to conservative and non-conservative distributed loads

```
        7        2
        0
```

Data set 2: Nodal coordinates

```
  0.0000000000000000E+00  0.0000000000000000E+00  0.0000000000000000E+00
  8.3333333333333329E-02  0.0000000000000000E+00  0.0000000000000000E+00
  0.1666666666666667      0.0000000000000000E+00  0.0000000000000000E+00
  0.2500000000000000      0.0000000000000000E+00  0.0000000000000000E+00
  0.3333333333333333      0.0000000000000000E+00  0.0000000000000000E+00
  0.4166666666666666      0.0000000000000000E+00  0.0000000000000000E+00
  0.0                     0.0000000000000000E+00  0.0000000000000000E+00
```

Data set 3: Element connection matrix: MELM

```
        1        2        3        4
        4        5        6        7
```

Data set 4: 1 DOF hinge connection matrix: MHING1

Data set 5: 1 DOF hinge direction matrix: HING1

Data set 6: Material constant matrices: CN, CM

```
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
   15000.00000000000     6249.999999999999     6249.999999999999
  0.2500000000000000     0.1250000000000000     0.1250000000000000
```

Data set 7: Prescribed zero displacements

```
 6
1  1
1  2
1  3
1  4
1  5
1  6
```

Data set 8: Prescribed non-zero point loads

0

Data set 9: Non-zero weight density

0

Data set 10: Non-zero direction fixed distributed load density

2

1

0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0

2

0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0

Data set 11: Non-zero pressure load density

2

1

0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

2

0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

Example 5 Bifurcation buckling of a circular arch with two hinges (h/L = 0.2)

Data set 1: General information of the problem

Bifurcation buckling of a circular arch with two hinges at ends

```
      15        4
       2
```

Data set 2: Nodal coordinates

```
  0.4999999999999999     0.5249999999999998     0.0000000000000000E+00
  0.4999999999999999     0.5249999999999998     0.0000000000000000E+00
  0.4295733585622031     0.5840305896214587     0.0000000000000000E+00
  0.3522453295933466     0.6336783314740015     0.0000000000000000E+00
  0.2692582403567252     0.6731456008918129     0.0000000000000000E+00
  0.1819453347593019     0.7017983294076193     0.0000000000000000E+00
  9.1709353883548561E-02 0.7191761914929206     0.0000000000000000E+00
  4.4393446469091548E-17 0.7249999999999999     0.0000000000000000E+00
 -9.1709353883548478E-02 0.7191761914929206     0.0000000000000000E+00
 -0.1819453347593018     0.7017983294076193     0.0000000000000000E+00
 -0.2692582403567250     0.6731456008918130     0.0000000000000000E+00
 -0.3522453295933465     0.6336783314740015     0.0000000000000000E+00
 -0.4295733585622031     0.5840305896214587     0.0000000000000000E+00
 -0.4999999999999998     0.5249999999999999     0.0000000000000000E+00
 -0.4999999999999998     0.5249999999999999     0.0000000000000000E+00
```

Data set 3: Element connection matrix: MELM

```
       2        3        4        5
       5        6        7        8
       8        9       10       11
      11       12       13       14
```

Data set 4: 1 DOF hinge connection matrix: MHING1

```
       1        2
      15       14
```

Data set 5: 1 DOF hinge direction matrix: HING1

```
 0.   0.   1.   0.
 0.   0.   1.   0.
```

Data set 6: Material constant matrices: CN, CM

```
  120000.0000000000     49999.99999999999     49999.99999999999
  2.000000000000000     25.00000000000000     1.000000000000000
  120000.0000000000     49999.99999999999     49999.99999999999
  2.000000000000000     25.00000000000000     1.000000000000000
  120000.0000000000     49999.99999999999     49999.99999999999
  2.000000000000000     25.00000000000000     1.000000000000000
  120000.0000000000     49999.99999999999     49999.99999999999
  2.000000000000000     25.00000000000000     1.000000000000000
  120000.0000000000     49999.99999999999     49999.99999999999
  2.000000000000000     25.00000000000000     1.000000000000000
  120000.0000000000     49999.99999999999     49999.99999999999
```

2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000
120000.0000000000	49999.99999999999	49999.99999999999
2.000000000000000	25.00000000000000	1.000000000000000

Data set 7: Prescribed zero displacements

12
1 1
1 2
1 3
1 4
1 5
1 6
15 1
15 2
15 3
15 4
15 5
15 6

Data set 8: Prescribed non-zero point loads

0

Data set 9: Non-zero weight density

0

Data set 10: Non-zero direction fixed distributed load density

0

Data set 11: Non-zero pressure load density

4
1
0.0 1.0 0.0
0.0 1.0 0.0

0.0 1.0 0.0
0.0 1.0 0.0
2
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0
3
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0
4
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0
0.0 1.0 0.0

6. ARCHCODE version 1.0 Fortran source files

```
C analy0.f:
C
c2345678901234567890123456789012345678901234567890123456789012345678901
2
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (MXNOD=300,MXELM=100,MXEBC=1212,MXNEQ=10000)
        DIMENSION XYZ(3*MXNOD),XYZN(3*MXNOD),X0(12*MXELM)
        DIMENSION MELM(4*MXELM),NODBC(6*MXNOD)
        DIMENSION mhing1(100),mhing3(100)
        DIMENSION hing1(400),idofh1(100)
        DIMENSION LM(24),MAXH(MXNEQ+1),MHT(MXNEQ+1)
        DIMENSION PTLOAD(6*MXNOD),DISTGG(12*MXELM),DISTNN(12*MXELM)
        DIMENSION DISTPP(12*MXELM)
        DIMENSION AST(290000),DISP(6*MXNOD)
        DIMENSION CNCM(24*MXELM)
        DIMENSION xnodel(12*MXELM), ttrg(54*MXELM),
     &           omegag(18*MXELM), da(6*MXNOD),dq(6*MXNOD),
     &           qext0(6*MXNOD),qint(6*MXNOD),
     &           da0(6*MXNOD),dq0(6*MXNOD),IPVT(6*MXNOD),WORK(6*MXNOD),
     &       atotl(6*MXNOD),atotl0(6*MXNOD),aext0(6*MXNOD),aint(6*MXNOD)
        DIMENSION t6rg(54*MXELM),om6gag(18*MXELM)
        DIMENSION st6rg(54*MXELM),som6gg(54*MXELM)
        DIMENSION sxnode(12*MXELM),sttrg(54*MXELM),
     &           somegg(18*MXELM),sda(6*MXNOD),sdq(6*MXNOD)

        DIMENSION WR(6*MXNOD),WI(6*MXNOD),Z(290000),LOC(6*MXNOD)
        DIMENSION FV1(6*MXNOD),IV1(6*MXNOD), AST1(290000),diag(6*MXNOD)

        write(*,*) ' Choose the problem to be solved: '
        write(*,*) '      '
        write(*,*) '  1 - for load control (with bifurcation analysis)
'
        write(*,*) '  2 - for arclength control (with bifurcation
analysis)'
        write(*,*) '  3 - for linearized bifurcation analysis'
        write(*,*) '      '
        read(*,*) isol
        if(isol.ne.3) then
          write(*,*) '      '
          write(*,*) '  Need to locate bifurcation point ??'
          write(*,*) '  if yes, enter 2; if no, enter 1:'
          write(*,*) '      '
          read(*,*) ip
          write(*,*) '   Enter incremental load factor :'
          read(*,*) alinc
          write(*,*) '   Enter the maximum load factor :'
          read(*,*) toltld
          write(*,*) '   Enter the node number '
          write(*,*) 'at which displacent will be printed:'
          read(*,*) nodepr
        end if
        if(isol.eq.3) then
          write(*,*) ' Choose the method to be used:'
```

```
          write(*,*) '      '
          write(*,*) '  1 - for classical solution: neglect rotations '
          write(*,*) '  2 - for complete linearized solution'
          write(*,*) '      '
          read(*,*) ip2
        end if
C
C      CN,CM = MATERIAL CONSTANTS
C

        OPEN (UNIT=5,FILE='archcode.in',STATUS='OLD')
        OPEN (UNIT=6,FILE='archcode.out',STATUS='UNKNOWN')
c       OPEN (UNIT=7,FILE='mohan.out',STATUS='UNKNOWN')
        OPEN (UNIT=8,FILE='archu',STATUS='unknown')
        OPEN (UNIT=9,FILE='archv',STATUS='unknown')
        OPEN (UNIT=11,FILE='archw',STATUS='unknown')
        OPEN (UNIT=13,FILE='archrx',STATUS='unknown')
        OPEN (UNIT=14,FILE='archry',STATUS='unknown')
        OPEN (UNIT=15,FILE='archrz',STATUS='unknown')
        open(10,file = 'bifurcation.out', status = 'unknown')
        open(12, file = 'archdet', status = 'unknown')
        OPEN (UNIT=16,FILE='arch1',STATUS='UNKNOWN')
        OPEN (UNIT=18,FILE='archshape',STATUS='unknown')
        OPEN (UNIT=19,FILE='archdisp',STATUS='unknown')

C
C  INPUT THE
C           TITLE:                    UP TO 80 LETTERS AND NUMBERS
C           TOTAL NUMNER OF NODES:    NTNODE
C           TOTAL NUMBER OF ELEMENT:  NELM
C
        CALL INPUT1(NTNODE,NELM,nhing1,nhing3)
C
C  INPUT ANALYSIS DATA AND PROCESS BOUNDARY CONDITIONS
C

        CALL INPUT2(NTNODE,NELM,nhing1,nhing3,
     &              XYZ,X0,MELM,mhing1,mhing3,
     &              hing1,
     &              NODBC,PTLOAD,DISTGG,DISTNN,DISTPP,
     &              CNCM,MAX)


        MAXEQN=MAX-1
        if(isol.ne.3) then
          write(8,*)'TITLE = "Displacent U at node = ',nodepr,'"'
          write(8,*)'VARIABLES = "U","load factor `l"'
          write(8,*)'ZONE T=" U ", I=  '
          write(9,*)'TITLE = "Displacent V at node = ',nodepr,'"'
          write(9,*)'VARIABLES = "V","load factor `l"'
          write(9,*)'ZONE T=" V ", I=  '
          write(11,*)'TITLE = "Displacent W at node = ',nodepr,'"'
          write(11,*)'VARIABLES = "W","load factor `l"'
          write(11,*)'ZONE T=" W ", I=  '
          write(13,*)'TITLE = "Rotation around X at node = ',nodepr,'"'
          write(13,*)'VARIABLES = "R_x","load factor `l"'
          write(13,*)'ZONE T=" Rx ", I=  '
```

```
          write(14,*)'TITLE = "Rotation around Y at node = ',nodepr,'"'
          write(14,*)'VARIABLES = "R_y","load factor `l"'
          write(14,*)'ZONE T=" Ry ", I=  '
          write(15,*)'TITLE = "Rotation around Z at node = ',nodepr,'"'
          write(15,*)'VARIABLES = "R_z","load factor `l"'
          write(15,*)'ZONE T=" Rz ", I=  '
          write(12,*)'TITLE = "Relative det(Kt) vs load factor"'
          write(12,*)'VARIABLES = "load factor `l","Relative det(Kt)"'
          write(12,*)'ZONE T="det ", I=  '
        end if

        write(18,*)'TITLE = "Structural shapes on X-Y plane"'
        write(18,*)'VARIABLES = "X","Y","Z"'
C       write(18,*)'ZONE T="shape at load =',alambd,'", I=',NTNODE
C
C
C  PREPARE TO DO ANLAYSIS
C
        NWK = max*max
C
C    INITIALIZE xnodel,ttrg,omegag
C

        CALL initl(NELM,x0,xnodel,ttrg,omegag)

        CALL initl(NELM,x0,xnodel,t6rg,om6gag)

C*****************GEOMETRICALLY NONLIN. STATIC LOAD***************
C   ISOL=1     LOAD CONTROL
C   ISOL=2     arclength control (RIKS-WEMPNER METHOD)
c   ISOL=3     linearized bifurcation analysis (and  branching)
c2345678901234567890123456789012345678901234567890123456789012345678901
2
C
C    dq0 --- initial unbalanced load at current load level
C    da0 --- displ due to initial unbalanced load dq0
C    dq  --- unbalanced load at current iteration loop and current load
level
C    da  --- displ due to unbalanced load dq
C
C    NLD --- load level number
C   TLOD --- total load factor at NLD
C  ALINC --- incremental load factor
C
C


        maxnld = 2000      ! maximum number of load steps: nld<<maxnld
                           !              nld<<maxnld

        maxldn = 10         ! maximum number of load reducing steps:
                           !           ldown << maxldn

        maxitr = 70        !  maximum number of iterations inside each
load step:
                           !             itr << maxitr
```

```
c       toltld = 50.      ! tolerance for maximum loading factor

        tolda= 0.00000001      ! tolerance for dispacement increment

        toldq= 0.00000001      ! tolerance for load residual
        if(isol.le.2.and.ip.eq.2) then
          tolda= 0.0000001       ! tolerance for dispacement increment
          toldq= 0.0000001      ! tolerance for load residual
        end if
        toldadq = 0.000005    ! tolerance for dispacement
increment&load residual

        tolrdet = 0.000001   ! tolerance for relative determinant of
tangent
                            !     stiffness matrix to locate bifurcation
point

        cadd = 0.000001       ! scalar to add eigenvecotor to
displacment vector

        Npr=0
        NLD=0
        alamb0=0.0D0        ! total load factor
        alambd=0.0D0

        det0=1.
        rdet=1.

c        ISOL=3

C
C    INITIALIZE xnodel,ttrg,omegag
C

        CALL initl(NELM,x0,xnodel,ttrg,omegag)

        CALL initl(NELM,x0,xnodel,t6rg,om6gag)
        if(isol.ne.3) then
          CALL PRDISP(NTNODE,NODBC,atotl,nodepr,alambd)
          write(12,*) '0.00000             1.0000000'

        end if

        CALL CLVEC(dq0,MAX)
        CALL CLVEC(da0,MAX)
        CALL CLVEC(dq,MAX)
        CALL CLVEC(da,MAX)
        CALL CLVEC(atotl,MAX)
        CALL CLVEC(atotl0,MAX)
        call tempsv(maxeqn,nelm,
     &              xnodel,ttrg,omegag,da,dq,
     &              sxnode,sttrg,somegg,sda,sdq)

        call tempsv(maxeqn,nelm,
     &              xnodel,t6rg,om6gag,da,dq,
     &              sxnode,st6rg,som6gg,sda,sdq)
```

```
   alamb0 = alambd

   if(isol.eq.1) then
     alinc0 = alinc
     call LODSTP(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,aext0,aint,
&                    wr,wi,z,loc,fv1,iv1,
&                    maxnld,maxldn,maxitr,
&                    NLD,ldown,ilast,
&                    toltld,tolda,toldq,toldadq,
&                    alamb0,alinc0,det0,rdet,nodepr,ip)

   end if

   if(isol.eq.2) then

     alinc0 = alinc
     call ARCSTP(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,aext0,aint,
&                    wr,wi,z,loc,fv1,iv1,
&                    maxnld,maxldn,maxitr,
&                    NLD,ldown,ilast,
&                    toltld,tolda,toldq,toldadq,cadd,
&                    alamb0,alinc0,det0,rdet,nodepr,ip)

   end if


   if(isol.eq.3) then

     call BIFURL(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
```

```
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,
&                    wr,wi,z,loc,fv1,iv1,
&                    maxnld,maxldn,maxitr,
&                    NLD,ldown,niter,
&                    toltld,tolda,toldq,toldadq,
&                    alambd,alamb0,det0,rdet,ip2)

   end if

   STOP
   END
```

```
C  analy1.f:
C
        SUBROUTINE INPUT1(NTNODE,NELM,nhing1,nhing3)
        INTEGER TITLE(20)

        READ(5,10) TITLE
        WRITE(6,20) TITLE

        READ(5,*) NTNODE,NELM
        WRITE(6,30) NTNODE,NELM
        read(5,*) nhing1
        WRITE(6,*) 'The number of 1 dof hinges: nhing1=',nhing1
c       read(5,*) nhing3
c       WRITE(6,50) nhing3
10      FORMAT(20A4)
20      FORMAT(1X,20A4//)
30      FORMAT(1X,'THE TOTAL NUMBER OF    NODES: NTNODE = ',I3/
     &         1X,'THE TOTAL NUMBER OF ELEMENTS:  NELM  = ',I3//)
35     FORMAT(1X,'THE TOTAL NUMBER OF  FREE 1DOF HINGES: NHINGE =
',I3/)
50     FORMAT(1X,'THE TOTAL NUMBER OF  FREE 3DOF HINGES: NHINGE =
',I3/)
        RETURN
        END

        SUBROUTINE INPUT2(NTNODE,NELM,nhing1,nhing3,
     &              XYZ,X0,MELM,mhing1,mhing3,
     &              hing1,
     &              NODBC,PTLOAD,DISTGG,DISTNN,DISTPP,
     &              CNCM,MAX)

        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION
XYZ(3,NTNODE),X0(3,4,NELM),MELM(NELM,4),NODBC(NTNODE,6)
        DIMENSION mhing1(2,*),mhing3(2,*)
        DIMENSION hing1(5,*)
        DIMENSION PTLOAD(*),DISTGG(3,4,NELM),DISTNN(3,4,NELM)
        DIMENSION DISTPP(3,4,NELM)
        DIMENSION CNCM(6,4,NELM)

C
C     NODAL COORDINATES: XYZ
C
        READ(5,*) ((XYZ(IJK,I),IJK=1,3),I=1,NTNODE)
        WRITE(6,*)'                                                  '
        WRITE(6,*)'                NODAL COODINATES                  '
        WRITE(6,*)'==============================================='
        WRITE(6,*)' NO. OF NODE        X            Y            Z   '
        WRITE(6,*)'                                                  '

        DO I=1,NTNODE
          WRITE(6,10) I,(XYZ(IJK,I),IJK=1,3)
10        FORMAT(5X,I3,6X,3D13.5)
        END DO
C
C     ELEMENT CONNECTION MATRIX: MELM
C
```

```
        READ(5,*) ((MELM(IDELEM,I),I=1,4),IDELEM=1,NELM)
        WRITE(6,*)'                                                    '
        WRITE(6,*)'                ELEMENT CONNECTION                  '
        WRITE(6,*)'================================================== '
        WRITE(6,*)'NO. OF ELEMENT   NODE 1   NODE 2   NODE 3   NODE 4'
        WRITE(6,*)'                                                    '
        DO IDELEM=1,NELM
          WRITE(6,20) IDELEM,(MELM(IDELEM,I),I=1,4)
20        FORMAT(7X,I3,10X,I3,6X,I3,6X,I3,6X,I3)
        END DO
C
C     1DOF HINGE CONNECTION MATRIX: MHING1
C
        if(nhing1.ne.0) then
        READ(5,*) ((MHING1(I,IHINGE),I=1,2),IHINGE=1,NHING1)
        WRITE(6,*)'                                                    '
        WRITE(6,*)'              1DOF HINGE CONNECTION
'
        WRITE(6,*)'================================================== '
        WRITE(6,*)'NO. OF ELEMENT   P NODE   Q NODE'
        WRITE(6,*)'                                                    '
        DO IHINGE=1,NHING1
          WRITE(6,21) IHINGE,(MHING1(I,IHINGE),I=1,2)
21        FORMAT(7X,I3,10X,I3,6X,I3)
        END DO
C
C     1DOF HINGE direction MATRIX: HING1
C

        read(5,*) ((HING1(I,IHINGE),I=1,4),IHINGE=1,NHING1)
        WRITE(6,*)'                                                    '
        WRITE(6,*)'               1DOF HINGE Direction                  '
        WRITE(6,*)'================================================== '
        WRITE(6,*)'NO. OF 1 dof hinge  x-dir   y-dir    z-dir     k  '
        WRITE(6,*)'                                                    '
        DO IHINGE=1,NHING1
          WRITE(6,22) IHINGE,(HING1(I,IHINGE),I=1,4)
22        FORMAT(7X,I3,10X,E15.7,6X,E15.7,6X,E15.7,6X,E15.7)
        END DO
        DO IHINGE=1,NHING1
          HING1(5,IHINGE)=0.
        END DO

        end if

C
C     3DOF HINGE CONNECTION MATRIX: MHING3
C

c       READ(5,*) ((MHING3(I,IHINGE),I=1,2),IHINGE=1,NHING3)
c        WRITE(6,*)'                                                    '
c        WRITE(6,*)'                3DOF HINGE CONNECTION               '
c        WRITE(6,*)'================================================== '
c        WRITE(6,*)'NO. OF ELEMENTT  P NODE   Q NODE'
c        WRITE(6,*)'                                                    '
c
```

```
c       DO IHINGE=1,NHING3
c         WRITE(6,25) IHINGE,(MHING3(I,IHINGE),I=1,2)
25        FORMAT(7X,I3,10X,I3,6X,I3)
c       END DO
C
C  TRANSFER XYZ TO X0
C
C

        DO IDELEM = 1, NELM
          DO NOD = 1,4

            DO IDOF = 1,3
              X0(IDOF,NOD,IDELEM) = XYZ(IDOF,MELM(IDELEM,NOD))
            END DO

          END DO

        END DO


C
C    MATERTIAL CONSTANT MATRICES: CN,CM
C
        WRITE(6,*)'                                                  '
        WRITE(6,*)'            CROSS-SECTIONAL PROPERTIES            '
        WRITE(6,*)'================================================ '
        WRITE(6,*)'                                                '
        DO IDELEM = 1, NELM
          WRITE(6,*) 'ELEMENT NO. ',IDELEM
          WRITE(6,*)'                                              '
          DO NOD = 1, 4
            READ(5,*) (CNCM(IDOF,NOD,IDELEM), IDOF=1,6)
            WRITE(6,*) 'AT NODE ',NOD
            WRITE(6,*)'                                            '
            WRITE(6,*)'         EAt           GAn          GAb     '
            WRITE(6,*)'                                            '
            WRITE(6,27)  (CNCM(IDOF,NOD,IDELEM), IDOF=1,3)
            WRITE(6,*)'                                            '
            WRITE(6,*)'             GJt          EIn          EIb '
            WRITE(6,*)'                                            '
            WRITE(6,27)  (CNCM(IDOF,NOD,IDELEM), IDOF=4,6)
            WRITE(6,*)'                                            '
          END DO
        END DO

27      format(4x,3D13.5)

C
C  DISPLACEMENT B.C.
C

        DO I=1,NTNODE
          DO IDOF=1,6
            NODBC(I,IDOF)=1
          END DO
```

```
        END DO
        READ(5,*) NPVBC

        WRITE(6,*) '                                                    '
        WRITE(6,*) 'THE TOTAL NUMBER OF PRESCRIBED ZERO DISPLACEMENT:'
        WRITE(6,*) '=============================================== '
        WRITE(6,*) '                                                   '
        WRITE(6,30) NPVBC
30      FORMAT(1X,5X, 'NPVBC = ',I3)
        DO I=1,NPVBC
          READ(5,*) INOD,IDOF
          NODBC(INOD,IDOF)=0
        END DO
        WRITE(6,*) '                                                   '

        CALL DISPBC(NTNODE,NODBC,MAX)
        WRITE(6,*) '                                                   '
C
C
C     LOADS
C                           POINT LOADS: PTLOAD
C                      LINE WEIGHT DENSIT: DISTGG
C           LINE DISTRIBUTED LOAD DENSITY: DISTNN
C
C


C
C     POINT LOAD
C

        DO I=1,MAX-1
          PTLOAD(I)=0.d0
        END DO

        READ(5,*) NSVBC
        WRITE(6,*) '                                                   '
        WRITE(6,*) 'THE TOTAL NUMBER OF PRESCRIBED NONZERO POINT
LOADS:'
        WRITE(6,*)
'================================================'
        WRITE(6,*) '                                                   '
        WRITE(6,40) NSVBC
40      FORMAT(1X,5X, 'NSVBC = ',I3)
        write(6,*) '                                                 '
        write(6,*) ' THE EFFECTIVE NONZERO POINT LOADS:              '
        WRITE(6,*)
'================================================'
        WRITE(6,*) '                                                   '
        write(6,*) ' No. of Node     No. of Component      Value     '
        if(NSVBC.ne.0) then
         DO I=1,NSVBC
          READ(5,*) INOD,IDOF,ALOD
          NLDOF=NODBC(INOD,IDOF)
          if(NLDOF.NE.0) THEN
            WRITE(6,50) INOD,IDOF,ALOD
            PTLOAD(NLDOF)=ALOD
          END IF
```

```
         END DO
        end if
50      FORMAT(5X,I3,16X,I3,6X,D15.6)
        WRITE(6,*) '                                                    '
        WRITE(6,*) '  Note: No. of Component =  1:  Px           '
        WRITE(6,*) '                            =  2:  Py           '
        WRITE(6,*) '                            =  3:  Pz           '
        WRITE(6,*) '                            =  4:  Mx           '
        WRITE(6,*) '                            =  5:  My           '
        WRITE(6,*) '                            =  6:  Mz           '
        WRITE(6,*) '        Otherwise, wrong data for point loads !'
        WRITE(6,*) '                                                    '
C
C  WEIGHT DENSITY
C


        DO IDELEM = 1, NELM
          DO NOD = 1,4
            DO IDOF = 1,3
              DISTGG(IDOF,NOD,IDELEM) = 0.D0
            END DO
          END DO
        END DO
        READ(5,*) NSVBCG
        WRITE(6,*) '                                                    '
        WRITE(6,*) 'THE TOTAL NUMBER OF PRESCRIBED NONZERO WT
ELEMENTS:'
        WRITE(6,*)
'================================================='
        WRITE(6,*) '                                                    '
        WRITE(6,45) NSVBCG

45      FORMAT(1X,5X, 'NSVBCG = ',I3)
        if(NSVBCG.ne.0) then
        write(6,*) '                                                '
        write(6,*) ' THE WEIGHT DENSITY:           '
        WRITE(6,*)
'================================================='
        WRITE(6,*) '                                                    '
        write(6,*) ' No. of EL-NODE    X-COMP     Y-COMP     Z-COMP '

        DO I=1,NSVBCG
          READ(5,*) IDELEM
          DO NOD = 1,4
            READ(5,*) (DISTGG(IDOF,NOD,IDELEM),IDOF = 1,3)
            WRITE(6,55) IDELEM,NOD,(DISTGG(IDOF,NOD,IDELEM),IDOF = 1,3)
          END DO
        END DO
55      FORMAT(3X,I4,3X,I4,3X,3D11.3)
        end if

C
C  DISTRIBUTED LOAD DENSITY
C
```

```
        DO IDELEM = 1, NELM
          DO NOD = 1,4
            DO IDOF = 1,3
              DISTNN(IDOF,NOD,IDELEM) = 0.D0
            END DO
          END DO
        END DO
        READ(5,*) NSVBCN
        WRITE(6,*) '                                                    '
        WRITE(6,*) 'THE TOTAL NUMBER OF NONZERO DISTRIB LOAD ELEMENTS:'
        WRITE(6,*)
'================================================='
        WRITE(6,*) '                                                    '
        WRITE(6,60) NSVBCN
60      FORMAT(1X,5X, 'NSVBCN = ',I3)
        write(6,*) '                                                '
        write(6,*) ' THE LOAD DENSITY:           '
        WRITE(6,*)
'================================================='
        WRITE(6,*) '                                                    '
        write(6,*) ' No. of EL-NODE    X-COMP     Y-COMP     Z-COMP '

        if(NSVBCN.ne.0) then
         DO I=1,NSVBCN
          READ(5,*) IDELEM
          DO NOD = 1,4
            READ(5,*) (DISTNN(IDOF,NOD,IDELEM),IDOF = 1,3)
            WRITE(6,55) IDELEM,NOD,(DISTNN(IDOF,NOD,IDELEM),IDOF = 1,3)
          END DO
         END DO
        end if

C
C  Hydrostatic load DENSITY
C
        DO IDELEM = 1, NELM
          DO NOD = 1,4
            DO IDOF = 1,3
              DISTPP(IDOF,NOD,IDELEM) = 0.D0
            END DO
          END DO
        END DO
        READ(5,*) NSVBCP
        WRITE(6,*) '                                                    '
        WRITE(6,*) 'THE TOTAL NUMBER OF NONZERO HYDRO LOAD ELEMENTS:'
        WRITE(6,*)
'================================================='
        WRITE(6,*) '                                                    '
        WRITE(6,70) NSVBCP
70      FORMAT(1X,5X, 'NSVBCP = ',I3)
        write(6,*) '                                                '
        write(6,*) ' THE LOAD DENSITY:           '
        WRITE(6,*)
'================================================='
        WRITE(6,*) '                                                    '
        write(6,*) ' No. of EL-NODE    t01-COMP   t02-COMP   t03-COMP '
```

```
      if(NSVBCP.ne.0) then
       DO I=1,NSVBCP
        READ(5,*) IDELEM
        DO NOD = 1,4
          READ(5,*) (DISTPP(IDOF,NOD,IDELEM),IDOF = 1,3)
          WRITE(6,55) IDELEM,NOD,(DISTPP(IDOF,NOD,IDELEM),IDOF = 1,3)
        END DO
       END DO
      end if


      RETURN
      END

C
C
C
C
C


      SUBROUTINE DISPBC(NTNODE,NODBC,MAX)
      DIMENSION NODBC(NTNODE,6)

      WRITE(6,10)

      NDOF=0
      DO II=1, NTNODE
        DO JJ=1,6
          IF(NODBC(II,JJ).NE.0) THEN
            NDOF=NDOF+1
            NODBC(II,JJ)=NDOF
          END IF
        END DO
        WRITE(6,20) II,(NODBC(II,JJ),JJ=1,6)
      END DO

      MAX=NDOF+1

10    FORMAT(/1x,'Node No.',21x,'The Sequential D.O.F.'//
     &         14x,'Ux',3x,'Uy',3x,'Uz',3x,
     &         'Rx',3x,'Ry',3x,'Rz'/)
20    FORMAT(3x,i5,3x,6i5)

      RETURN
      END


C
C
      SUBROUTINE NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION MELM(NELM,4),XNODEL(3,4,NELM),XYZN(3,NTNODE)

      DO NOD=1,NTNODE
        DO IDOF=1,3
          XYZN(IDOF,NOD)=0.D0
```

```
          END DO
        END DO

        DO IDELEM = 1, NELM
          DO NOD = 1,4

            DO IDOF = 1,3
              XYZN(IDOF,MELM(IDELEM,NOD))=XNODEL(IDOF,NOD,IDELEM)
            END DO

          END DO

        END DO


        write(18,*)'ZONE T="shape at load =',alambd,'", I=',NTNODE
        DO NOD=1,NTNODE
          WRITE(18,*) (XYZN(IDOF,NOD),IDOF=1,3)
        END DO

        RETURN
        END
C
        SUBROUTINE PRDISP(NTNODE,NODBC,atotl,node,alambd)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION NODBC(ntnode,6),atotl(*),atotlw(6,500)

        DO NOD=1,NTNODE
          DO IDOF=1,6
           atotlw(IDOF,NOD)=0.D0
          END DO
        END DO

        DO NOD=1,NTNODE

          DO IDOF=1,6
            ndof=NODBC(NOD,IDOF)
            if(ndof.ne.0) then
              atotlw(IDOF,NOD)=atotl(ndof)
            end if
          end do

        END DO

        write(19,*) '    '
        write(19,*) ' Nodal Displacement at load factor = ',alambd
        write(19,*) '-------------------------------------------'
        write(19,*) ' U       V        W       Rx      Ry       Rz'
        write(19,*) '    '
        do nod = 1,NTNODE
        write(19,*)  (atotlw(j,NOD), j=1,6)
        end do

        write(8,*)  atotlw(1,NODE),alambd

        write(9,*)  atotlw(2,NODE),alambd
```

```
write(11,*) atotlw(3,NODE),alambd

write(13,*) atotlw(4,NODE),alambd

write(14,*) atotlw(5,NODE),alambd

write(15,*) atotlw(6,NODE),alambd
RETURN
END

SUBROUTINE RESIDL(qext0,qint,maxeqn,alambd,dq)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION qext0(*),qint(*),dq(*)

do ii=1,maxeqn
  dq(ii) = alambd*qext0(ii) - qint(ii)
end do

return
END
SUBROUTINE INCREA(aext0,aint,maxeqn,alambd,da)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION aext0(*),aint(*),da(*)

do ii=1,maxeqn
  da(ii) = alambd*aext0(ii) - aint(ii)
end do

return
END
```

```
C analy2.f:
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c
c        This subroutine presents cubic lagrangian basis function
values
c
c  and their derivatives (wrt t) for a given value of the parameter t
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        subroutine Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
     &     ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)
        implicit real*8 (a-h,o-z)

        if(idv.eq.0) then
c Compute the Lagrangian basis functions

        ala1 = (2.d0 - 11.d0*t + 18.d0*t**2 - 9.d0*t**3)/2.d0
        ala2 = 9.d0*t*(2.d0 - 5.d0*t + 3.d0*t**2)/2.d0
        ala3 = 9.d0*t*(-1.d0 + 4.d0*t - 3.d0*t**2)/2.d0
        ala4 = t*(2.d0 - 9.d0*t + 9.d0*t**2)/2.d0
        end if

        if(idv.eq.1) then
c Compute the 1st order derivatives of the Lagrangian  basis functions

        dla1 = (-11.d0 + 36.d0*t - 27.d0*t**2)/2.d0
        dla2 = 9.d0 - 45.d0*t + 81.d0*t**2/2.d0
        dla3 = 9.d0*(-1.d0 + 8.d0*t - 9.d0*t**2)/2.d0
        dla4 = 1.d0 - 9.d0*t + 27.d0*t**2/2.d0

        end if

        if(idv.eq.2) then
c Compute the 2nd order derivatives of the  Lagrangian  basis functions

        ddla1 = 18.d0 - 27.d0*t
        ddla2 = -45.d0 + 81.d0*t
        ddla3 = 36.d0 - 81.d0*t
        ddla4 = -9.d0 + 27.d0*t

        end if

        if(idv.eq.3) then
c Compute the 3rd order derivatives of the  Lagrangian  basis functions

        dddla1 = - 27.d0
        dddla2 = 81.d0
        dddla3 = - 81.d0
        dddla4 = 27.d0

        end if
```

```
        return
        end

c
c
c234567890123456789012345678901234567890123456789012345678901234567890
2
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c
c    This subroutine is used to find the
c
c
c        unit tangential vector 't': tt0(ii,1)/
c        its first order derivative: d1tt0(ii,1)
c
c        unit   normal   vector 'n': tt0(ii,2)/
c        its first order derivative: d1tt0(ii,2) and
c
c        unit  binormal  vector 'b': tt0(ii,3)/
c        its first order derivative: d1tt0(ii,3)
c
c
c  of the curve piece 'i' at any point given by parameter t.
c
c    The curve piece is in the form of cubic lagrangian interpolation
c  function,
c
c  with x0(ii,jj,idelem ) as jj th (1 <= jj <=3) component of the node
ii
c  (1 <= ii <= 4) for the element idelem.
c
c    The tangential vector 't', normal vector 'n', and binormal vector
c  'b' will be stored in the first, second, and third column  of the 3
c  by 3 matrix tt0(i,j), respectively.
c
c
c    Besides, the determinant of Jacobian detj is an output.
c
c
c    Memo:
c
c    Note that the procedure may fail if the curve piece is straight or
c  the curvature at this point is zero. If we make an assumption that
c  the Frenet frame coincides with the principal axes of the curved
beam,
c  then the problem is treatable. But, then, is that easy? It is easy
if
c  the arch or frame is planar one. What about the spacial arch?
c
c    Planar arch or frame: the binormal vector is known beforehand
except
c  for the sign. If the curvature does not vanish, then the normal
vector
c  'n' can be determined. If the curvature vanishes, take any unit
vector
```

```
c  which is normal to the arch plane as binormal vector 'b'. Then the
c  normal vector 'n' can be determined without difficulty.
c
c    Spacial arch or frame: borrow that ideas for planar arch or frame,
c  we specify a binormal vector 'b' at the first knot on each B-spline
c  curve (which is composed of several Bezier curves). If the curvature
c  at this point is zero, then accept the prescribed binormal vector.
c  Otherwise, both the normal and binormal vectors can be calculated
from
c  the known curve. Therefore, the Frenet frame can be determined at
the
c  first point on the curve in any case. If the Frenet frame cannot be
c  determined from the curve itself at some point after the first
point,
c  then take the last Frenet frame at the last point as the current
frame.
c  This is reasonable as at the point of zero curvature, the derivative
c  of the Frenet frame and the torque are zero with continuous
conditions
c  for the frame.
c
c
c
c234567890123456789012345678901234567890123456789012345678901234567890
2
c
        subroutine Frenet(idelem,t,x0,tt0,d1tt0,detj,
     &               ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4)

        implicit real*8 (a-h,o-z)
        dimension x0(3,4,*), tt0(3,3), d1tt0(3,3),d2tt0(3,3)
        dimension xyz(3),d1xyz(3),d2xyz(3),d3xyz(3)

        tol=1.d-20         ! Tolerance for curvature checking.
c
c     Start to find the 0 to 2nd derivatives of the position vector
c          xyz(ii) at t on the curve piece idelem.
c
c  with x0(jj,ii,idelem ) as jj th (1 <= jj <=3) component of the node
ii
c  (1 <= ii <= 4) for the element idelem.

          do idv=0,3
            call Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
     &         ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)
          end do

          do jj=1,3
            xyz(jj)=ala1*x0(jj,1,idelem)+ala2*x0(jj,2,idelem)
     &             +ala3*x0(jj,3,idelem)+ala4*x0(jj,4,idelem)
          end do

          do jj=1,3
            d1xyz(jj)=dla1*x0(jj,1,idelem)+dla2*x0(jj,2,idelem)
     &               +dla3*x0(jj,3,idelem)+dla4*x0(jj,4,idelem)
          end do
```

```
            do jj=1,3
              d2xyz(jj)=ddla1*x0(jj,1,idelem)+ddla2*x0(jj,2,idelem)
     &               +ddla3*x0(jj,3,idelem)+ddla4*x0(jj,4,idelem)
            end do

            do jj=1,3
              d3xyz(jj)=dddla1*x0(jj,1,idelem)+dddla2*x0(jj,2,idelem)
     &               +dddla3*x0(jj,3,idelem)+dddla4*x0(jj,4,idelem)
            end do

            detj = dsqrt(d1xyz(1)*d1xyz(1)
     &              +d1xyz(2)*d1xyz(2)
     &              +d1xyz(3)*d1xyz(3))


            do jj=1,3
              d1xyz(jj)= d1xyz(jj)/detj
            end do

            do jj=1,3
              d2xyz(jj)= d2xyz(jj)/(detj*detj)
            end do

            do jj=1,3
              d3xyz(jj)= d3xyz(jj)/(detj*detj*detj)
            end do

c
c     The end of finding the 0 to 2nd derivatives of the position
c         vector  xyz(jj) at t on the curve piece idelem.
c

          do jj = 1,3
            tt0(jj,1) = d1xyz(jj)
          end do

          do jj = 1,3
            d1tt0(jj,1) = d2xyz(jj)
          end do

          do jj = 1,3
            d2tt0(jj,1) = d3xyz(jj)
          end do
C
C          icurve = 1   curved beam, all three frame axes are calculated.
c                       the curved beam can be 3D curved, not limited to
C                       on X-Y plane
C
C          icurve = 2   straight beam, 'b' is given as parallel to Z axis
C                       and assume that the beam is on X-Y plane
C
          icurve = 2

          if(icurve.eq.1) then  ! curved beam on X-Y plane
c
c     Start to find the unit binormal vector 'n'.
c
```

```
         adt01 = dsqrt(d1tt0(1,1)*d1tt0(1,1)
     &               +d1tt0(2,1)*d1tt0(2,1)
     &               +d1tt0(3,1)*d1tt0(3,1))

         dadt01 = (d1tt0(1,1)*d2tt0(1,1)
     &           + d1tt0(2,1)*d2tt0(2,1)
     &           + d1tt0(3,1)*d2tt0(3,1))/adt01



         do jj = 1,3
           tt0(jj,2) = d1tt0(jj,1)/adt01
         end do
c
c     End to find the unit binormal vector 'n'.
c
c     Start to find the unit binormal vector 'b'.
c
         tt0(1,3) = tt0(2,1)*tt0(3,2)-tt0(3,1)*tt0(2,2)
         tt0(2,3) = tt0(3,1)*tt0(1,2)-tt0(1,1)*tt0(3,2)
         tt0(3,3) = tt0(1,1)*tt0(2,2)-tt0(2,1)*tt0(1,2)

c
c     The end of finding the unit binormal vector 'b'.
c
         d1tt0(1,3) = d1tt0(2,1)*tt0(3,2)+tt0(2,1)*d1tt0(3,2)
     &               -d1tt0(3,1)*tt0(2,2)-tt0(3,1)*d1tt0(2,2)
         d1tt0(2,3) = d1tt0(3,1)*tt0(1,2)+tt0(3,1)*d1tt0(1,2)
     &               -d1tt0(1,1)*tt0(3,2)-tt0(1,1)*d1tt0(3,2)
         d1tt0(3,3) = d1tt0(1,1)*tt0(2,2)+tt0(1,1)*d1tt0(2,2)
     &               -d1tt0(2,1)*tt0(1,2)-tt0(2,1)*d1tt0(1,2)

         end if

         if(icurve.eq.2) then    ! straight beam on X-Y plane

c     Start to find the unit binormal vector 'b'.
c



           tt0(1,3) = 0.d0
           tt0(2,3) = 0.d0
           tt0(3,3) = -1.d0



c
c     Start to find the unit binormal vector 'n'.
c
         tt0(1,2) = tt0(2,3)*tt0(3,1)-tt0(3,3)*tt0(2,1)
         tt0(2,2) = tt0(3,3)*tt0(1,1)-tt0(1,3)*tt0(3,1)
         tt0(3,2) = tt0(1,3)*tt0(2,1)-tt0(2,3)*tt0(1,1)

         end if

         return
         end
```

```
c
c The end of subroutine Frenet

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c  Update the total rotational matrix 'ttrg' and the spatial rotational
c  strain measure 'omegag' at gauss pts element by element
c
c  ttrg(ii,jj,ig,idelem) - input, old total rotational matrix;
c               output, new total rotational matrix
c
c  omega(jjii,jj,ig,idelem) - input the old rotational strain measure;
c               output the new rotational strain measure
c
c  dwnod(ii,nod) -  input, element nodal rotational vector increment
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c

         subroutine ttromg(NTNODE,NELM,MELM,NODBC,max,
     &                     x0,da,ttrg,omegag)
         implicit real*8 (a-h,o-z)

         dimension ttrg(3,3,6,*),omegag(3,6,*),ttr(3,3),omega(3)
         dimension gauspt(6),gauswt(6),x0(3,4,*),d1xyz(3)
         dimension da(*),dwnod(3,4)
         dimension MELM(NELM,4),NODBC(NTNODE,6)

         NGPT=3
         CALL GAUSS(GAUSPT,GAUSWT,NGPT)

         do idelem = 1,nelm

           do nod = 1,4      ! get dwnod from da
             NODE=MELM(IdELEM,NOD)
             do ii = 1,3
               NDOF=NODBC(NODE,3+ii)
               dwnod(ii,nod)=0.d0
               IF(NDOF.NE.0) THEN
                 dwnod(ii,nod) = da(NDOF)
               end if
             end do
           end do            ! get dwnod from da
c          write(6,*) 'da=',(da(ii),ii=1,max-1)
c          write(6,*)'dw=',((dwnod(ii,nod),ii=1,3),nod=1,4)
           DO  IG=1,NGPT     ! GAUSS INTEGRATION LOOP

             U=GAUSPT(IG)
             W=GAUSWT(IG)
             T=0.5D0*(U+1.D0)

             do idv=0,1                    ! compute detj
               call Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
     &         ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)
             end do
```

```
            do jj=1,3
              d1xyz(jj)=dla1*x0(jj,1,idelem)+dla2*x0(jj,2,idelem)
     &               +dla3*x0(jj,3,idelem)+dla4*x0(jj,4,idelem)
            end do

            detj = dsqrt(d1xyz(1)*d1xyz(1)
     &               +d1xyz(2)*d1xyz(2)
     &               +d1xyz(3)*d1xyz(3))  ! compute detj

            call getttr(idelem,ig,ttrg,ttr)
            call getomg(idelem,ig,omegag,omega)
c           write(6,*) 'detj=',detj
c           write(6,*) 'ttr=',ttr
c           write(6,*) 'omega=', omega
            call rtmatr(t,detj,dwnod,ttr,omega)

            do ii = 1,3
              do jj = 1,3
                ttrg(ii,jj,ig,idelem) = ttr(ii,jj)
              end do
              omegag(ii,ig,idelem) = omega(ii)
            end do

          end do
        end do

        return
        end

        subroutine t6romg(NTNODE,NELM,MELM,NODBC,max,
     &                    x0,da,ttrg,omegag)
        implicit real*8 (a-h,o-z)

        dimension ttrg(3,3,6,*),omegag(3,6,*),ttr(3,3),omega(3)
        dimension gauspt(6),gauswt(6),x0(3,4,*),d1xyz(3)
        dimension da(*),dwnod(3,4)
        dimension MELM(NELM,4),NODBC(NTNODE,6)

        NGPT=3
        CALL GAUSS(GAUSPT,GAUSWT,NGPT)

        do idelem = 1,nelm

          do nod = 1,4      ! get dwnod from da
            NODE=MELM(IdELEM,NOD)
            do ii = 1,3
              NDOF=NODBC(NODE,3+ii)
              dwnod(ii,nod)=0.d0
              IF(NDOF.NE.0) THEN
                dwnod(ii,nod) = da(NDOF)
              end if
            end do
          end do            ! get dwnod from da
c         write(6,*) 'da=',(da(ii),ii=1,max-1)
c         write(6,*)'dw=',((dwnod(ii,nod),ii=1,3),nod=1,4)
          DO  IG=1,NGPT     ! GAUSS INTEGRATION LOOP
```

```
              U=GAUSPT(IG)
              W=GAUSWT(IG)
              T=0.5D0*(U+1.D0)

              do idv=0,1                      ! compute detj
                call Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
     &          ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)
              end do

              do jj=1,3
                d1xyz(jj)=dla1*x0(jj,1,idelem)+dla2*x0(jj,2,idelem)
     &                 +dla3*x0(jj,3,idelem)+dla4*x0(jj,4,idelem)
              end do

              detj = dsqrt(d1xyz(1)*d1xyz(1)
     &                +d1xyz(2)*d1xyz(2)
     &                +d1xyz(3)*d1xyz(3))  ! compute detj

              call getttr(idelem,ig,ttrg,ttr)
              call getomg(idelem,ig,omegag,omega)
c             write(6,*) 'detj=',detj
c             write(6,*) 'ttr=',ttr
c             write(6,*) 'omega=', omega
              call rtmatr(t,detj,dwnod,ttr,omega)

              do ii = 1,3
                do jj = 1,3
                  ttrg(ii,jj,ig,idelem) = ttr(ii,jj)
                end do
                omegag(ii,ig,idelem) = omega(ii)
              end do

            end do
          end do

          return
          end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c  Update the total rotational matrix 'ttr' and the spatial rotational
c  strain measure 'omega' at point t for an element
c
c  ttr(ii,jj) - input, old total rotational matrix;
c               output, new total rotational matrix
c
c  omega(jj) - input the old rotational strain measure;
c              output the new rotational strain measure
c
c  dwnod(ii,nod) -  input, element nodal rotational vector increment
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c

          subroutine rtmatr(t,detj,dwnod,ttr,omega)
```

```
            implicit real*8 (a-h,o-z)
            dimension dwnod(3,4),ttr(3,3),omega(3)
            dimension tmatdw(3,3),dtmtdw(3,3)
            dimension wk0(3,3),wk1(3,3),wk2(3,3),wk3(3,3)

            call  rtmatd(t,detj,dwnod,tmatdw,dtmtdw)

            do ii = 1,3
              do jj = 1,3
                wk1(ii,jj) = ttr(ii,jj)
              end do
            end do

            call matmat(tmatdw,wk1,ttr)

            do ii = 1,3
              do jj = 1,3
                wk1(ii,jj) = tmatdw(jj,ii)
              end do
            end do


            call vskew(omega,wk0)

            call matmat(dtmtdw,wk1,wk2)

            call matmat(tmatdw,wk0,wk3)

            call matmat(wk3,wk1,wk0)

            do ii = 1,3
              do jj = 1,3
                wk3(ii,jj) = wk2(ii,jj) + wk0(ii,jj)
              end do
            end do

            call skewv(wk3,omega)

          return
          end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Rotational matrix 'tmatdw' and its derivative 'dtmtdw' with respect
c  to arc length paramterized by rotational vector-dw.
c
c  dwnod(ii,nod) - input, nodal rotational vector
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

          subroutine rtmatd(t,detj,dwnod,tmatdw,dtmtdw)

            implicit real*8 (a-h,o-z)
            dimension dwnod(3,4),tmatdw(3,3),dtmtdw(3,3)
            dimension dwvec(3),ddwvec(3)
            dimension sk(3,3),dsk(3,3)
            dimension wk1(3,3),wk2(3,3),wk3(3,3)
```

```
      idv = 0
       call Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
&         ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)
      idv = 1
       call Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
&         ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)

    do jj=1,3

       dwvec(jj) = ala1*dwnod(jj,1)
&                 +ala2*dwnod(jj,2)
&                 +ala3*dwnod(jj,3)
&                 +ala4*dwnod(jj,4)
       ddwvec(jj) = (dla1*dwnod(jj,1)
&                   +dla2*dwnod(jj,2)
&                   +dla3*dwnod(jj,3)
&                   +dla4*dwnod(jj,4))/detj


    end do

    dw = dsqrt(dwvec(1)*dwvec(1)
&              +dwvec(2)*dwvec(2)
&              +dwvec(3)*dwvec(3))


    ddw = (dwvec(1)*ddwvec(1)
&         +dwvec(2)*ddwvec(2)
&         +dwvec(3)*ddwvec(3))/dw


    do ii = 1,3
      do jj = 1,3
        tmatdw(ii,jj) = 0.d0
        dtmtdw(ii,jj) = 0.d0
      end do
      tmatdw(ii,ii) = 1.d0
    end do
    if(dw.lt.1.d-10) then

      return

    end if
    call vskew(dwvec,sk)
    call vskew(ddwvec,dsk)
    call matmat(sk,sk,wk1)
    call matmat(dsk,sk,wk2)
    call matmat(sk,dsk,wk3)

    r1= dsin(dw)/dw
    r2= (1.d0-dcos(dw))/dw/dw
    dr1 = (dcos(dw) - r1)/dw*ddw
    dr2 = (r1-2.d0*r2)/dw*ddw


    do ii = 1,3
      do jj = 1,3
```

```
              tmatdw(ii,jj)=tmatdw(ii,jj)+r1*sk(ii,jj)+r2*wk1(ii,jj)
            end do
          end do

          do ii = 1,3
            do jj = 1,3
              dtmtdw(ii,jj)=dtmtdw(ii,jj)+dr1*sk(ii,jj)+r1*dsk(ii,jj)
     &                     +dr2*wk1(ii,jj)+r2*(wk2(ii,jj)+wk3(ii,jj))
            end do
          end do

          return
          end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
c
c  Update the overal orthogonal matrix 'tt'.
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc

        subroutine otmat(ttr,tt0,tt)
          implicit real*8 (a-h,o-z)
          dimension ttr(3,3),tt0(3,3),tt(3,3)

          call matmat(ttr,tt0,tt)

        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
c
c  Update the global nodal displacement vectors "atotl".
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
        subroutine upatot(ntnode,NODBC,atotl,da)

        IMPLICIT REAL*8 (A-H,O-Z)
        dimension atotl(*), dwvec(3), theta(3), dtheta(3)
        dimension NODBC(ntnode,6),da(*)

        do ii=1,ntnode
          do jj=1,3
            ndof=NODBC(II,JJ)
            if(ndof.ne.0) then
              atotl(ndof) = atotl(ndof) + da(ndof)
            end if
          end do
            dwvec(1)=0.
            dwvec(2)=0.
            dwvec(3)=0.
            theta(1)=0.
```

```
            theta(2)=0.
            theta(3)=0.
          do jj=1,3
            ndof=NODBC(II,JJ+3)
            if(ndof.ne.0) then
              theta(jj)=atotl(ndof)
              dwvec(jj)= da(ndof)
            end if
          end do
          call dwthet(dwvec,theta,dtheta)
          do jj=1,3
            ndof=NODBC(II,JJ+3)
            if(ndof.ne.0) then
              atotl(ndof)=atotl(ndof)+dtheta(jj)
            end if
          end do

        end do


        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
c
c  get dtheta from dwvec
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
        subroutine dwthet(dwvec,theta,dtheta)
        implicit real*8 (a-h,o-z)
        dimension dwvec(3),theta(3),dtheta(3),Tinv(3,3)

        call invTmt(theta,Tinv)

        do ii = 1,3
          dtheta(ii) = 0.
        end do

        do ii = 1,3
          dtheta(ii) = dtheta(ii)
     &               + Tinv(ii,1)*dwvec(1)
     &               + Tinv(ii,2)*dwvec(2)
     &               + Tinv(ii,3)*dwvec(3)
        end do

        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
c
c  get inverse of T(theta): Tinv
c
c
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
        subroutine invTmt(theta,Tinv)
        implicit real*8 (a-h,o-z)
        dimension theta(3), Tinv(3,3),skthet(3,3)

        atheta=0.
        do ii = 1,3
          atheta = atheta + theta(ii)*theta(ii)
        end do
        atheta = dsqrt(atheta)

        call vskew(theta,skthet)

        do ii = 1,3
          do jj = 1,3
            Tinv(ii,jj) = 0.
          end do
          Tinv(ii,ii) = 1.
        end do

        if(atheta.gt.1.d-100) then
          c1= 0.5*atheta/dtan(0.5*atheta)
          c2=(1.-c1)/(atheta*atheta)
          do ii = 1,3
            Tinv(ii,ii) = c1*Tinv(ii,ii)
          end do
          do ii = 1,3
            do jj = 1,3
              Tinv(ii,jj) = Tinv(ii,jj)
     &                      -0.5*skthet(ii,jj)+c2*theta(ii)*theta(jj)
            end do
          end do
        end if

        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   Get skew matrix 'skv' from vector 'v'.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


        subroutine vskew(v,skv)
          implicit real*8 (a-h,o-z)
          dimension v(3),skv(3,3)

          skv(1,1) = 0.d0
          skv(1,2) = -v(3)
          skv(1,3) = v(2)
          skv(2,1) = v(3)
          skv(2,2) = 0.d0
          skv(2,3) = -v(1)
          skv(3,1) = -v(2)
          skv(3,2) = v(1)
```

```
          skv(3,3) = 0.d0

        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   Get vector 'v' from skew matrix 'skv'
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine skewv(skv,v)
          implicit real*8 (a-h,o-z)
          dimension v(3),skv(3,3)

           v(1) = 0.5d0*(skv(3,2) - skv(2,3))
           v(2) = 0.5d0*(skv(1,3) - skv(3,1))
           v(3) = 0.5d0*(skv(2,1) - skv(1,2))

        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Get product 'c' of two 3 by 3 matrices 'a' an 'b':
c
c           c = a b
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


        subroutine matmat(a,b,c)
          implicit real*8 (a-h,o-z)
          dimension a(3,3),b(3,3),c(3,3)

          do i = 1,3
            do j = 1,3
              c(i,j) = 0.d0
              do k = 1,3
                c(i,j) = c(i,j) + a(i,k)*b(k,j)
              end do
            end do
          end do
        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc
c
c  Updated
c      position vector 'phiv' and its derivative 'dphiv' at 't'
c      for an element
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc
        subroutine posvec(t,detj,x,phiv,dphiv)
          implicit real*8 (a-h,o-z)
```

```
         dimension x(3,4),phiv(3),dphiv(3)
         idv =0
         call Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
     &     ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)

         idv =1
         call Lagran(t,ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4,
     &     ddla1,ddla2,ddla3,ddla4,dddla1,dddla2,dddla3,dddla4,idv)


         do ii = 1,3

           phiv(ii) = ala1*x(ii,1)
     &              +ala2*x(ii,2)
     &              +ala3*x(ii,3)
     &              +ala4*x(ii,4)

           dphiv(ii) = (dla1*x(ii,1)
     &               +dla2*x(ii,2)
     &               +dla3*x(ii,3)
     &               +dla4*x(ii,4))/detj

         end do

         return
         end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Calculate:
c             ccnm = tt.Cnm.Transpose(tt)
c
c               cnm = cn or cm
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         subroutine cnmmat(tt,cnm,ccnm)
         implicit real*8 (a-h,o-z)
         dimension tt(3,3),cnm(3,3),ccnm(3,3)
         ccnm(1,1) = cnm(1,1)*tt(1,1)*tt(1,1)
     &             + cnm(2,2)*tt(1,2)*tt(1,2)
     &             + cnm(3,3)*tt(1,3)*tt(1,3)
         ccnm(1,2) = cnm(1,1)*tt(1,1)*tt(2,1)
     &             + cnm(2,2)*tt(1,2)*tt(2,2)
     &             + cnm(3,3)*tt(1,3)*tt(2,3)
         ccnm(1,3) = cnm(1,1)*tt(1,1)*tt(3,1)
     &             + cnm(2,2)*tt(1,2)*tt(3,2)
     &             + cnm(3,3)*tt(1,3)*tt(3,3)
         ccnm(2,1) = cnm(1,1)*tt(1,1)*tt(2,1)
     &             + cnm(2,2)*tt(1,2)*tt(2,2)
     &             + cnm(3,3)*tt(1,3)*tt(2,3)
         ccnm(2,2) = cnm(1,1)*tt(2,1)*tt(2,1)
     &             + cnm(2,2)*tt(2,2)*tt(2,2)
     &             + cnm(3,3)*tt(2,3)*tt(2,3)
         ccnm(2,3) = cnm(1,1)*tt(2,1)*tt(3,1)
     &             + cnm(2,2)*tt(2,2)*tt(3,2)
     &             + cnm(3,3)*tt(2,3)*tt(3,3)
         ccnm(3,1) = cnm(1,1)*tt(1,1)*tt(3,1)
```

```
     &            + cnm(2,2)*tt(1,2)*tt(3,2)
     &            + cnm(3,3)*tt(1,3)*tt(3,3)
        ccnm(3,2) = cnm(1,1)*tt(2,1)*tt(3,1)
     &            + cnm(2,2)*tt(2,2)*tt(3,2)
     &            + cnm(3,3)*tt(2,3)*tt(3,3)
        ccnm(3,3) = cnm(1,1)*tt(3,1)*tt(3,1)
     &            + cnm(2,2)*tt(3,2)*tt(3,2)
     &            + cnm(3,3)*tt(3,3)*tt(3,3)
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Calculate translational strain measure in spacial form:
c     gamma = dphiv - ttr. tt0(*,1)
c
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine gammav(tt0,ttr,dphiv,gamma)
        implicit real*8 (a-h,o-z)
        dimension tt0(3,3),ttr(3,3),dphiv(3),gamma(3)
        do ii = 1,3
         gamma(ii) = 0.d0
         gamma(ii) = dphiv(ii) - (ttr(ii,1)*tt0(1,1)
     &                         + ttr(ii,2)*tt0(2,1)
     &                         + ttr(ii,3)*tt0(3,1))

        end do

        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       Calculate the  stress resultants in spacial form:
c
c
c         sn = (tt.cn.transform[tt]).gamma = ccn.gamma
c         sm = (tt.cm.transform[tt]).omega = ccm.omega
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sstrss(ccn,ccm,gamma,omega,sn,sm)
        implicit real*8 (a-h,o-z)
        dimension ccn(3,3),ccm(3,3),gamma(3),omega(3),sn(3),sm(3)
        do ii =1,3
          sn(ii) = ccn(ii,1)*gamma(1)
     &           + ccn(ii,2)*gamma(2)
     &           + ccn(ii,3)*gamma(3)
          sm(ii) = ccm(ii,1)*omega(1)
     &           + ccm(ii,2)*omega(2)
     &           + ccm(ii,3)*omega(3)
        end do
        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
```

```
c    Calculate the real stress resultants:
c
c       rn = transpose[tt].sn
c       rm = transpose[tt].sm
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine rstrss(tt,sn,sm,rn,rm)
        implicit real*8 (a-h,o-z)
        dimension tt(3,3),sn(3),sm(3),rn(3),rm(3)
        do ii = 1,3
          rn(ii) = tt(1,ii)*sn(1)+tt(2,ii)*sn(2)+tt(3,ii)*sn(3)
          rm(ii) = tt(1,ii)*sm(1)+tt(2,ii)*sm(2)+tt(3,ii)*sm(3)
        end do
        return
        end



ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c
c
c   Note: Degrees of freedom for an element
c
c    Incremental displacements          Internal loads
c
c                                     ( External loads :)
c                                    (change 'int' to 'ext')
c
c     DOF
c
c      1    dae(1) = dx(1,1)      qinte(1) = pinte(1,1)
c      2    dae(2) = dx(2,1)      qinte(2) = pinte(2,1)
c      3    dae(3) = dx(3,1)      qinte(3) = pinte(3,1)
c      4    dae(4) = dwnod(1,1)   qinte(4) = aminte(1,1)
c      5    dae(5) = dwnod(2,1)   qinte(5) = aminte(2,1)
c      6    dae(6) = dwnod(3,1)   qinte(6) = aminte(3,1)
c      7    dae(7) = dx(1,2)      qinte(7) = pinte(1,2)
c      8    dae(8) = dx(2,2)      qinte(8) = pinte(2,2)
c      9    dae(9) = dx(3,2)      qinte(9) = pinte(3,2)
c     10    dae(10) = dwnod(1,2)   qinte(10) = aminte(1,2)
c     11    dae(11) = dwnod(2,2)   qinte(11) = aminte(2,2)
c     12    dae(12) = dwnod(3,2)   qinte(12) = aminte(3,2)
c     13    dae(13) = dx(1,3)      qinte(13) = pinte(1,3)
c     14    dae(14) = dx(2,3)      qinte(14) = pinte(2,3)
c     15    dae(15) = dx(3,3)      qinte(15) = pinte(3,3)
c     16    dae(16) = dwnod(1,3)   qinte(16) = aminte(1,3)
c     17    dae(17) = dwnod(2,3)   qinte(17) = aminte(2,3)
c     18    dae(18) = dwnod(3,3)   qinte(18) = aminte(3,3)
c     19    dae(19) = dx(1,4)      qinte(19) = pinte(1,4)
c     20    dae(20) = dx(2,4)      qinte(20) = pinte(2,4)
c     21    dae(21) = dx(3,4)      qinte(21) = pinte(3,4)
c     22    dae(22) = dwnod(1,4)   qinte(22) = aminte(1,4)
c     23    dae(23) = dwnod(2,4)   qinte(23) = aminte(2,4)
c     24    dae(24) = dwnod(3,4)   qinte(24) = aminte(3,4)
c
```

```
c
c
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
2
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c        Element tangent Stiffness matrix:
c
c
c         stifel(ii,jj) -- (ii=1,2,...,24; jj=1,2,...,24)
c
c
c
        subroutine elstif(idelem,x0,xnodel,ttrg,omegag,cncm,
     &                    t6rg,om6gag,distrn,distrp,stifel)
        implicit real*8 (a-h,o-z)
        dimension stifel(24,24),stifem(24,24),stifeg(24,24)
        dimension stifen(24,24),stifep(24,24)
        dimension gauspt(6),gauswt(6)
        dimension x0(3,4,*),xnodel(3,4,*),x(3,4)
        dimension tt0(3,3),d1tt0(3,3)
        dimension ttrg(3,3,6,*),ttr(3,3),tt(3,3)
        dimension t6rg(3,3,6,*)
        dimension omegag(3,6,*),omega(3)
        dimension om6gag(3,6,*)
        dimension phiv(3),dphiv(3),dphisk(3,3),gamma(3)
        dimension cncm(6,4,*)
        dimension cn(3,3),ccn(3,3),cm(3,3),ccm(3,3)
        dimension sn(3),snsk(3,3),sm(3),smsk(3,3), rn(3),rm(3)
        dimension ccnphi(3,3),phiccn(3,3),phiphi(3,3)
        dimension phisn(3,3),snphi(3,3),shpn(4),dshpn(4)
        dimension pv(3),pskw(3,3),pv1(3),distrn(3,4),distrp(3,4)
        dimension anv(3),dv(3),wkn(3,3)

        dv(1)= 1.d0
        dv(1)= 0.d0
        dv(1)= 0.d0

        DO II=1,24
          DO JJ=1,24
            STIFel(II,JJ)=0.D0
            STIFem(II,JJ)=0.D0
            STIFeg(II,JJ)=0.D0
            STIFen(II,JJ)=0.D0
            STIFep(II,JJ)=0.D0
          END DO
        END DO

        NGPT=3
        CALL GAUSS(GAUSPT,GAUSWT,NGPT)

        DO  IG=1,NGPT     ! GAUSS INTEGRATION LOOP

          U=GAUSPT(IG)
          W=GAUSWT(IG)
```

```
            T=0.5D0*(U+1.D0)

            call Frenet(idelem,t,x0,tt0,d1tt0,detj,
     &                  ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4)

c           write(6,*) 'idelem=',idelem
c           write(6,*) 'tt0=', tt0

            detjh=.5D0*DETJ

            shpn(1) = ala1
            shpn(2) = ala2
            shpn(3) = ala3
            shpn(4) = ala4
            dshpn(1) = dla1/detj
            dshpn(2) = dla2/detj
            dshpn(3) = dla3/detj
            dshpn(4) = dla4/detj

            call getcnm(idelem,cncm,cm,cn,shpn)
            call getttr(idelem,ig,ttrg,ttr)

            call getomg(idelem,ig,omegag,omega)

            call otmat(ttr,tt0,tt)

            call getx(idelem,xnodel,x)


            call posvec(t,detj,x,phiv,dphiv)

            call gammav(tt0,ttr,dphiv,gamma)

            call cnmmat(tt,cn,ccn)
            call cnmmat(tt,cm,ccm)

            call sstrss(ccn,ccm,gamma,omega,sn,sm)

            call vskew(dphiv,dphisk)

            call matmat(ccn,dphisk,ccnphi)

            call matmat(dphisk,ccn,phiccn)

            call matmat(dphisk,ccnphi,phiphi)   ! ?????????

            call vskew(sn,snsk)
            call vskew(sm,smsk)

            call matmat(dphisk,snsk,phisn)
            call matmat(snsk,dphisk,snphi)

            call VECVEC(dphiv,dphiv,andphi,3)
            andphi = dsqrt(andphi)
            call VECVEC(dv,dphiv,ddphi,3)

            coe1 = ddphi/(andphi*andphi)
```

```
      coe2 = -1.d0/andphi


      do jj = 1,3
        anv(jj)= shpn(1)*distrn(jj,1)+
     &           shpn(2)*distrn(jj,2)+
     &           shpn(3)*distrn(jj,3)+
     &           shpn(4)*distrn(jj,4)
        pv(jj) = shpn(1)*distrp(jj,1)+
     &           shpn(2)*distrp(jj,2)+
     &           shpn(3)*distrp(jj,3)+
     &           shpn(4)*distrp(jj,4)
      end do


      do ii = 1,3
       do jj = 1,3
         wkn(ii,jj) =anv(ii)*(coe1*dphiv(jj) + coe2*dv(jj))
       end do
      end do

      do jj = 1,3
        pv1(jj) = tt(jj,1)*pv(1)+
     &            tt(jj,2)*pv(2)+
     &            tt(jj,3)*pv(3)
      end do

      call vskew(pv1, pskw)

      do ii = 1,4
        do jj = 1,4

          do i = 1,3
            do j = 1,3
              stifem(6*(ii-1)+i,6*(jj-1)+j) =
     &                  stifem(6*(ii-1)+i,6*(jj-1)+j)
     &                + w*dshpn(ii)*dshpn(jj)*ccn(i,j)*detjh
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifem(6*(ii-1)+i,6*(jj-1)+3+j) =
     &                  stifem(6*(ii-1)+i,6*(jj-1)+3+j)
     &                + w*dshpn(ii)*shpn(jj)*ccnphi(i,j)*detjh
            end do
          end do


          do i = 1,3
            do j = 1,3
              stifem(6*(ii-1)+3+i,6*(jj-1)+j) =
     &                  stifem(6*(ii-1)+3+i,6*(jj-1)+j)
     &                -w*shpn(ii)*dshpn(jj)*phiccn(i,j)*detjh
            end do
          end do
```

```
          do i = 1,3
            do j = 1,3
              stifem(6*(ii-1)+3+i,6*(jj-1)+3+j) =
&                       stifem(6*(ii-1)+3+i,6*(jj-1)+3+j)
&                      +w*(dshpn(ii)*dshpn(jj)*ccm(i,j)
&                      -shpn(ii)*shpn(jj)*phiphi(i,j))*detjh
            end do
          end do

        end do
      end do


      do ii = 1,4
        do jj = 1,4

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+i,6*(jj-1)+j) =
&                       stifeg(6*(ii-1)+i,6*(jj-1)+j)
&                      + 0.d0
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+i,6*(jj-1)+3+j) =
&                     stifeg(6*(ii-1)+i,6*(jj-1)+3+j)
&                - w*dshpn(ii)*shpn(jj)*snsk(i,j)*detjh
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+3+i,6*(jj-1)+j) =
&                     stifeg(6*(ii-1)+3+i,6*(jj-1)+j)
&                + w*shpn(ii)*dshpn(jj)*snsk(i,j)*detjh
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+3+i,6*(jj-1)+3+j) =
&                     stifeg(6*(ii-1)+3+i,6*(jj-1)+3+j)
&                + w*(shpn(ii)*shpn(jj)*phisn(i,j)
&                - dshpn(ii)*shpn(jj)*smsk(i,j))*detjh
            end do
          end do

        end do
      end do

      do ii = 1,4
        do jj = 1,4

          do i = 1,3
            do j = 1,3
```

```
            stifen(6*(ii-1)+i,6*(jj-1)+j) =
&                       stifen(6*(ii-1)+i,6*(jj-1)+j)
&                     + w*shpn(ii)*dshpn(jj)*wkn(i,j)*detjh
          end do
        end do

        do i = 1,3
          do j = 1,3
            stifen(6*(ii-1)+i,6*(jj-1)+3+j) =
&                   stifen(6*(ii-1)+i,6*(jj-1)+3+j)
&               + 0.d0
          end do
        end do

        do i = 1,3
          do j = 1,3
            stifen(6*(ii-1)+3+i,6*(jj-1)+j) =
&                   stifen(6*(ii-1)+3+i,6*(jj-1)+j)
&               + 0.d0
          end do
        end do

        do i = 1,3
          do j = 1,3
            stifen(6*(ii-1)+3+i,6*(jj-1)+3+j) =
&                   stifen(6*(ii-1)+3+i,6*(jj-1)+3+j)
&               + 0.d0
          end do
        end do

      end do
    end do

    do ii = 1,4
      do jj = 1,4

        do i = 1,3
          do j = 1,3
            stifep(6*(ii-1)+i,6*(jj-1)+j) =
&                       stifep(6*(ii-1)+i,6*(jj-1)+j)
&                     + 0.d0
          end do
        end do

        do i = 1,3
          do j = 1,3
            stifep(6*(ii-1)+i,6*(jj-1)+3+j) =
&                   stifep(6*(ii-1)+i,6*(jj-1)+3+j)
&               + w*shpn(ii)*shpn(jj)*pskw(i,j)*detjh
          end do
        end do

        do i = 1,3
          do j = 1,3
            stifep(6*(ii-1)+3+i,6*(jj-1)+j) =
&                   stifep(6*(ii-1)+3+i,6*(jj-1)+j)
&               + 0.d0
```

```
          end do
        end do

        do i = 1,3
          do j = 1,3
            stifep(6*(ii-1)+3+i,6*(jj-1)+3+j) =
&                 stifep(6*(ii-1)+3+i,6*(jj-1)+3+j)
&              + 0.d0
          end do
        end do

      end do
    end do

  end do

  do ii = 1,24
    do jj = 1,24
      stifel(ii,jj) = stifem(ii,jj)
&                   + stifeg(ii,jj)
&                   + stifen(ii,jj)
&                   + stifep(ii,jj)
    end do
  end do

  return
  end

  subroutine getcnm(idelem,cncm,cm,cn,shpn)
  implicit real*8 (a-h,o-z)
  dimension cncm(6,4,*),cn(3,3),cm(3,3),shpn(4)

  do ii=1,3
    do jj =1,3
      cn(ii,jj) = 0.
      cm(ii,jj) = 0.
    end do
  end do

  do ii = 1,3
    do nod = 1,4
      cn(ii,ii) = cn(ii,ii)+shpn(nod)*cncm(ii,nod,idelem)
      cm(ii,ii) = cm(ii,ii)+shpn(nod)*cncm(3+ii,nod,idelem)
    end do
  end do

  return
  end


  subroutine elstin(idelem,x0,xnodel,ttrg,omegag,cncm,
&                   t6rg,om6gag,distrn,distrp,stifel)
  implicit real*8 (a-h,o-z)
  dimension stifel(24,24),stifem(24,24),stifeg(24,24)
  dimension stifen(24,24),stifep(24,24)
  dimension gauspt(6),gauswt(6)
  dimension x0(3,4,*),xnodel(3,4,*),x(3,4)
```

```
      dimension tt0(3,3),d1tt0(3,3)
      dimension ttrg(3,3,6,*),ttr(3,3),tt(3,3)
      dimension t6rg(3,3,6,*)
      dimension omegag(3,6,*),omega(3)
      dimension om6gag(3,6,*)
      dimension phiv(3),dphiv(3),dphisk(3,3),gamma(3)
      dimension cncm(6,4,*)
      dimension cn(3,3),ccn(3,3),cm(3,3),ccm(3,3)
      dimension sn(3),snsk(3,3),sm(3),smsk(3,3), rn(3),rm(3)
      dimension ccnphi(3,3),phiccn(3,3),phiphi(3,3)
      dimension phisn(3,3),snphi(3,3),shpn(4),dshpn(4)
      dimension pv(3),pskw(3,3),pv1(3),distrn(3,4),distrp(3,4)
      dimension anv(3),dv(3),wkn(3,3)

      dv(1)= 1.d0
      dv(1)= 0.d0
      dv(1)= 0.d0

      DO II=1,24
        DO JJ=1,24
          STIFel(II,JJ)=0.D0
          STIFem(II,JJ)=0.D0
          STIFeg(II,JJ)=0.D0
          STIFen(II,JJ)=0.D0
          STIFep(II,JJ)=0.D0
        END DO
      END DO

      NGPT=3
      CALL GAUSS(GAUSPT,GAUSWT,NGPT)

      DO  IG=1,NGPT     ! GAUSS INTEGRATION LOOP

        U=GAUSPT(IG)
        W=GAUSWT(IG)
        T=0.5D0*(U+1.D0)

        call Frenet(idelem,t,x0,tt0,d1tt0,detj,
     &              ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4)

c       write(6,*) 'idelem=',idelem
c       write(6,*) 'tt0=', tt0

        detjh=.5D0*DETJ

        shpn(1) = ala1
        shpn(2) = ala2
        shpn(3) = ala3
        shpn(4) = ala4
        dshpn(1) = dla1/detj
        dshpn(2) = dla2/detj
        dshpn(3) = dla3/detj
        dshpn(4) = dla4/detj

        call getcnm(idelem,cncm,cm,cn,shpn)
        call getttr(idelem,ig,ttrg,ttr)
```

```
      call getomg(idelem,ig,omegag,omega)

      call otmat(ttr,tt0,tt)

      call getx(idelem,xnodel,x)


      call posvec(t,detj,x,phiv,dphiv)

      call gammav(tt0,ttr,dphiv,gamma)

      call cnmmat(tt,cn,ccn)
      call cnmmat(tt,cm,ccm)

      call sstrss(ccn,ccm,gamma,omega,sn,sm)

      call vskew(dphiv,dphisk)

      call matmat(ccn,dphisk,ccnphi)

      call matmat(dphisk,ccn,phiccn)

      call matmat(dphisk,ccnphi,phiphi)   ! ?????????

      call vskew(sn,snsk)
      call vskew(sm,smsk)

      call matmat(dphisk,snsk,phisn)
      call matmat(snsk,dphisk,snphi)

      call VECVEC(dphiv,dphiv,andphi,3)
      andphi = dsqrt(andphi)
      call VECVEC(dv,dphiv,ddphi,3)

      coe1 = ddphi/(andphi*andphi)
      coe2 = -1.d0/andphi


      do jj = 1,3
        anv(jj)= shpn(1)*distrn(jj,1)+
     &           shpn(2)*distrn(jj,2)+
     &           shpn(3)*distrn(jj,3)+
     &           shpn(4)*distrn(jj,4)
        pv(jj) = shpn(1)*distrp(jj,1)+
     &           shpn(2)*distrp(jj,2)+
     &           shpn(3)*distrp(jj,3)+
     &           shpn(4)*distrp(jj,4)
      end do


      do ii = 1,3
       do jj = 1,3
         wkn(ii,jj) =anv(ii)*(coe1*dphiv(jj) + coe2*dv(jj))
       end do
      end do

      do jj = 1,3
```

```
        pv1(jj) = tt(jj,1)*pv(1)+
&                 tt(jj,2)*pv(2)+
&                 tt(jj,3)*pv(3)
      end do

      call vskew(pv1, pskw)

      do ii = 1,4
        do jj = 1,4

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+i,6*(jj-1)+j) =
&                         stifeg(6*(ii-1)+i,6*(jj-1)+j)
&                       + 0.d0
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+i,6*(jj-1)+3+j) =
&                       stifeg(6*(ii-1)+i,6*(jj-1)+3+j)
&                 - w*dshpn(ii)*shpn(jj)*snsk(i,j)*detjh
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+3+i,6*(jj-1)+j) =
&                       stifeg(6*(ii-1)+3+i,6*(jj-1)+j)
&                 + w*shpn(ii)*dshpn(jj)*snsk(i,j)*detjh
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifeg(6*(ii-1)+3+i,6*(jj-1)+3+j) =
&                       stifeg(6*(ii-1)+3+i,6*(jj-1)+3+j)
&                 + w*(shpn(ii)*shpn(jj)*phisn(i,j)
&                 - dshpn(ii)*shpn(jj)*smsk(i,j))*detjh
            end do
          end do

        end do
      end do

      do ii = 1,4
        do jj = 1,4

          do i = 1,3
            do j = 1,3
              stifen(6*(ii-1)+i,6*(jj-1)+j) =
&                         stifen(6*(ii-1)+i,6*(jj-1)+j)
&                       + w*shpn(ii)*dshpn(jj)*wkn(i,j)*detjh
            end do
          end do
```

```
          do i = 1,3
            do j = 1,3
              stifen(6*(ii-1)+i,6*(jj-1)+3+j) =
&                     stifen(6*(ii-1)+i,6*(jj-1)+3+j)
&               + 0.d0
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifen(6*(ii-1)+3+i,6*(jj-1)+j) =
&                     stifen(6*(ii-1)+3+i,6*(jj-1)+j)
&               + 0.d0
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifen(6*(ii-1)+3+i,6*(jj-1)+3+j) =
&                    stifen(6*(ii-1)+3+i,6*(jj-1)+3+j)
&               + 0.d0
            end do
          end do

        end do
      end do

      do ii = 1,4
        do jj = 1,4

          do i = 1,3
            do j = 1,3
              stifep(6*(ii-1)+i,6*(jj-1)+j) =
&                       stifep(6*(ii-1)+i,6*(jj-1)+j)
&                     + 0.d0
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifep(6*(ii-1)+i,6*(jj-1)+3+j) =
&                     stifep(6*(ii-1)+i,6*(jj-1)+3+j)
&               + w*shpn(ii)*shpn(jj)*pskw(i,j)*detjh
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifep(6*(ii-1)+3+i,6*(jj-1)+j) =
&                     stifep(6*(ii-1)+3+i,6*(jj-1)+j)
&               + 0.d0
            end do
          end do

          do i = 1,3
            do j = 1,3
              stifep(6*(ii-1)+3+i,6*(jj-1)+3+j) =
```

```
     &                    stifep(6*(ii-1)+3+i,6*(jj-1)+3+j)
     &               + 0.d0
                 end do
               end do

             end do
           end do

         end do

         do ii = 1,24
           do jj = 1,24
             stifel(ii,jj) =  stifeg(ii,jj)
     &                     + stifen(ii,jj)
     &                     + stifep(ii,jj)
           end do
         end do

         return
         end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Calculate internal nodal loading of an element
c
c
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         subroutine qintel(idelem,x0,xnodel,ttrg,omegag,
     &                     cncm,qinte)
         IMPLICIT REAL*8 (A-H,O-Z)

         dimension qinte(24)
         dimension gauspt(6),gauswt(6)
         dimension x0(3,4,*),xnodel(3,4,*),x(3,4)
         dimension tt0(3,3),d1tt0(3,3)
         dimension ttrg(3,3,6,*),ttr(3,3),tt(3,3)
         dimension omegag(3,6,*),omega(3)
         dimension phiv(3),dphiv(3),dphisk(3,3),gamma(3)
         dimension cncm(6,4,*)
         dimension cn(3,3),ccn(3,3),cm(3,3),ccm(3,3)
         dimension sn(3),sm(3),rn(3),rm(3)
         dimension shpn(4),dshpn(4)
c         write(6,*) 'idelem=',idelem
c         write(6,*) '       '
         do ii=1,24
           qinte(ii)=0.d0
         end do

         NGPT=3
         CALL GAUSS(GAUSPT,GAUSWT,NGPT)

         DO  IG=1,NGPT     ! GAUSS INTEGRATION LOOP

           U=GAUSPT(IG)
```

```
      W=GAUSWT(IG)
      T=0.5D0*(U+1.D0)

      call Frenet(idelem,t,x0,tt0,d1tt0,detj,
     &              ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4)

      DETJH=.5D0*DETJ
      shpn(1) = ala1
      shpn(2) = ala2
      shpn(3) = ala3
      shpn(4) = ala4
      dshpn(1) = dla1/detj
      dshpn(2) = dla2/detj
      dshpn(3) = dla3/detj
      dshpn(4) = dla4/detj

      call getcnm(idelem,cncm,cm,cn,shpn)
      call getttr(idelem,ig,ttrg,ttr)
      call getomg(idelem,ig,omegag,omega)
      call otmat(ttr,tt0,tt)

      call getx(idelem,xnodel,x)
      call posvec(t,detj,x,phiv,dphiv)
      call gammav(tt0,ttr,dphiv,gamma)
c      write(6,*) 'phiv=',phiv
c      write(6,*) 'dphiv=',dphiv
c      write(6,*) 'gamma=',gamma

      call cnmmat(tt,cn,ccn)
      call cnmmat(tt,cm,ccm)
c      write(6,*) 'tt0=',tt0

c      write(6,*) 'ttr='
c      do iii=1,3
c         write(6,*) (ttr(iii,jjj),jjj=1,3)
c      end do
      call sstrss(ccn,ccm,gamma,omega,sn,sm)
c      write(6,*) 'sn=',sn
c      write(6,*) 'sm=',sm

      call vskew(dphiv,dphisk)


      call rstrss(tt,sn,sm,rn,rm)
c      write(6,*) 'rn=',rn
c      write(6,*) 'rm=',rm

      do ii = 1,4
        do i = 1,3

          qinte(6*(ii-1)+i) = qinte(6*(ii-1)+i)
     &                        +w*dshpn(ii)*sn(i)*detjh

          qinte(6*(ii-1)+3+i) = qinte(6*(ii-1)+3+i)
     &                      + w*(dshpn(ii)*sm(i)
     &                      - shpn(ii)*(dphisk(i,1)*sn(1)
     &                                 +dphisk(i,2)*sn(2)
```

```
     &                                  +dphisk(i,3)*sn(3)))*detjh

            end do
          end do

        end do

        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc
c
c
c  Calculate external nodal loading of an element
c
c  distrg(3,4) -- distributed load density value of weight on nodes
c
c  distrn(3,4) -- distributed load density value of force on nodes
c
c  distrm(3,4) -- distributed load density value of moment on nodes
c
c   qexte(24)  -- nodal load array of external load
c
c      x0(3,4,*) -- position vectors of nodes of elements (undeformed)
c
c      xnodel(3,4,*) -- position vectors of nodes of elements (current)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc
c
        subroutine qextel(idelem,x0,xnodel,
     &                    distrg,distrn,distrp,ttrg,qexte)
        IMPLICIT REAL*8 (A-H,O-Z)

        dimension distrg(3,4),distrn(3,4),distrp(3,4)
        dimension distrm(3,4),qexte(24)
        dimension gauspt(6),gauswt(6),shpn(4),d1xyz(3)
        dimension x0(3,4,*),xnodel(3,4,*),x(3,4)
        dimension tt0(3,3),d1tt0(3,3)
        dimension ttrg(3,3,6,*)
        dimension pv(3), pv1(3),ttr(3,3),tt(3,3)
c
c
c   at this time distrm = 0, no distrib. moment load
c
        do i=1,3
          do j =1,4
            distrm(i,j)=0.d0
          end do
          pv1(i) = 0.
        end do

        do ii=1,24
          qexte(ii)=0.d0
        end do
```

```
      NGPT=3
      CALL GAUSS(GAUSPT,GAUSWT,NGPT)

      DO  IG=1,NGPT     ! GAUSS INTEGRATION LOOP

        U=GAUSPT(IG)
        W=GAUSWT(IG)
        T=0.5D0*(U+1.D0)

        call Frenet(idelem,t,x0,tt0,d1tt0,detj,
     &              ala1,ala2,ala3,ala4,dla1,dla2,dla3,dla4)


        shpn(1) = ala1
        shpn(2) = ala2
        shpn(3) = ala3
        shpn(4) = ala4

        call getttr(idelem,ig,ttrg,ttr)
        call otmat(ttr,tt0,tt)

        do jj = 1,3
          pv(jj) = shpn(1)*distrp(jj,1)+
     &             shpn(2)*distrp(jj,2)+
     &             shpn(3)*distrp(jj,3)+
     &             shpn(4)*distrp(jj,4)
        end do

        do jj = 1,3
          pv1(jj) = tt(jj,1)*pv(1)+
     &              tt(jj,2)*pv(2)+
     &              tt(jj,3)*pv(3)
        end do

c        write(6,*) 'pv1=', pv1

        DETJH=.5D0*DETJ
        do ii = 1,4
          do i = 1,3

            qexte(6*(ii-1)+i) = qexte(6*(ii-1)+i)
     &       +w*shpn(ii)*(shpn(1)*(distrg(i,1)+distrn(i,1)*tt(1,1))
     &                   +shpn(2)*(distrg(i,2)+distrn(i,2)*tt(1,1))
     &                   +shpn(3)*(distrg(i,3)+distrn(i,3)*tt(1,1))
     &                   +shpn(4)*(distrg(i,4)+distrn(i,4)*tt(1,1))
     &                   + pv1(i))*detjh

            qexte(6*(ii-1)+3+i) = qexte(6*(ii-1)+3+i)
     &        +w*shpn(ii)*(shpn(1)*distrm(i,1)
     &                     +shpn(2)*distrm(i,2)
     &                     +shpn(3)*distrm(i,3)
     &                     +shpn(4)*distrm(i,4))*detjh

          end do
        end do

      end do
```

```
        return
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc
c
c  Update the
c      nodal position vector 'xnodel(ii,nod,idelem)'  element by
element
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cc
c
c
        subroutine upxnod(ntnode,nelm,MELM,NODBC,xnodel,da)

        IMPLICIT REAL*8 (A-H,O-Z)
        dimension xnodel(3,4,*), x(3,4)
        dimension MELM(NELM,4),NODBC(ntnode,6),da(*)

        do idelem = 1,nelm

          do nod = 1,4
            NODE=MELM(IdELEM,NOD)
            do ii = 1,3
              NDOF=NODBC(NODE,ii)
              IF(NDOF.NE.0) THEN
                xnodel(ii,nod,idelem) = xnodel(ii,nod,idelem)
     &                                  +da(NDOF)
              end if
            end do
          end do

        end do

        return
        end

        subroutine getx(idelem,xnodel,x)
        IMPLICIT REAL*8 (A-H,O-Z)
        dimension  xnodel(3,4,*), x(3,4)

        do nod = 1,4
          do ii = 1,3
            x(ii,nod) = xnodel(ii,nod,idelem)
          end do
        end do

        return
        end


        subroutine  getttr(idelem,ig,ttrg,ttr)
        IMPLICIT REAL*8 (A-H,O-Z)
        dimension ttrg(3,3,6,*),ttr(3,3)
```

```
do ii = 1,3
  do jj = 1,3
    ttr(ii,jj) =ttrg(ii,jj,ig,idelem)
  end do
end do
return
end


subroutine getomg(idelem,ig,omegag,omega)
IMPLICIT REAL*8 (A-H,O-Z)
dimension  omegag(3,6,*),omega(3)
do ii = 1,3
  omega(ii) = omegag(ii,ig,idelem)
end do
return
end

subroutine getgld(idelem,distgg,distrg,alambd)
IMPLICIT REAL*8 (A-H,O-Z)
dimension distgg(3,4,*),distrg(3,4)

do i = 1,3
  do j = 1,4
    distrg(i,j)=alambd*distgg(i,j,idelem)
  end do
end do

return
end


subroutine getnld(idelem,distnn,distrn,alambd)
IMPLICIT REAL*8 (A-H,O-Z)
dimension distnn(3,4,*),distrn(3,4)

do i = 1,3
  do j = 1,4
    distrn(i,j)=alambd*distnn(i,j,idelem)
  end do
end do

return
end

subroutine getpld(idelem,distpp,distrp,alambd)
IMPLICIT REAL*8 (A-H,O-Z)
dimension distpp(3,4,*),distrp(3,4)

do i = 1,3
  do j = 1,4
    distrp(i,j)=alambd*distpp(i,j,idelem)
  end do
end do

return
end
```

```
      SUBROUTINE GAUSS(X,W,N)
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(N),W(N)

      if(n.eq.2) then
        X(1)=-0.577350269189626D0
        X(2)=-X(1)
        W(1)=1.D0
        W(2)=1.D0
      end if

      if(n.eq.3) then
        X(1)=-0.774596669241483D0
        X(2)=0.d0
        X(3)=-X(1)
        W(1)=0.5555555555555555D0
        W(2)=0.8888888888888889D0
        W(3)=W(1)
      end if

      if(n.eq.4) then
        X(1)=-0.861136311594053D0
        X(2)=-0.339981043584856D0
        X(3)=-X(2)
        X(4)=-X(1)
        W(1)=0.347854845137454D0
        W(2)=0.652145154862546D0
        W(3)=W(2)
        W(4)=W(1)
      end if

      if(n.eq.5) then
        X(1)=-0.906179845938664D0
        X(2)=-0.538469310105683D0
        X(3)=0.000000000000D0
        X(4)=-X(2)
        X(5)=-X(1)
        W(1)=0.236926885056189D0
        W(2)=0.478628670499366D0
        W(3)=0.568888888888889D0
        W(4)=W(2)
        W(5)=W(1)
      end if

      if(n.eq.6) then
        X(1)=-0.932469514283197d0
        X(2)=-0.661209386466265d0
        X(3)=-0.238619186083197d0
        X(4)= 0.238619186083197d0
        X(5)= 0.661209386466265d0
        X(6)= 0.932469514283197d0
        W(1)= 0.17132449237970d0
        W(2)= 0.360761573048139d0
        W(3)= 0.467913934572691d0
```

```
          W(4)= 0.467913934572691d0
          W(5)= 0.360761573048139d0
          W(6)= 0.171324492379700d0
        end if


        RETURN
        END

        subroutine initl(nelem,x0,xnodel,ttrg,omegag)
        IMPLICIT REAL*8 (A-H,O-Z)
        dimension x0(3,4,*),xnodel(3,4,*)
        dimension ttrg(3,3,6,*),omegag(3,6,*)
        ngpt=6
        do idelem = 1,nelem
          do ig = 1,ngpt
            do ii = 1,3

              do jj = 1,3
                ttrg(ii,jj,ig,idelem) = 0.d0
              end do
              ttrg(ii,ii,ig,idelem) = 1.d0

              omegag(ii,ig,idelem) = 0.d0

            end do
          end do

          do nod = 1,4
            do ii = 1,3
              xnodel(ii,nod,idelem) = x0(ii,nod,idelem)
            end do
          end do

        end do


        return
        end

c
c
c
c



        subroutine tempsv(maxeqn,nelem,
     &                    xnodel,ttrg,omegag,da,dq,
     &                    sxnode,sttrg,somegg,sda,sdq)
        IMPLICIT REAL*8 (A-H,O-Z)
        dimension xnodel(3,4,*)
        dimension sxnode(3,4,*)
        dimension ttrg(3,3,6,*),omegag(3,6,*)
        dimension sttrg(3,3,6,*),somegg(3,6,*)
        dimension da(*),dq(*),sda(*),sdq(*)
        ngpt=6
```

```
      do idelem = 1,nelem
        do ig = 1,ngpt
          do ii = 1,3
            do jj = 1,3
              sttrg(ii,jj,ig,idelem)=ttrg(ii,jj,ig,idelem)
            end do
            somegg(ii,ig,idelem)=omegag(ii,ig,idelem)
          end do
        end do

        do nod = 1,4
          do ii = 1,3
            sxnode(ii,nod,idelem) = xnodel(ii,nod,idelem)
          end do
        end do

      end do

      do ii = 1,maxeqn
        sda(ii)=da(ii)
        sdq(ii)=dq(ii)
      end do
c     OPEN (UNIT=,FILE='archcode.in',STATUS='UNKNOWN')


      return
      end
      subroutine tempgt(maxeqn,nelem,
     &                  xnodel,ttrg,omegag,da,dq,
     &                  sxnode,sttrg,somegg,sda,sdq)
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension xnodel(3,4,*)
      dimension sxnode(3,4,*)
      dimension ttrg(3,3,6,*),omegag(3,6,*)
      dimension sttrg(3,3,6,*),somegg(3,6,*)
      dimension da(*),dq(*),sda(*),sdq(*)
      ngpt=6
      do idelem = 1,nelem
        do ig = 1,ngpt
          do ii = 1,3
            do jj = 1,3
              ttrg(ii,jj,ig,idelem)=sttrg(ii,jj,ig,idelem)
            end do
            omegag(ii,ig,idelem)=somegg(ii,ig,idelem)
          end do
        end do

        do nod = 1,4
          do ii = 1,3
            xnodel(ii,nod,idelem)=sxnode(ii,nod,idelem)
          end do
        end do

      end do

      do ii = 1,maxeqn
        da(ii)=sda(ii)
```

```
          dq(ii)=sdq(ii)
        end do

        return
        end


c234567890123456789012345678901234567890123456789012345678901234567890123456789012
2

        SUBROUTINE ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)
        IMPLICIT REAL*8 (A-H,O-Z)

        DIMENSION x0(3,4,*),xnodel(3,4,*),cncm(6,4,*)
        DIMENSION MELM(NELM,4),NODBC(NTNODE,6)
        DIMENSION mhing1(2,*),mhing3(2,*)
        DIMENSION hing1(5,*),idofh1(*)
        DIMENSION LM(24),MAXH(MAXEQN),MHT(MAXEQN)
        dimension ttrg(3,3,6,*),omegag(3,6,*),da(*)
        dimension t6rg(3,3,6,*),om6gag(3,6,*)
        DIMENSION AST(MAXEQN,MAXEQN)
        DIMENSION MAXA(24),STIFEL(24,24)
        dimension qext0(*),qint(*),qexte0(24),qinte(24)
        dimension dae(24)
        dimension distgg(3,4,*),distnn(3,4,*),distpp(3,4,*),ptload(*)
        dimension distrg(3,4),distrn(3,4),distrp(3,4)
        DIMENSION HPQ(3),SPQA(6,6),SPQC(6,6),WK(6,6),HPQsk(3,3)

        MAX = MAXEQN+1
        if(LCODE.EQ.2) then
          do ii = 1, MAX
             qext0(ii) = ptload(ii) !
             qint(ii) = 0.0
          end do

        end if

        DO IDELEM=1,NELM


          if(LCODE.EQ.1) then   ! assemble stiffness matrix only

            call getnld(idelem,distnn,distrn,alambd)
            call getpld(idelem,distpp,distrp,alambd)

            call elstif(idelem,x0,xnodel,ttrg,omegag,CNCM,
     &                  t6rg,om6gag,distrn,distrp,stifel)
            ipr=2
            if(ipr.eq.1) then
              open(20,file='stiff.dat',status='unknown')
```

```
              nepr=nelm/2+1
              if(idelem.eq.nepr) then
                write(20,*) ((stifel(ii,jj),ii=1,24),jj=1,24)
              end if
            end if
          end if   !  end of assembling stiffness matrix only

          DO IT=1,24
            LM(IT)=0
          END DO

          IT=0
          DO INOD=1,4
            NODNUM=MELM(IDELEM,INOD)
            DO IDOF=1,6
              IT=IT+1
              LM(IT)=NODBC(NODNUM,IDOF)
            END DO
          END DO
              if(LCODE.EQ.2) then     ! assemble load vectors only
                call clvec(dae,24)       !get element node displ
increment

                do nod=1,4
                  NODE=MELM(IdELEM,NOD)
                  DO IDOF=1,6
                    NDOF=NODBC(NODE,IDOF)
                    dae(6*(NOD-1)+IDOF)=0.d0
                    IF(NDOF.NE.0) THEN
                      dae(6*(NOD-1)+IDOF)=da(NDOF)
                    END IF
                  END DO
                END DO       !get element node displ increment
c               write(6,*) 'dae=',dae

                call qintel(idelem,x0,xnodel,t6rg,om6gag,
     &                    CNCM,qinte)
                almbd=1.
                call getgld(idelem,distgg,distrg,almbd)
                call getnld(idelem,distnn,distrn,almbd)
                call getpld(idelem,distpp,distrp,almbd)

                call qextel(idelem,x0,xnodel,
     &                    distrg,distrn,distrp,ttrg,qexte0)

              end if   !  end of assembling load vectors only
          DO IR=1,24

            IST=LM(IR)
            IF(IST.NE.0) THEN
              if(LCODE.EQ.1) then   ! assemble stiffness matrix only
                DO JC=1,24
                  IF(LM(JC).NE.0) THEN
                    JST=LM(JC)

                    AST(IST,JST)=AST(IST,JST)+STIFEL(IR,JC)
                  END IF
```

```
              END DO
            end if   !  end of assembling stiffness matrix only
            if(LCODE.EQ.2) then     ! assemble load vectors only

              qext0(ist) = qext0(ist) + qexte0(ir)
              qint(ist) = qint(ist) + qinte(ir)

            end if   !  end of assembling load vectors only

          END IF
        END DO

      END DO



      if(LCODE.EQ.1) then
c       write(22,*) 'ast0=', (ast(ii,ii), ii=1, maxeqn)
      DO nhing =1, nhing1
        do i=1,3
           HPQ(i)=hing1(i,nhing)
        end do
        IP=mhing1(1,nhing)
        IQ=mhing1(2,nhing)
c             write(6,*)'HPQ=',HPQ
        call vskew(HPQ,HPQsk)
        do i = 1,3
          IR=NODBC(IP,3+i)

          if(IR.ne.0) then
            do j=1,3
              IC=NODBC(IP,3+j)
              if(IC.ne.0) then
                AST(IR,IC)=AST(IR,IC)+
     &            hing1(4,nhing)*(hing1(5,nhing)*HPQsk(i,j)+
     &                            HPQ(i)*HPQ(j))

              end if
              IC=NODBC(IQ,3+j)
              if(IC.ne.0) then
                AST(IR,IC)=AST(IR,IC)-hing1(4,nhing)*HPQ(i)*HPQ(j)
              end if

            end do
          end if
        end do

        do i = 1,3
          IR=NODBC(IQ,3+i)
          if(IR.ne.0) then
            do j=1,3
              IC=NODBC(IP,3+j)
              if(IC.ne.0) then
                AST(IR,IC)=AST(IR,IC)-hing1(4,nhing)*HPQ(i)*HPQ(j)
              end if
              IC=NODBC(IQ,3+j)
              if(IC.ne.0) then
```

```
                AST(IR,IC)=AST(IR,IC)+
     &            hing1(4,nhing)*(hing1(5,nhing)*HPQsk(i,j)+
     &                            HPQ(i)*HPQ(j))
              end if

            end do
          end if
        end do

        call HING1M(HPQ,SPQA,SPQC,iching)
c       write(6,*) 'SPQA=',SPQA
c       write(6,*) 'SPQC=',SPQC
        idofh1(nhing)=iching

        DO nod = 1,NTNODE

          do i = 1,6
            do j = 1,6
              WK(i,j)=0.
            end do
          end do

          do IDOFR = 1,6

            IR=NODBC(nod,IDOFR)
            if(IR.NE.0) then
              do IDOFC = 1,6
                ICP=NODBC(IP,IDOFC)
                if(ICP.NE.0) then
                  do j = 1,6
                    ICQ=NODBC(IQ,j)
                    if(ICQ.ne.0) then
                      AST(IR,ICP)=AST(IR,ICP)
     &                           +AST(IR,ICQ)*SPQA(j,IDOFC)
                    end if
                  end do
                end if
              end do
            end if

          end do

        end do



        DO nod = 1,NTNODE
          do i = 1,6
            do j = 1,6
              WK(i,j)=0.
            end do
          end do

          do IDOFR = 1,6

            IR=NODBC(nod,IDOFR)
            if(IR.NE.0) then
```

```
            do IDOFC = 1,6
              ICQ=NODBC(IQ,IDOFC)
              if(ICQ.NE.0) then
                do j = 1,6
                  jj = NODBC(IQ,j)
                  if(jj.ne.0) then
                    WK(IDOFR,IDOFC) = WK(IDOFR,IDOFC)+
&                                     AST(IR,jj)*SPQC(j,IDOFC)
                  end if
                end do
              end if
            end do
          end if

        end do

        do IDOFR = 1,6

          IR=NODBC(nod,IDOFR)
          if(IR.NE.0) then
            do IDOFC = 1,6
              ICQ=NODBC(IQ,IDOFC)
              if(ICQ.NE.0) then
                AST(IR,ICQ) = WK(IDOFR,IDOFC)
              end if
            end do
          end if

        end do

      end do

      DO nod = 1,NTNODE
        do IDOFC = 1,6
          IC=NODBC(nod,IDOFC)
          if(IC.NE.0) then
            do IDOFR = 1,6
              IRP=NODBC(IP,IDOFR)
              IRQ=NODBC(IQ,IDOFR)
              if(IRP.NE.0) then
                if(IRQ.NE.0) then
                  AST(IRP,IC)=AST(IRP,IC)
&                             +AST(IRQ,IC)
                end if
              end if
            end do
          end if
        end do
      end do

      DO IDOFR=1,3
        IRQ=NODBC(IQ,IDOFR)
        IRP=NODBC(IP,IDOFR)
        if(IRQ.ne.0.) then

            DO nod = 1,NTNODE
              DO IDOFC=1,6
```

```
                    IC=NODBC(nod,IDOFC)
                    if(IC.ne.0) then
                      AST(IRQ,IC)=0.
                      if(IRQ.eq.IC) then
                        AST(IRQ,IC)=1.
                      end if
                    end if
                  end do
                end do

            end if

          end do
ccccccccccc
          DO IDOFR=1,3
            IRQ=NODBC(IQ,IDOFR)
            IRP=NODBC(IP,IDOFR)
            if(IRP.ne.0.) then
            if(IRQ.eq.0.) then

                DO nod = 1,NTNODE
                  DO IDOFC=1,6
                    IC=NODBC(nod,IDOFC)
                    if(IC.ne.0) then
                      AST(IRP,IC)=0.
                      if(IRP.eq.IC) then
                        AST(IRP,IC)=1.
                      end if
                    end if
                  end do
                end do

            end if
            end if

          end do

ccccccccccccc
          DO IDOFR=1,3
            IR=NODBC(IQ,IDOFR+3)
            if(IR.ne.0) then
              DO nod = 1,NTNODE
                DO IDOFC=1,6
                  IC=NODBC(nod,IDOFC)
                  if(IC.ne.0) then
                    wk1=0.
                    do k = 1,3
                      kr = NODBC(IQ,k+3)
                      if(kr.ne.0) then
                        wk1 = wk1 + HPQ(IDOFR)*HPQ(k)*AST(kr,IC)
                      end if
                    end do
                    AST(IR,IC) = wk1
                  end if
                end do
              end do
            end if
```

```
            end do

            DO nod = 1,NTNODE
              do IDOFC = 1,6
                IC=NODBC(nod,IDOFC)
                if(IC.NE.0) then
                  do IDOFR = 4,6
                    IRP=NODBC(IP,IDOFR)
                    IRQ=NODBC(IQ,IDOFR)
                    if(IRP.NE.0) then
                      if(IRQ.NE.0) then
                        AST(IRP,IC)=AST(IRP,IC)
     &                             -AST(IRQ,IC)
                      end if
                    end if
                  end do
                end if
              end do
            end do

            DO IDOFR=1,3
              IRQ=NODBC(IQ,IDOFR+3)
              IRP=NODBC(IP,IDOFR+3)
              if(IRQ.ne.0) then
                DO nod = 1,NTNODE
                  DO IDOFC=1,6
                    IC=NODBC(nod,IDOFC)
                    if(IC.ne.0) then
                      if(IDOFR.ne.iching) then
                        if(IC.eq.IRQ) then
                          AST(IRQ,IC) = 1.
                        end if
                      end if
                    end if
                  end do
                end do
              end if
            end do
ccccccccccc
            DO IDOFR=1,3
              IRQ=NODBC(IQ,IDOFR+3)
              IRP=NODBC(IP,IDOFR+3)
              if(IRP.ne.0) then
              if(IRQ.eq.0) then
                DO nod = 1,NTNODE
                  DO IDOFC=1,6
                    IC=NODBC(nod,IDOFC)
                    if(IC.ne.0) then
                      if(IDOFR.ne.iching) then
                        if(IC.eq.IRP) then
                          AST(IRP,IC) = 1.
                        end if
                      end if
                    end if
                  end do
                end do
              end if
```

```
            end if
           end do
ccccccccccc

         end do
c        write(22,*) 'ast1=',(ast(ii,ii), ii=1, maxeqn)
         end if

         if(LCODE.EQ.2) then
c         write(6,*) 'qext3=',(qext0(iii),iii=1,maxeqn)
c         write(6,*) 'qint3=',(qint(iii),iii=1,maxeqn)
         do nhing =1, nhing1

           do i=1,3
              HPQ(i)=hing1(i,nhing)
           end do

           IP=mhing1(1,nhing)
           IQ=mhing1(2,nhing)
c          write(6,*) 'spr=', hing1(4,nhing),'thet=',hing1(5,nhing)
           do i=1,3
             IR=NODBC(IP,3+i)
             wk1 =0.
             if(IR.ne.0) then

               qint(IR)=qint(IR)+hing1(4,nhing)*hing1(5,nhing)*HPQ(i)
             end if
           end do

           do i=1,3
             IR=NODBC(IQ,3+i)
             if(IR.ne.0) then
               qint(IR)=qint(IR)-hing1(4,nhing)*hing1(5,nhing)*HPQ(i)
             end if
           end do
c         write(6,*) 'qext4=',(qext0(iii),iii=1,maxeqn)
c         write(6,*) 'qint4=',(qint(iii),iii=1,maxeqn)
           do i = 1,6
             IRP = NODBC(IP,i)
             IRQ = NODBC(IQ,i)
             if(IRP.NE.0) then
               if(IRQ.NE.0) then
                 qext0(IRP) = qext0(IRP)+qext0(IRQ)
                 qint(IRP) = qint(IRP)+qint(IRQ)
               end if
             end if
           end do

           do i = 1,3
             IRQ = NODBC(IQ,i)
             if(IRQ.NE.0) then
               qext0(IRQ) = 0.
               qint(IRQ) = 0.
             end if
           end do
ccccccc
           do i = 1,3
```

```
              IRQ = NODBC(IQ,i)
              IRP = NODBC(IP,i)
              if(IRP.NE.0) then

              if(IRQ.EQ.0) then
                qext0(IRP) = 0.
                qint(IRP) = 0.
              end if

              end if
            end do
ccccccccc
            do i = 1,3
                IR = NODBC(IQ,i+3)
                if(IR.NE.0) then
                  wk1=0.
                  wk2=0.
                  do k = 1, 3
                    kr = NODBC(IQ,k+3)
                    if(kr.ne.0) then
                      wk1 = wk1 + HPQ(i)*HPQ(k)*qext0(kr)
                      wk2 = wk2 + HPQ(i)*HPQ(k)*qint(kr)
                    end if
                  end do
c                 write(6,*) 'wk1,wk2 =', wk1,wk2
                  qext0(IR) = wk1
                  qint(IR) = wk2
                end if
            end do


            do i = 1,3
                IRP = NODBC(IP,i+3)
                IRQ = NODBC(IQ,i+3)
                if(IRP.NE.0) then
                  if(IRQ.NE.0) then
                    qext0(IRP) = qext0(IRP)-qext0(IRQ)
                    qint(IRP) = qint(IRP)-qint(IRQ)
                  end if
                end if
            end do
ccccccccc
            do i = 1,3
                IRP = NODBC(IP,i+3)
                IRQ = NODBC(IQ,i+3)
                if(IRP.NE.0) then
                  if(IRQ.EQ.0) then
                    qext0(IRP) = 0.
                    qint(IRP) = 0.
                  end if
                end if
            end do
cccccccccc
c           write(6,*) 'qext5=',(qext0(iii),iii=1,maxeqn)
c           write(6,*) 'qint5=',(qint(iii),iii=1,maxeqn)
           end do
           end if
```

```
      RETURN
      END


c234567890123456789012345678901234567890123456789012345678901234567890123456789012
2
      SUBROUTINE ASSEMN(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                  nhing1,nhing3,
     &                  mhing1,mhing3,
     &                  hing1,idofh1,
     &                  distgg,distnn,distpp,ptload,alambd,
     &                  ttrg,t6rg,omegag,om6gag,da,
     &                  LM,MAXH,MHT,NWK,MK,
     &                  AST,qint,qext0,LCODE)
      IMPLICIT REAL*8 (A-H,O-Z)

      DIMENSION x0(3,4,*),xnodel(3,4,*),cncm(6,4,*)
      DIMENSION MELM(NELM,4),NODBC(NTNODE,6)
      DIMENSION mhing1(2,*),mhing3(2,*)
      DIMENSION hing1(5,*),idofh1(*)
      DIMENSION LM(24),MAXH(MAXEQN),MHT(MAXEQN)
      dimension ttrg(3,3,6,*),omegag(3,6,*),da(*)
      dimension t6rg(3,3,6,*),om6gag(3,6,*)
      DIMENSION AST(MAXEQN,MAXEQN)
      DIMENSION MAXA(24),STIFEL(24,24)
      dimension qext0(*),qint(*),qexte0(24),qinte(24)
      dimension dae(24)
      dimension distgg(3,4,*),distnn(3,4,*),distpp(3,4,*),ptload(*)
      dimension distrg(3,4),distrn(3,4),distrp(3,4)
      DIMENSION HPQ(3),SPQA(6,6),SPQC(6,6),WK(6,6),HPQsk(3,3)

      MAX = MAXEQN+1

      DO IDELEM=1,NELM

        if(LCODE.EQ.1) then   ! assemble stiffness matrix only
          call getnld(idelem,distnn,distrn,alambd)
          call getpld(idelem,distpp,distrp,alambd)
          call elstin(idelem,x0,xnodel,ttrg,omegag,CNCM,
     &                t6rg,om6gag,distrn,distrp,stifel)

        end if   !  end of assembling stiffness matrix only

        DO IT=1,24
          LM(IT)=0
        END DO

        IT=0
        DO INOD=1,4
          NODNUM=MELM(IDELEM,INOD)
          DO IDOF=1,6
            IT=IT+1
            LM(IT)=NODBC(NODNUM,IDOF)
```

```
            END DO
          END DO

          DO IR=1,24

            IST=LM(IR)
            IF(IST.NE.0) THEN
              if(LCODE.EQ.1) then   ! assemble stiffness matrix only
                DO JC=1,24
                  IF(LM(JC).NE.0) THEN
                    JST=LM(JC)

                    AST(IST,JST)=AST(IST,JST)+STIFEL(IR,JC)
                  END IF
                END DO
              end if   !  end of assembling stiffness matrix only

            END IF
          END DO

        END DO



        if(LCODE.EQ.1) then
c       write(22,*) 'ast0=', (ast(ii,ii), ii=1, maxeqn)
        DO nhing =1, nhing1
          do i=1,3
             HPQ(i)=hing1(i,nhing)
          end do
          IP=mhing1(1,nhing)
          IQ=mhing1(2,nhing)
c               write(6,*)'HPQ=',HPQ
          call vskew(HPQ,HPQsk)
          do i = 1,3
            IR=NODBC(IP,3+i)

            if(IR.ne.0) then
              do j=1,3
                IC=NODBC(IP,3+j)
                if(IC.ne.0) then
c                 AST(IR,IC)=AST(IR,IC)+
c     &              hing1(4,nhing)*(hing1(5,nhing)*HPQsk(i,j)+
c     &                              HPQ(i)*HPQ(j))
                  AST(IR,IC)=AST(IR,IC)+
     &              hing1(4,nhing)*(hing1(5,nhing)*HPQsk(i,j)+
     &                              0.)
                end if
                IC=NODBC(IQ,3+j)
                if(IC.ne.0) then
c                 AST(IR,IC)=AST(IR,IC)-hing1(4,nhing)*HPQ(i)*HPQ(j)
                  AST(IR,IC)=AST(IR,IC)-0.
                end if

              end do
            end if
          end do
```

```
          do i = 1,3
            IR=NODBC(IQ,3+i)
            if(IR.ne.0) then
              do j=1,3
                IC=NODBC(IP,3+j)
                if(IC.ne.0) then
c                 AST(IR,IC)=AST(IR,IC)-hing1(4,nhing)*HPQ(i)*HPQ(j)
                  AST(IR,IC)=AST(IR,IC)-0.
                end if
                IC=NODBC(IQ,3+j)
                if(IC.ne.0) then
c                 AST(IR,IC)=AST(IR,IC)+
c     &              hing1(4,nhing)*(hing1(5,nhing)*HPQsk(i,j)+
c     &                              HPQ(i)*HPQ(j))
                  AST(IR,IC)=AST(IR,IC)+
     &              hing1(4,nhing)*(hing1(5,nhing)*HPQsk(i,j)+
     &                              0.)
                end if

              end do
            end if
          end do

          call HING1M(HPQ,SPQA,SPQC,iching)
c         write(6,*) 'SPQA=',SPQA
c         write(6,*) 'SPQC=',SPQC
          idofh1(nhing)=iching

          DO nod = 1,NTNODE

            do i = 1,6
              do j = 1,6
                WK(i,j)=0.
              end do
            end do

            do IDOFR = 1,6

              IR=NODBC(nod,IDOFR)
              if(IR.NE.0) then
                do IDOFC = 1,6
                  ICP=NODBC(IP,IDOFC)
                  if(ICP.NE.0) then
                    do j = 1,6
                      ICQ=NODBC(IQ,j)
                      if(ICQ.ne.0) then
                        AST(IR,ICP)=AST(IR,ICP)
     &                              +AST(IR,ICQ)*SPQA(j,IDOFC)
                      end if
                    end do
                  end if
                end do
              end if

            end do
```

```
      end do



      DO nod = 1,NTNODE
        do i = 1,6
          do j = 1,6
            WK(i,j)=0.
          end do
        end do

        do IDOFR = 1,6

          IR=NODBC(nod,IDOFR)
          if(IR.NE.0) then
            do IDOFC = 1,6
              ICQ=NODBC(IQ,IDOFC)
              if(ICQ.NE.0) then
                do j = 1,6
                  jj = NODBC(IQ,j)
                  if(jj.ne.0) then
                    WK(IDOFR,IDOFC) = WK(IDOFR,IDOFC)+
     &                              AST(IR,jj)*SPQC(j,IDOFC)
                  end if
                end do
              end if
            end do
          end if

        end do

        do IDOFR = 1,6

          IR=NODBC(nod,IDOFR)
          if(IR.NE.0) then
            do IDOFC = 1,6
              ICQ=NODBC(IQ,IDOFC)
              if(ICQ.NE.0) then
                AST(IR,ICQ) = WK(IDOFR,IDOFC)
              end if
            end do
          end if

        end do

      end do

      DO nod = 1,NTNODE
        do IDOFC = 1,6
          IC=NODBC(nod,IDOFC)
          if(IC.NE.0) then
            do IDOFR = 1,6
              IRP=NODBC(IP,IDOFR)
              IRQ=NODBC(IQ,IDOFR)
              if(IRP.NE.0) then
                if(IRQ.NE.0) then
                  AST(IRP,IC)=AST(IRP,IC)
```

```
     &                            +AST(IRQ,IC)
                    end if
                  end if
                end do
              end if
            end do
          end do

          DO IDOFR=1,3
            IRQ=NODBC(IQ,IDOFR)
            IRP=NODBC(IP,IDOFR)
            if(IRQ.ne.0.) then

                DO nod = 1,NTNODE
                  DO IDOFC=1,6
                    IC=NODBC(nod,IDOFC)
                    if(IC.ne.0) then
                      AST(IRQ,IC)=0.
                      if(IRQ.eq.IC) then
                        AST(IRQ,IC)=1.
                      end if
                    end if
                  end do
                end do

            end if

          end do
ccccccccccc
          DO IDOFR=1,3
            IRQ=NODBC(IQ,IDOFR)
            IRP=NODBC(IP,IDOFR)
            if(IRP.ne.0.) then
            if(IRQ.eq.0.) then

                DO nod = 1,NTNODE
                  DO IDOFC=1,6
                    IC=NODBC(nod,IDOFC)
                    if(IC.ne.0) then
                      AST(IRP,IC)=0.
                      if(IRP.eq.IC) then
                        AST(IRP,IC)=1.
                      end if
                    end if
                  end do
                end do

            end if
            end if

          end do

ccccccccccccc
          DO IDOFR=1,3
            IR=NODBC(IQ,IDOFR+3)
            if(IR.ne.0) then
              DO nod = 1,NTNODE
```

```
                DO IDOFC=1,6
                  IC=NODBC(nod,IDOFC)
                  if(IC.ne.0) then
                    wk1=0.
                    do k = 1,3
                      kr = NODBC(IQ,k+3)
                      if(kr.ne.0) then
                        wk1 = wk1 + HPQ(IDOFR)*HPQ(k)*AST(kr,IC)
                      end if
                    end do
                    AST(IR,IC) = wk1
                  end if
                end do
              end do
            end if
          end do

          DO nod = 1,NTNODE
            do IDOFC = 1,6
              IC=NODBC(nod,IDOFC)
              if(IC.NE.0) then
                do IDOFR = 4,6
                  IRP=NODBC(IP,IDOFR)
                  IRQ=NODBC(IQ,IDOFR)
                  if(IRP.NE.0) then
                    if(IRQ.NE.0) then
                      AST(IRP,IC)=AST(IRP,IC)
     &                            -AST(IRQ,IC)
                    end if
                  end if
                end do
              end if
            end do
          end do

          DO IDOFR=1,3
            IRQ=NODBC(IQ,IDOFR+3)
            IRP=NODBC(IP,IDOFR+3)
            if(IRQ.ne.0) then
              DO nod = 1,NTNODE
                DO IDOFC=1,6
                  IC=NODBC(nod,IDOFC)
                  if(IC.ne.0) then
                    if(IDOFR.ne.iching) then
                      if(IC.eq.IRQ) then
                        AST(IRQ,IC) = 1.
                      end if
                    end if
                  end if
                end do
              end do
            end if
          end do
ccccccccccc
          DO IDOFR=1,3
            IRQ=NODBC(IQ,IDOFR+3)
            IRP=NODBC(IP,IDOFR+3)
```

```
            if(IRP.ne.0) then
            if(IRQ.eq.0) then
              DO nod = 1,NTNODE
                DO IDOFC=1,6
                  IC=NODBC(nod,IDOFC)
                  if(IC.ne.0) then
                    if(IDOFR.ne.iching) then
                      if(IC.eq.IRP) then
                        AST(IRP,IC) = 1.
                      end if
                    end if
                  end if
                end do
              end do
            end if
            end if
           end do
ccccccccccc

        end do
c       write(22,*) 'ast1=',(ast(ii,ii), ii=1, maxeqn)
        end if



        RETURN
        END



C
      SUBROUTINE LUDCMP(A,N,NP,INDX,D)
        IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NP,NP),INDX(N),VV(10000)
C
      TINY=1.D-20
      D=1.D0
      DO 12 I=1,N
         AAMAX=0.D0
         DO 11 J=1,N
            IF(DABS(A(I,J)).GT.AAMAX) AAMAX=DABS(A(I,J))
   11    CONTINUE
         IF(DABS(AAMAX).LT.TINY/10.D0) THEN
            WRITE(6,100)
  100       FORMAT(1X,'SINGULAR MATRIX')
            STOP
         ENDIF
         VV(I)=1.D0/AAMAX
   12 CONTINUE
      DO 19 J=1,N
         IF(J.GT.1) THEN
            DO 14 I=1,J-1
               SUM=A(I,J)
               IF(I.GT.1) THEN
                  DO 13 K=1,I-1
                     SUM=SUM-A(I,K)*A(K,J)
   13             CONTINUE
```

```
                  A(I,J)=SUM
               ENDIF
   14       CONTINUE
         ENDIF
         AAMAX=0.D0
         DO 16 I=J,N
            SUM=A(I,J)
            IF(J.GT.1) THEN
               DO 15 K=1,J-1
                  SUM=SUM-A(I,K)*A(K,J)
   15          CONTINUE
               A(I,J)=SUM
            ENDIF
            DUM=VV(I)*DABS(SUM)
            IF(DUM.GE.AAMAX) THEN
               IMAX=I
               AAMAX=DUM
            ENDIF
   16    CONTINUE
         IF(J.NE.IMAX) THEN
            DO 17 K=1,N
               DUM=A(IMAX,K)
               A(IMAX,K)=A(J,K)
               A(J,K)=DUM
   17       CONTINUE
            D=-D
            VV(IMAX)=VV(J)
         ENDIF
         INDX(J)=IMAX
         IF(J.NE.N) THEN
            IF(DABS(A(J,J)).LT.TINY) A(J,J)=TINY
            DUM=1.D0/A(J,J)
            DO 18 I=J+1,N
               A(I,J)=A(I,J)*DUM
   18       CONTINUE
         ENDIF
   19 CONTINUE
      IF(DABS(A(N,N)).LT.TINY) A(N,N)=TINY
      RETURN
      END
C
      SUBROUTINE LUBKSB(A,N,NP,INDX,B)
        IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NP,NP),INDX(N),B(N)
C
      II=0
      DO 12 I=1,N
         LL=INDX(I)
         SUM=B(LL)
         B(LL)=B(I)
         IF(II.NE.0) THEN
            DO 11 J=II,I-1
               SUM=SUM-A(I,J)*B(J)
   11       CONTINUE
         ELSE IF(SUM.NE.0.D0) THEN
            II=I
         ENDIF
```

```
         B(I)=SUM
   12 CONTINUE
      DO 14 I=N,1,-1
         SUM=B(I)
         IF(I.LT.N) THEN
            DO 13 J=I+1,N
               SUM=SUM-A(I,J)*B(J)
   13       CONTINUE
         ENDIF
         B(I)=SUM/A(I,I)
   14 CONTINUE
      RETURN
      END
C
      SUBROUTINE INVMAT(A,Y,N,NP)
C
C     PROGRAM FROM NUMERICAL RECIPES BY PRESS, WILLIAM H. ET. AL.
C
        IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NP,NP),Y(NP,NP),INDX(10000)
C
      DO 12 J=1,N
         DO 11 I=1,N
            Y(I,J)=0.D0
   11    CONTINUE
         Y(J,J)=1.D0
   12 CONTINUE
      CALL LUDCMP(A,N,NP,INDX,D)
      DO 13 J=1,N
         CALL LUBKSB(A,N,NP,INDX,Y(1,J))
   13 CONTINUE
      RETURN
      END

        SUBROUTINE CLVEC(A,N)
C
C
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION A(N)
        DO 1 I=1,N
        A(I)=0.0D0
 1      CONTINUE
        RETURN
        END
C
CC
        SUBROUTINE VECVEC(A,B,C,M)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION A(M),B(M)
C
        C=0.D0
        DO 1 I=1,M
        C=C+A(I)*B(I)
 1      CONTINUE
        RETURN
        END
C
```

```
C
C
      SUBROUTINE COLSOL(A,V,MAXA,NN,NWK,NNM,KKK)
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NWK),V(1),MAXA(1)
      IF(KKK-2)40,150,150
 40   DO 140 N=1,NN
      KN=MAXA(N)
      KL=KN+1
      KU=MAXA(N+1)-1
      KH=KU-KL
      IF(KH)110,90,50
 50   K=N-KH
      IC=0
      KLT=KU
      DO 80 J=1,KH
      IC=IC+1
      KLT=KLT-1
      KI=MAXA(K)
      ND=MAXA(K+1)-KI-1
      IF(ND)80,80,60
 60   KK=MIN0(IC,ND)
      C=0.0D0
      DO 70 L=1,KK
 70   C=C+A(KI+L)*A(KLT+L)
      A(KLT)=A(KLT)-C
 80   K=K+1
 90   K=N
      B=0.0D0
      DO 100 KK=KL,KU
      K=K-1
      KI=MAXA(K)
      C=A(KK)/A(KI)
      B=B+C*A(KK)
 100  A(KK)=C
      A(KN)=A(KN)-B
 110  IF(A(KN))120,120,140
c 120   PRINT 2000,N,A(KN)
 120  continue
c      STOP
 140  CONTINUE
      RETURN
C
C     REDUCE RIGHT HAND SIDE LOAD VECTER
C
 150  DO 180 N=1,NN
      KL=MAXA(N)+1
      KU=MAXA(N+1)-1
      IF(KU-KL)180,160,160
 160   K=N
       C=0.0D0
       DO 170 KK=KL,KU
       K=K-1
 170   C=C+A(KK)*V(K)
       V(N)=V(N)-C
```

```
 180     CONTINUE
C
C
C
         DO 200 N=1,NN
         K=MAXA(N)
 200     V(N)=V(N)/A(K)
         IF(NN.EQ.1)RETURN
         N=NN
         DO 230 L=2,NN
         KL=MAXA(N)+1
         KU=MAXA(N+1)-1
         IF(KU-KL)230,210,210
 210     K=N
         DO 220 KK=KL,KU
         K=K-1
 220     V(K)=V(K)-A(KK)*V(N)
 230     N=N-1
         RETURN
 2000    FORMAT(//10X,'STOP--MATRIX NOT POSITIVE DEFINITE',//10X,
     &'NON POSITIVE PIVOT FOR EQUATION',I5//10X,'PIVOT=',E20.12)
          END
C
C
C
        SUBROUTINE DECOMP1(NDIM,N,A,COND,IPVT,NCHG,WORK)
        IMPLICIT REAL*8 (A-H, O-Z)
        DIMENSION A(NDIM,N), WORK(N), IPVT(N)
        nchg=0
        IPVT(N) = 1
        IF(N.EQ.1) GO TO 80
        NM1 = N-1
        ANORM = 0.0
        DO 10 J=1,N
        T=0.0
        DO 5  I = 1,N
        T = T +DABS(A(I,J))
  5     CONTINUE
        IF(T.GT.ANORM) ANORM = T
  10    CONTINUE
        DO 35 K = 1,NM1
        KP1 = K+1
        M=K
        DO 15 I=KP1,N
        IF(DABS(A(I,K)).GT.DABS(A(M,K))) M=I
  15    CONTINUE
        IPVT(K) = M
        IF(M.NE.K) IPVT(N) = -IPVT(N)
        IF(M.NE.K) NCHG=NCHG+1
        T=A(M,K)
        A(M,K) =A(K,K)
        A(K,K) = T
        IF(T.EQ.0.0) GO TO 35
        DO 20 I=KP1, N
        A(I,K) = -A(I,K)/T
  20    CONTINUE
        DO 30 J = KP1,N
```

```
      T = A(M,J)
      A(M,J) = A(K,J)
      A(K,J) = T
      IF(T.EQ.0.0) GO TO 30
      DO 25 I = KP1, N
      A(I,J) = A(I,J) + A(I,K)*T
25    CONTINUE
30    CONTINUE
35    CONTINUE
      DO 50 K = 1,N
      T = 0.0
      IF(K.EQ.1) GO TO 45
      KM1 = K-1
      DO 40 I = 1,KM1
      T = T + A(I,K)*WORK(I)
40    CONTINUE
45    EK = 1.0
      IF(T.LT.0.0) EK = - 1.0
      IF(A(K,K).EQ.0.0) GO TO 90
      WORK(K) = -(EK+T)/A(K,K)
50    CONTINUE
      DO 60 KB = 1,NM1
      K = N - KB
      T = 0.0
      KP1 = K + 1
      DO 55 I = KP1, N
      T = T + A(I,K)*WORK(K)
55    CONTINUE
      WORK(K) = T
      M = IPVT(K)
      IF(M.EQ.K) GO TO 60
      T = WORK(M)
      WORK(M) = WORK(K)
      WORK(K) = T
60    CONTINUE
      YNORM = 0.0
      DO 65 I = 1,N
      YNORM = YNORM + DABS(WORK(I))
65    CONTINUE
      CALL SOLVE1(NDIM,N,A,WORK,IPVT)
      ZNORM = 0.0
      DO 70 I = 1,N
      ZNORM = ZNORM + DABS(WORK(I))
70    CONTINUE
      COND = ANORM*ZNORM/YNORM
      IF(COND.LT.1.0) COND = 1.0
      RETURN
80    COND = 1.0
      IF(A(1,1).NE.0.0) RETURN
90    COND = 1.0E+32
      RETURN
      END

      SUBROUTINE SOLVE1(NDIM,N,A,B,IPVT)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NDIM,N), B(N), IPVT(N)
      IF(N.EQ.1) GO TO 50
```

```
      NM1 = N-1
      DO 20 K = 1,NM1
      KP1 = K+1
      M = IPVT(K)
      T = B(M)
      B(M) = B(K)
      B(K) = T
      DO 10 I = KP1,N
      B(I) = B(I) + A(I,K)*T
10    CONTINUE
20    CONTINUE
      DO 40 KB = 1,NM1
      KM1 = N - KB
      K = KM1 + 1
      B(K) = B(K)/A(K,K)
      T = -B(K)
      DO 30 I = 1,KM1
      B(I) = B(I) + A(I,K)*T
30    CONTINUE
40    CONTINUE
50    B(1) = B(1)/A(1,1)
      RETURN
      END

      subroutine recda1(da,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)
C
C  this subroutine is used to recover the incremental dispacements
C  at node Q
C
C
C  hinge1(1,nhing),hinge1(2,nhing), and hinge1(3,nhing) give
C     three components of unit hinge axis vector HPQ.
C
C  hinge1(4,idhing) stores torsional spring constant around HPQ.
C  hinge1(5,idhing) stores relative rotation angle between P and
C     Q (around HPQ)
C
C  idofh1(nhing)  = 1: |HPQ1| = max{|HPQ1|,|HPQ2|,|HPQ3|}
C                 = 2: |HPQ2| = max{|HPQ1|,|HPQ2|,|HPQ3|}
C                 = 3: |HPQ3| = max{|HPQ1|,|HPQ2|,|HPQ3|}
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION mhing1(2,*),hing1(5,*),idofh1(*),NODBC(NTNODE,6)
      DIMENSION da(*),HPQ(3),SPQA(6,6),SPQC(6,6),daP(6),daQ(6),C(6)

      do nhing=1,nhing1
        IP=mhing1(1,nhing)
        IQ=mhing1(2,nhing)
        iching=idofh1(nhing)
        do i=1,3
          HPQ(i)=hing1(i,nhing)
        end do
        call HING1M(HPQ,SPQA,SPQC,ichin1)

        if(ichin1.ne.iching) then
          stop 'hinge calculation is NOT stable!!'
```

```
            end if

            do i = 1,6
              daP(i)=0.
              daQ(i)=0.
              C(i)=0.
            end do

            do i = 1,6
              idof=NODBC(IP,i)
              if(idof.ne.0) then
                daP(i)=da(idof)
              end if
            end do

            do i = 1,6
              idof=NODBC(IQ,i)
              if(idof.ne.0) then
                C(i)=da(idof)
              end if
            end do



            do i = 1,6
              wk1=0.
              wk2=0.
              do j = 1,6
                wk1=wk1 +SPQA(i,j)*daP(j)
              end do
              do j = 1,6
                wk2=wk2+SPQC(i,j)*C(j)
              end do

              daQ(i) = wk1 +wk2
            end do

            do i = 1,6
              idof=NODBC(IQ,i)
              if(idof.ne.0) then
                da(idof)=daQ(i)
              end if
            end do
c           write(6,*) 'SPQA=',SPQA
c           write(6,*) 'SPQC=',SPQC
c           write(6,*) 'daP=',daP
c           write(6,*) 'C=',C
c           write(6,*) 'daQ=',daQ
          end do

          return
          end
          SUBROUTINE HING1M(HPQ,SPQA,SPQC,iching)
          IMPLICIT REAL*8 (A-H,O-Z)
          DIMENSION HPQ(3),SPQA(6,6),SPQC(6,6),HPQW(3,3),HPQC(3,3)

          DO I = 1, 6
```

```
      DO J = 1, 6
        SPQA(I,J)= 0.
        SPQC(I,J)= 0.
      END DO
    END DO
    DO I = 1,3
     SPQA(I,I)= 1.

    END DO
    if(dabs(HPQ(1)).ge.dabs(HPQ(2)).and.
&      dabs(HPQ(1)).ge.dabs(HPQ(3))) then
       iching=1
       HPQW(1,1)=0.
       HPQW(1,2)=0.
       HPQW(1,3)=0.
       HPQW(2,1)=-HPQ(2)/HPQ(1)
       HPQW(2,2)=1.
       HPQW(2,3)=0.
       HPQW(3,1)=-HPQ(3)/HPQ(1)
       HPQW(3,2)=0.
       HPQW(3,3)=1.
       HPQC(1,1)=1.
       HPQC(1,2)=0.
       HPQC(1,3)=0.
       HPQC(2,1)=HPQ(2)/HPQ(1)
       HPQC(2,2)=0.
       HPQC(2,3)=0.
       HPQC(3,1)=HPQ(3)/HPQ(1)
       HPQC(3,2)=0.
       HPQC(3,3)=0.
    end if

    if(dabs(HPQ(2)).ge.dabs(HPQ(3)).and.
&      dabs(HPQ(2)).ge.dabs(HPQ(1))) then

       iching=2
       HPQW(1,1)=1.
       HPQW(1,2)=-HPQ(1)/HPQ(2)
       HPQW(1,3)=0.
       HPQW(2,1)=0.
       HPQW(2,2)=0.
       HPQW(2,3)=0.
       HPQW(3,1)=0.
       HPQW(3,2)=-HPQ(3)/HPQ(2)
       HPQW(3,3)=1.
       HPQC(1,1)=0.
       HPQC(1,2)=HPQ(1)/HPQ(2)
       HPQC(1,3)=0.
       HPQC(2,1)=0.
       HPQC(2,2)=1.
       HPQC(2,3)=0.
       HPQC(3,1)=0.
       HPQC(3,2)=HPQ(3)/HPQ(2)
       HPQC(3,3)=0.

    end if
```

```
         if(dabs(HPQ(3)).ge.dabs(HPQ(1)).and.
     &     dabs(HPQ(3)).ge.dabs(HPQ(2))) then

            iching=3
            HPQW(1,1)=1.
            HPQW(1,2)=0.
            HPQW(1,3)=-HPQ(1)/HPQ(3)
            HPQW(2,1)=0.
            HPQW(2,2)=1.
            HPQW(2,3)=-HPQ(2)/HPQ(3)
            HPQW(3,1)=0.
            HPQW(3,2)=0.
            HPQW(3,3)=0.
            HPQC(1,1)=0.
            HPQC(1,2)=0.
            HPQC(1,3)=HPQ(1)/HPQ(3)
            HPQC(2,1)=0.
            HPQC(2,2)=0.
            HPQC(2,3)=HPQ(2)/HPQ(3)
            HPQC(3,1)=0.
            HPQC(3,2)=0.
            HPQC(3,3)=1.

         end if

         DO I = 1, 3
           DO J = 1, 3
             SPQA(3+I,3+J) = HPQW(I,J)
             SPQC(3+I,3+J) = HPQC(I,J)
           END DO
         END DO


         return
         end

         subroutine ttthet(tt,theta)
c
c this subroutine is used to extract rotation vector "theta"
c from rotaion matrix "tt". notice that it is valid only in the
c case that the norm of theta is from zero to 2 pi.
c
c
         implicit real*8(a-h,o-z)
         dimension tt(3,3),ttt(3,3),tti(3,3),q(3),theta(3)

         am = -1000.
         imax = 0
         tr=0.

         do ii = 1,3
           tr = tr + tt(ii,ii)
           if(am.lt.tt(ii,ii)) then
             am= tt(ii,ii)
             imax = ii
           end if
         end do
```

```
          if(am.le.tr) then
            am=tr
          end if

          if(am.eq.tr) then
            q0 = 0.5*dsqrt(1. + tr)
            q(1) = (tt(3,2)-tt(2,3))/(4.*q0)
            q(2) = (tt(1,3)-tt(3,1))/(4.*q0)
            q(3) = (tt(2,1)-tt(1,2))/(4.*q0)
          end if

          if(am.ne.tr) then
            if(imax.eq.1) then
              q(1) = dsqrt(0.5*tt(1,1)+0.25*(1.-tr))
              q0 = (tt(3,2)-tt(2,3))/(4.*q(1))
              q(2) = (tt(2,1)+tt(1,2))/(4.*q(1))
              q(3) = (tt(3,1)+tt(1,3))/(4.*q(1))
            end if
            if(imax.eq.2) then
              q(2) = dsqrt(0.5*tt(2,2)+0.25*(1.-tr))
              q0 = (tt(1,3)-tt(3,1))/(4.*q(2))
              q(1) = (tt(1,2)+tt(2,1))/(4.*q(2))
              q(3) = (tt(3,2)+tt(2,3))/(4.*q(2))
            end if
            if(imax.eq.3) then
              q(3) = dsqrt(0.5*tt(3,3)+0.25*(1.-tr))
              q0 = (tt(2,1)-tt(1,2))/(4.*q(3))
              q(1) = (tt(1,3)+tt(3,1))/(4.*q(3))
              q(2) = (tt(2,3)+tt(3,2))/(4.*q(3))
            end if
          end if

            do i = 1,3
              do j = 1,3
                tti(i,j) = 0.d0
                do k = 1,3
                  tti(i,j) = tti(i,j) + tt(i,k)*ttt(k,j)
                end do
              end do
            end do

c          write(6,*) 'tti', tti
c          write(6,*) 'q =', q0,q

          sumq = q(1)*q(1)+q(2)*q(2)+q(3)*q(3)

          atheta= 2.*dasin(dsqrt(sumq))

          do ii = 1,3
            theta(ii) = 0.
          end do

          if(sumq.gt.1.d-15) then
            do ii = 1,3
              theta(ii) = atheta*q(ii)/dsqrt(sumq)
            end do
```

```
        end if

        call thettt(theta,ttt)

          do i = 1,3
            do j = 1,3
              tti(i,j) = 0.d0
              do k = 1,3
                tti(i,j) = tti(i,j) + tt(i,k)*ttt(k,j)
              end do
            end do
          end do

          anorm = 0.
          do i = 1,3
            anorm = anorm + tti(i,i)
          end do

          if(dabs(anorm-3.).lt.5.d-5) then
            do i = 1,3
              theta(i) = -theta(i)
            end do
          end if

        return
        end

        subroutine thettt(theta,tt)
c
c this subroutine is used to obtain rotation matrix "tt" from
c rotation vector "theta".
c
        implicit real*8(a-h,o-z)
        dimension tt(3,3),q(3),theta(3)

        atheta = dsqrt(theta(1)*theta(1)
     &               +theta(2)*theta(2)
     &               +theta(3)*theta(3))

        q0=dcos(atheta/2.)

        do ii = 1,3
         q(ii) = 0.
        end do
        if(atheta.gt.1.d-15) then
          do ii = 1,3
            q(ii) = (theta(ii)/atheta)*dsin(atheta/2.)
          end do
        end if

        tt(1,1) = 2.*(q0*q0+q(1)*q(1))-1.
        tt(2,2) = 2.*(q0*q0+q(2)*q(2))-1.
        tt(3,3) = 2.*(q0*q0+q(3)*q(3))-1.

        tt(1,2) = 2.*(q(1)*q(2)-q(3)*q0)
        tt(1,3) = 2.*(q(1)*q(3)+q(2)*q0)
```

```
        tt(2,1) = 2.*(q(2)*q(1)+q(3)*q0)
        tt(2,3) = 2.*(q(2)*q(3)-q(1)*q0)

        tt(3,1) = 2.*(q(3)*q(1)-q(2)*q0)
        tt(3,2) = 2.*(q(3)*q(2)+q(1)*q0)

        return
        end

c234567890123456789012345678901234567890123456789012345678901234567890123456789012
2
        subroutine
thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)
C
C  this subroutine is used to the relative rotation angle between
C  node P and node Q.
C
C
C  hinge1(1,nhing),hinge1(2,nhing), and hinge1(3,nhing) give
C     three components of unit hinge axis vector HPQ.
C
C  hinge1(4,idhing) stores torsional spring constant around HPQ.
C  hinge1(5,idhing) stores relative rotation angle between P and
C     Q (around HPQ)
C
C  idofh1(nhing)  = 1: |HPQ1| = max{|HPQ1|,|HPQ2|,|HPQ3|}
C                 = 2: |HPQ2| = max{|HPQ1|,|HPQ2|,|HPQ3|}
C                 = 3: |HPQ3| = max{|HPQ1|,|HPQ2|,|HPQ3|}
C
C
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION mhing1(2,*),hing1(5,*),idofh1(*),NODBC(NTNODE,6)
        DIMENSION atotl(*),thetaP(3),thetaQ(3),thetPQ(3)
        DIMENSION ttrP(3,3),ttrQ(3,3),ttrPQ(3,3)

        do nhing=1,nhing1
          IP=mhing1(1,nhing)
          IQ=mhing1(2,nhing)
          iching=idofh1(nhing)

          do i = 1,3
            thetaP(i)=0.
            thetaQ(i)=0.
          end do

          do i = 1,3
            IR=NODBC(IP,3+i)
            if(IR.ne.0) then
              thetaP(i)=atotl(IR)
            end if
          end do

          do i = 1,3
            IR=NODBC(IQ,3+i)
            if(IR.ne.0) then
              thetaQ(i)=atotl(IR)
            end if
```

```
          end do
c         write(6,*) 'thetP=',thetaP
c         write(6,*) 'thetQ=',thetaQ

          call thettt(thetaP,ttrP)
          call thettt(thetaQ,ttrQ)

c         write(6,*) 'ttrP=',ttrP
c         write(6,*) 'ttrQ=',ttrQ

          do i=1,3
            do j =1,3
              ttrPQ(i,j)=ttrQ(j,i)
            end do
          end do

          do i=1,3
            do j =1,3
              ttrQ(i,j)=ttrPQ(i,j)
            end do
          end do

          call matmat(ttrQ,ttrP,ttrPQ)
c         write(6,*) 'ttrPQ=',ttrPQ
          call ttthet(ttrPQ,thetPQ)
c         write(6,*) 'thetPQ=',thetPQ

          hing1(5,nhing) =thetPQ(1)*hing1(1,nhing)+
     &                            thetPQ(2)*hing1(2,nhing)+
     &                            thetPQ(3)*hing1(3,nhing)

        end do

        return
        end
```

```
C analy3.f:
C
        SUBROUTINE LDSTEP(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)

        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION LM(*),MAXH(*),MHT(*)
        DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(*)
        DIMENSION mhing1(2,*),mhing3(2,*)
        DIMENSION hing1(5,*),idofh1(*)
        DIMENSION CNCM(6,4,nelm)
        DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
        DIMENSION qext0(*),qint(*)
        DIMENSION da(*),dq(*),da0(*),dq0(*)
        DIMENSION sda(*),sdq(*),sxnode(*)
        DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
        DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
        DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
        DIMENSION atotl(*),atotl0(*)

        Npr=1
        ldown=0
        niter=0

        IF(NLD.EQ.maxnld) then
           STOP 'MAXIMUM NUMBER OF LOAD STEPS EXCEEDED'
        END IF

        IF(alambd.GT.toltld) then
           stop 'MAXIMUM LOAD FACTOR EXCEEDED'
        END IF
        alinc=alambd-alamb0

        Npr=Npr+1
        NLD=NLD+1

18      continue

c       write(6,*) '   '
c       write(6,*) 'load level NLD=', nld, '  alambd =',alambd
c       write(6,*) '   '
```

```
        CALL CLVEC(AST,NWK)
        CALL CLVEC(dq0,MAX)

        MAXEQN=MAX-1

C
C   ASSEMBLY THE TANGENT STIFFNESS MATRIX: AST
C
C
        CALL CLVEC(AST,NWK)
        CALL CLVEC(dq0,MAX)

        LCODE = 1
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)

        DO IASSIGN = 1, NWK
          AST1(IASSIGN)=AST(IASSIGN)
        END DO
C
C   ASSEMBLY THE INTITIAL UNBALANCED LOAD,
C               AT CURRENT LOAD LEVEL
C
        CALL CLVEC(dq,MAX)
        LCODE = 2
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)
        CALL RESIDL(qext0,qint,maxeqn,alambd,dq)

c       write(6,*) 'intitial unbalanced LOAD'
c       write(6,*) 'dq=',(dq(iii),iii=1,maxeqn)

        if(nld.eq.1) then
          DO ING=1,MAXEQN
            dq0(ING)=dq(ING)  ! get the  intitial unbalanced LOAD
          END DO
        end if

C
C   DECOMPOSITION OF TANGENT STIFFNESS MATRIX
C
        CALL DECOMP1(MAXEQN,MAXEQN,AST,COND,IPVT,NCHG,WORK)
C       CALL LUDCMP(AST,MAXEQN,MAXEQN,IPVT,D)
C       CALL COLSOL(AST,da,MAXH,MAXEQN,NWK,MAX,1)
```

```
64      CONTINUE

        DO ING=1,MAXEQN
          da(ING)=dq(ING)  ! get the unbalanced load
        END DO

        ITR=1     ! number of iterations inside the incremental load
loop

C-----SOLVE THE MATRIX EQUATIONS TO GET DISPLACEMENTS
C     DUE TO INTITIAL UNBALANCED LOAD

c       CALL LUBKSB(AST,MAXEQN,MAXEQN,IPVT,da)
        CALL SOLVE1(MAXEQN,MAXEQN,AST,da,IPVT)
C       CALL COLSOL(AST,da,MAXH,MAXEQN,NWK,MAX,2)
C
C     RECOVER daQ
C
        call recda1(da,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)

        if(nld.eq.1) then
          DO ING=1,MAXEQN
            da0(ING)=da(ING) !store the displ due to intitial
unbalanced LOAD
          END DO
          CALL VECVEC(dq0,dq0,dq0dq0,MAXEQN)  ! inner product dq0.dq0
          CALL VECVEC(da0,da0,da0da0,MAXEQN)  ! inner product  da0.da0
        end if

24      CONTINUE        ! iterations inside a load step start here

        ITR=ITR+1
        niter=niter+1
C       IF(ITR.GT.76) GO TO 28

C
C     UPDATE xnodel
C
        CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,da)
        CALL upatot(ntnode,NODBC,atotl,da)
        CALL thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)
C
C     UPDATE ttrg & omegag
C
        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,ttrg,omegag)
        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,t6rg,om6gag)
C
C   ASSEMBLY THE UPDATED TANGENT STIFFNESS MATRIX: AST
C
C
        CALL CLVEC(AST,NWK)
        LCODE = 1
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
```

```
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)
        DO IASSIGN = 1, NWK
          AST1(IASSIGN)=AST(IASSIGN)
        END DO

C
C   DECOMPOSITION OF TANGENT STIFFNESS MATRIX
C
        CALL DECOMP1(MAXEQN,MAXEQN,AST,COND,IPVT,NCHG,WORK)
C       CALL LUDCMP(AST,MAXEQN,MAXEQN,IPVT,D)
C       CALL COLSOL(AST,da,MAXH,MAXEQN,NWK,MAX,1)

C
C   ASSEMBLY THE unbalanced LOAD for updated configuration
C               at current load level
C
        CALL CLVEC(dq,MAX)
        LCODE = 2
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)
        CALL RESIDL(qext0,qint,maxeqn,alambd,dq)


        CALL VECVEC(dq,dq,dqdq,MAXEQN)  ! inner product dq.dq

c
c     store current unbalanced force into da temporarily.
c

        do ing = 1,maxeqn
          da(ing) = dq(ing)
        end do

c
c-----SOLVE THE MATRIX EQUATIONS TO GET DISPLACEMENTS under unbalanced
force
c
c       CALL LUBKSB(AST,MAXEQN,MAXEQN,IPVT,da)
        CALL SOLVE1(MAXEQN,MAXEQN,AST,da,IPVT)
c       CALL COLSOL(AST,da,MAXH,MAXEQN,NWK,MAX,2)
C
C     RECOVER daQ
C
        call recda1(da,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)

        CALL VECVEC(da,da,dada,MAXEQN)  ! inner product  da.da
```

```
        dq0dq0=1.d0
        da0da0=1.d0

        RATIOL=DSQRT(dqdq/dq0dq0)
        RATIOD=DSQRT(dada/da0da0)
        RATLD=DSQRT(dqdq/dada)
        WRITE (16,*) itr,RATIOL
C       WRITE (6,*) 'RATLD=',RATLD

C
C   CONVERGENCE CHECKING FOR LOAD RESIDUAL
C

        IF(RATIOL.LE.toldq) GO TO 28    !CONVERGENCE

C
C   CONVERGENCE CHECKING FOR DISPLACEMENT INCREMENT
C
        IF(RATIOD.LE.tolda) GO TO 28    !CONVERGENCE

C
C   CONVERGENCE CHECKING FOR LOAD RESIDUAL AND DISPLACEMENT INCREMENT
C
C       IF(RATIOD.LE.toldadq.AND.RATIOL.LE.toldadq) GO TO 28
!CONVERGENCE

C
C   CHECK MAXIMUM NUMBER OF ITERATIONS INSIDE OF CURRENT LOAD LOOP
C

        IF(ITR.GT.MAXITR) GO TO 66           ! EXCEED


        GO TO 24                          ! GO TO ANOTHER ITR LOOP

28      CONTINUE                          ! GET HERE IF CONVERGENCE
ACHIEVES.

        WRITE(6,*) 'The number of iteration itr =',itr
        WRITE(6,*) 'dada=',dada, 'dqdq=',dqdq
        WRITE (6,20003) RATIOD,RATIOL
20003   FORMAT(//2X,'RATIOS: RATIOD =',1PE20.10,'RATIOL =',1PE20.10)
c        WRITE (6,*) 'RATLD=',RATLD

        WRITE(10,*) '    '
        WRITE(10,*) '============================================='
        write(10,*) 'load level NLD=', nld, '     alambd=',alambd
C
C
C
        CALL DIADET(AST,IPVT,DIAG,MAXEQN,NCHG,NNEGA,NNEGD,DET)
        idet = IPVT(MAXEQN)*(-1)**NNEGA
        if(nld.eq.1) then
          det0=det
        else if (nld.gt.1) then
          rdet=dfloat(idet)*dexp(det-det0)
```

```
          write(10,*) 'Relative Det =', rdet
        end if


        call tempsv(maxeqn,nelm,
     &              xnodel,ttrg,omegag,da,dq,
     &              sxnode,sttrg,somegg,sda,sdq)

        call tempsv(maxeqn,nelm,
     &              xnodel,t6rg,om6gag,da,dq,
     &              sxnode,st6rg,som6gg,sda,sdq)

        return        ! go to new load loop
C
C
C
66      CONTINUE
C
C     CONVERGENCE NOT ACHIEVED. REDUCE THE LOAD STEP BY HALF.
C
        write(6,*) 'Load increment is reduced by half'


        ldown=ldown+1
        ALINC=ALINC*0.5D0
        alambd=alamb0+ALINC

        write(6,*) 'Load level is reduced to alambd= ', alambd

        call tempgt(maxeqn,nelm,
     &             xnodel,ttrg,omegag,da,dq,
     &             sxnode,sttrg,somegg,sda,sdq)
        call tempgt(maxeqn,nelm,
     &             xnodel,t6rg,om6gag,da,dq,
     &             sxnode,st6rg,som6gg,sda,sdq)

        IF(ldown.GT.maxldn) then
          return
        end if

        GO TO 18

        END
```

```
C  analy4.f
C
C
c   this subroutine is used to locate bifurcation point by bisection
method
c
c   when the relative determinant of tangential stiffness matrix rdet
changes
c   sign (from positive to negative). this method will fail to locate
the exact
c   bifurcation pointif the noise is large.
c
c
c

      SUBROUTINE BIFUR2(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                  LM,MAXH,MHT,
     &                  X0,XYZN,xnodel,melm,NODBC,
     &                  nhing1,nhing3,
     &                  mhing1,mhing3,
     &                  hing1,idofh1,
     &                  cncm,
     &                  ptload,distgg,distnn,distpp,
     &                  qext0,qint,da,dq,da0,dq0,
     &                  sda,sdq,sxnode,
     &                  ttrg,t6rg,omegag,om6gag,
     &                  sttrg,st6rg,somegg,som6gg,
     &                  AST,AST1,diag,IPVT,WORK,
     &                  atotl,atotl0,
     &                  maxnld,maxldn,maxitr,
     &                  NLD,ldown,niter,
     &                  toltld,tolda,toldq,toldadq,
     &                  alambd,alamb0,det0,rdet,
     &                  aaa,bbb,faaa,fbbb)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(1),Y(1),S(1)
      DIMENSION LM(*),MAXH(*),MHT(*)
      DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(*)
      DIMENSION mhing1(2,*),mhing3(2,*)
      DIMENSION hing1(5,*),idofh1(*)
      DIMENSION CNCM(6,4,*)
      DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
      DIMENSION qext0(*),qint(*)
      DIMENSION da(*),dq(*),da0(*),dq0(*)
      DIMENSION sda(*),sdq(*),sxnode(*)
      DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
      DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
      DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
      DIMENSION atotl(*),atotl0(*)

      do ii = 1,50
        ccc=0.5*(aaa+bbb)
        X(1)=ccc
        CALL OBJ(X,F,
     &                  NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                  LM,MAXH,MHT,
```

```
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)
          ccc=alambd
          fccc=rdet

          if(faaa*fccc.lt.0.) then
            bbb=ccc
            fbbb=fccc
          else
            aaa=ccc
            faaa=fccc
          end if
          if(dabs(bbb-aaa)/dabs(ccc).le.1.d-5
     &       .and.dabs(fccc).le.1.d-5)then
            return
          end if
        end do

        return
        end

c
c   this subroutine is used to locate bifurcation point by bisection
method
c
c   when the relative determinant of tangential stiffness matrix rdet
starts
c   to increase after it decreases and when rdet is small enough (to
make sure
c   that there is a point inside at which rdet = 0. it is
c   especially useful when rdet >= 0. when rdet is not smooth enough,
it may
c   fail to locate the exact bifurcation point.
c
c   if rdet changes sign from positive to negative, it is better use
bisection
c   method.
c
c   it is not used currently as there is some bugs inside.
c
c
c
        SUBROUTINE BISEC2(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
```

```
&                     LM,MAXH,MHT,
&                     X0,XYZN,xnodel,melm,NODBC,
&                     nhing1,nhing3,
&                     mhing1,mhing3,
&                     hing1,idofh1,
&                     cncm,
&                     ptload,distgg,distnn,distpp,
&                     qext0,qint,da,dq,da0,dq0,
&                     sda,sdq,sxnode,
&                     ttrg,t6rg,omegag,om6gag,
&                     sttrg,st6rg,somegg,som6gg,
&                     AST,AST1,diag,IPVT,WORK,
&                     atotl,atotl0,
&                     maxnld,maxldn,maxitr,
&                     NLD,ldown,niter,
&                     toltld,tolda,toldq,toldadq,
&                     alambd,alamb0,det0,rdet,
&                     aaa,bbb,ccc,faaa,fbbb,fccc)

   IMPLICIT REAL*8(A-H,O-Z)
   DIMENSION X(1),Y(1),S(1)
   DIMENSION LM(*),MAXH(*),MHT(*)
   DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(*)
   DIMENSION mhing1(2,*),mhing3(2,*)
   DIMENSION hing1(5,*),idofh1(*)
   DIMENSION CNCM(6,4,*)
   DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
   DIMENSION qext0(*),qint(*)
   DIMENSION da(*),dq(*),da0(*),dq0(*)
   DIMENSION sda(*),sdq(*),sxnode(*)
   DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
   DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
   DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
   DIMENSION atotl(*),atotl0(*)

   do ii = 1,50

     aaa1=0.5*(aaa+ccc)
      X(1)=aaa1
     CALL OBJ(X,faaa1,
&                     NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                     LM,MAXH,MHT,
&                     X0,XYZN,xnodel,melm,NODBC,
&                     nhing1,nhing3,
&                     mhing1,mhing3,
&                     hing1,idofh1,
&                     cncm,
&                     ptload,distgg,distnn,distpp,
&                     qext0,qint,da,dq,da0,dq0,
&                     sda,sdq,sxnode,
&                     ttrg,t6rg,omegag,om6gag,
&                     sttrg,st6rg,somegg,som6gg,
&                     AST,AST1,diag,IPVT,WORK,
&                     atotl,atotl0,
&                     maxnld,maxldn,maxitr,
&                     NLD,ldown,niter,
&                     toltld,tolda,toldq,toldadq,
```

```
&                    alambd,alamb0,det0,rdet)

      if(faaa1.ge.fccc) then
        aaa = aaa1
        faaa = faaa1

        bbb1 = 0.5*(ccc+bbb)
        X(1)=bbb1
        CALL OBJ(X,fbbb1,
&                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,
&                    maxnld,maxldn,maxitr,
&                    NLD,ldown,niter,
&                    toltld,tolda,toldq,toldadq,
&                    alambd,alamb0,det0,rdet)
         bbb1 =  alambd
        if(fbbb1.ge.fccc) then
          bbb = bbb1
          fbbb = fbbb1
        else if(fbbb1.lt.fccc) then
          ccc = bbb1
          fccc = fbbb1
        end if
      else if(faaa1.lt.fccc) then
        ccc = aaa1
        fccc = faaa1

        bbb1 = 0.5*(ccc+bbb)
        X(1)=bbb1
        CALL OBJ(X,fbbb1,
&                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,
&                    maxnld,maxldn,maxitr,
```

```
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)
            bbb1 =  alambd
            if(fbbb1.ge.fccc) then
              bbb = bbb1
              fbbb = fbbb1
            else if(fbbb1.lt.fccc) then
              ccc = bbb1
              fccc = fbbb1
            end if
          end if

          if(dabs(bbb-aaa)/dabs(ccc).le.1.d-5
     &       .or.dabs(fccc).le.1.d-5)then
            return
          end if
        end do

        return
        end
c
c
c   this subroutine is used to locate bifurcation point by 3 points
method
c
c   when the relative determinant of tangential stiffness matrix rdet
starts
c   to increase after it decreases and when rdet is small enough (to
make sure
c   that there is a point inside at which rdet = 0. it is
c   especially useful when rdet >= 0. when rdet is not smooth enough,
it may
c   fail to locate the exact bifurcation point.
c
c   if rdet changes sign from positive to negative, it is better use
bisection
c   method.
c
c
c
c
        SUBROUTINE BIFUR3(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
```

```
&                    NLD,ldown,niter,
&                    toltld,tolda,toldq,toldadq,
&                    alambd,alamb0,det0,rdet)
    IMPLICIT REAL*8(A-H,O-Z)
    DIMENSION X(1),Y(1),S(1)
    DIMENSION LM(*),MAXH(*),MHT(*)
    DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(*)
    DIMENSION mhing1(2,*),mhing3(2,*)
    DIMENSION hing1(5,*),idofh1(*)
    DIMENSION CNCM(6,4,*)
    DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
    DIMENSION qext0(*),qint(*)
    DIMENSION da(*),dq(*),da0(*),dq0(*)
    DIMENSION sda(*),sdq(*),sxnode(*)
    DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
    DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
    DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
    DIMENSION atotl(*),atotl0(*)

    X(1) = alambd
    FX=1.
    S(1) = alambd-alamb0
    ITERR = 0

    CALL DSCPOW(X,FX,Y,FY,S,S,1,2,0.00001,
&                    1,ITERR,IEX,
&                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,
&                    maxnld,maxldn,maxitr,
&                    NLD,ldown,niter,
&                    toltld,tolda,toldq,toldadq,
&                    alambd,alamb0,det0,rdet)


    return
    end

    SUBROUTINE OBJ(X,FX,
&                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
```

```
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,
&                    maxnld,maxldn,maxitr,
&                    NLD,ldown,niter,
&                    toltld,tolda,toldq,toldadq,
&                    alambd,alamb0,det0,rdet)

   IMPLICIT REAL*8(A-H,O-Z)
   DIMENSION X(1)
   DIMENSION LM(*),MAXH(*),MHT(*)
   DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(*)
   DIMENSION mhing1(2,*),mhing3(2,*)
   DIMENSION hing1(5,*),idofh1(*)
   DIMENSION CNCM(6,4,*)
   DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
   DIMENSION qext0(*),qint(*)
   DIMENSION da(*),dq(*),da0(*),dq0(*)
   DIMENSION sda(*),sdq(*),sxnode(*)
   DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
   DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
   DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
   DIMENSION atotl(*),atotl0(*)

   alambd= X(1)

   CALL LDSTEP(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                    LM,MAXH,MHT,
&                    X0,XYZN,xnodel,melm,NODBC,
&                    nhing1,nhing3,
&                    mhing1,mhing3,
&                    hing1,idofh1,
&                    cncm,
&                    ptload,distgg,distnn,distpp,
&                    qext0,qint,da,dq,da0,dq0,
&                    sda,sdq,sxnode,
&                    ttrg,t6rg,omegag,om6gag,
&                    sttrg,st6rg,somegg,som6gg,
&                    AST,AST1,diag,IPVT,WORK,
&                    atotl,atotl0,
&                    maxnld,maxldn,maxitr,
&                    NLD,ldown,niter,
&                    toltld,tolda,toldq,toldadq,
&                    alambd,alamb0,det0,rdet)

   X(1)=alambd
   FX=dsqrt(dsqrt(rdet*rdet))

   RETURN
   END
   SUBROUTINE DSCPOW(X,FX,Y,FY,S,DELX,N,INDIC,TOL,
&                    ITOL,ITERR,IEX,
&                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
```

```
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)

        IMPLICIT REAL*8(A-H,O-Z)

        DIMENSION X(N),Y(N),S(N),DELX(N)

        DIMENSION LM(*),MAXH(*),MHT(*)
        DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(*)
        DIMENSION mhing1(2,*),mhing3(2,*)
        DIMENSION hing1(5,*),idofh1(*)
        DIMENSION CNCM(4,6,*)
        DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
        DIMENSION qext0(*),qint(*)
        DIMENSION da(*),dq(*),da0(*),dq0(*)
        DIMENSION sda(*),sdq(*),sxnode(*)
        DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
        DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
        DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
        DIMENSION atotl(*),atotl0(*)

        IF(ITOL.EQ.1) GO TO 110
        TOL = .001
110     FTOL2 = TOL/1000.
        NTOL = 0
        IEX = 1
        K = -2
        M = 0
        FA = FX
        DAA = 0.
        STEP = 1.
        D = STEP
        IF(INDIC.EQ.2) GO TO 1
        IF(ITERR.EQ.0) GO TO 1
        DXNORM = 0.
        SNORM = 0.
        DO 102 I = 1,N
        DXNORM = DXNORM + DELX(I)*DELX(I)
102     SNORM = SNORM + S(I)*S(I)
        IF(INDIC.NE.1) GO TO 103
        IF(DXNORM.GE.SNORM) GO TO 1
103     RATIO = DXNORM/SNORM
```

```
        STEP = DSQRT(RATIO)
        D = STEP
 1      DO 2 I =1,N
 2      Y(I) = X(I) + D*S(I)
        CALL OBJ(Y,F,
     &                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)
c       if(dabs(f).lt.ftol2) return
        K = K + 1
        IF(F-FA) 5,3,6
 3      D = 0.5*(DAA+D)
        DO 4 I = 1,N
 4      Y(I) = X(I) + D*S(I)
        CALL OBJ(Y,F,
     &                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)
c       if(dabs(f).lt.ftol2) return
        IF(F-FA) 204,202,205
 202    DO 203 I = 1,N
 203    Y(I) = X(I) + DAA*S(I)
        FY = FA
        GO TO 326
 204    FC = FA
        DC = DAA + 2.*(D-DAA)
        FB = F
```

```
        DB = D
        GO TO 21
205     STEP = 0.5*STEP
        IF(K) 7,8,206
206     DC = D
        D = DAA
        DAA = DB
        DB = D
        FC = F
        F = FA
        FA = FB
        FB = F
        GO TO 10
5       FC = FB
        FB = FA
        FA = F
        DC = DB
        DB = DAA
        DAA = D
        D = 2.*D + STEP
        GO TO 1
6       IF(K)7,8,9
7       FB = F
        DB = D
        D = -D
        STEP = -STEP
        GO TO 1
8       FC = FB
        FB = FA
        FA = F
        DC = DB
        DB = DAA
        DAA = D
        GO TO 21
9       DC = DB
        DB = DAA
        DAA = D
        FC = FB
        FB = FA
        FA = F
10      D = 0.5*(DAA+DB)
        DO 11 I = 1,N
11      Y(I) = X(I) + D*S(I)
        CALL OBJ(Y,F,
     &                     NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                     LM,MAXH,MHT,
     &                     X0,XYZN,xnodel,melm,NODBC,
     &                     nhing1,nhing3,
     &                     mhing1,mhing3,
     &                     hing1,idofh1,
     &                     cncm,
     &                     ptload,distgg,distnn,distpp,
     &                     qext0,qint,da,dq,da0,dq0,
     &                     sda,sdq,sxnode,
     &                     ttrg,t6rg,omegag,om6gag,
     &                     sttrg,st6rg,somegg,som6gg,
     &                     AST,AST1,diag,IPVT,WORK,
```

```
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)

c        if(dabs(f).lt.ftol2) return
 12     IF((DC-D)*(D-DB)) 15,13,18
 13     DO 14 I = 1,N
 14     Y(I) =X(I) + DB*S(I)
        FY = FB
        IF(IEX.EQ.0) GO TO 32
        GO TO 326
 15     IF(F-FB) 16,13,17
 16     FC = FB
        FB = F
        DC = DB
        DB = D
        GO TO 21
 17     FA = F
        DAA = D
        GO TO 21
 18     IF(F-FB) 19,13,20
 19     FA = FB
        FB = F
        DAA = DB
        DB = D
        GO TO 21
 20     FC = F
        DC = D
 21     IF(DABS(DAA-DB).GE.DABS(DC-DB)) GO TO 50
        D = DAA
        DAA = DC
        DC = D
        F = FA
        FA = FC
        FC = F
 50     IF(DABS((DC-DB)/(DAA-DB)).GT.0.25) GO TO 26
        D = 0.5*(DAA+DB)
        DO 51 I = 1,N
 51     Y(I) = X(I) + D*S(I)
        CALL OBJ(Y,F,
     &                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
```

```
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)
c       if(dabs(f).lt.ftol2) return
        IF(F-FB) 52,13,53
 52     FC = FB
        FB = F
        DC = DB
        DB = D
        GO TO 26
 53     FA = F
        DAA = D
 26     A = FA*(DB-DC) + FB*(DC-DAA) + FC*(DAA-DB)
        IF(A) 22,30,22
 22     D = 0.5*((DB*DB-DC*DC)*FA + (DC*DC-DAA*DAA)*FB +
     &          (DAA*DAA-DB*DB)*FC)/A
        IF((DAA-D)*(D-DC)) 13,13,23
 23     DO 24 I = 1,N
 24     Y(I) = X(I) + D*S(I)
        CALL OBJ(Y,F,
     &                    NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)
c       if(dabs(f).lt.ftol2) return
        IF(DABS(FB-F)-(DABS(FB)+FTOL2)*TOL) 28,28,12
 28     IEX = 0
        IF(F-FB) 29,13,13
 29     FY = F
        GO TO 32
 30     IF(M) 31,31,13
 31     M = M +1
        GO TO 10
 32     DO 99 I = 1,N
        IF(Y(I).NE.X(I)) GO TO 326
 99     CONTINUE
 33     IF(NTOL.EQ.5) GO TO 34
        IEX = 1
        NTOL = NTOL + 1
        TOL = TOL/10.
        GO TO 12
 34     IEX = 1
        GO TO 35
```

```
 326    IF(FY.LT.FX) GO TO 36
        IEX = 1
        GO TO 35
 36     IF(IEX.EQ.0) GO TO 35
        IEX = 2
 35     RETURN
        END
c
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccc
c
c
c   this subroutine is used to perform linearized bifurcation buckling
c   analysis by solving eigenproblem:
c
c                (Kl + lambda Knl) u = 0              (1)
c
c   According to the theory  of stabilty, a bifurcation point
corresponds
c   to the condition that the first variation of the virtual work
vanishes:
c
c                        ddW = 0                      (2)
c
c   or
c
c                      det(Kt) = 0                    (3)
c
c   or
c                       Kt u = 0                      (4)
c
c   or
c
c         (Kt + dlambda I) u = 0   with dlambda = 0   (5)
c
c   where Kt is the tangential stiffness matrix at load level lambda
c
c
c   assume that we start from a reference load level lambda0 and
increase
c   load by dlambda to lambda:
c
c               lambda = lambda0 + dlambda            (6)
c
c   now we expand Kt using Talor series and negelect higher order of
terms:
c
c             Kt = Kt(lambda)
c                = Kt(lambda0 + dlambda)
c                = Kt(lambda0) + [dKt(lambda0)/dlambda] dlambda
c   or
c             Kt = Kl + dlambda Knl                   (7)
c
c   where
c                  Kl = Kt(lambda0)                  (7a)
```

```
c              Knl = dKt(lambda0)/dlambda             (7b)
c
c   using eqns (7) and (4), we obtain eigen problem:
c
c             (Kl + dlambda Knl) u = 0                (8)
c
c   considering the case
c
c                lambda0 = 0                          (9)
c
c   which means that
c
c             lambda = dlambda                        (10)
c
c   we obtain  linearized bifurcation buckling problem (1)
c
c
c   now we examine the Kt in detail. we can seperate Kt as
c
c             Kt = Km + Kg + Kp                       (11)
c
c   where
c
c             Km is the material part of Kt
c
c             Kg is the geometric part of Kt
c
c             Kp is the pressure part of Kt
c
c
c   Km is a function of material properties and rotation matrix Lambda
c
c              Km = Km(Cn,Cm,Lambda)                   (12)
c
c   Kg is a function of stress resultants and couples
c
c              Kg = Kg(n,m)                            (13)
c
c   Kp is a function of load factor lambda and rotation matrix Lambda
c
c              Kp = Kp(lambda, Lambda)                 (14)
c
c   Simply pushing it, Km, Kg, and Kp all are functon of load factor
lambda.
c   so we can linearize them as before:
c
c              Km  = Kml + dlambda Kmnl                (15)
c   with
c                  Kml = Km(lambda0)                   (15a)
c              Kmnl = dKm(lambda0)/dlambda             (15b)
c
c              Kg = Kgl + dlambda Kgnl                 (16)
c   with
c                  Kgl = Kg(lambda0)                   (16a)
c              Kgnl = dKg(lambda0)/dlambda             (16b)
c
c   and
```

```
c               Kp = Kpl + dlambda Kpnl                (17)
c   with
c                  Kpl = Kp(lambda0)                   (17a)
c              Kpnl = dKp(lambda0)/dlambda             (17b)
c
c
c   now substitute eqns (15-17) into eqn (11), we get:
c
c       Kt = (Kml + Kgl + Kpl) + dlambda (Kmnl + Kgnl + Kpnl)
c                                                      (18)
c
c   by comparision of eqn (18) with eqn (7), we obtain:
c
c                Kl = Kml + Kgl + Kpl                  (18a)
c
c              Knl = Kmnl + Kgnl + Kpnl                (18b)
c
c  assume that lambda0 = 0, (i.e., lambda = dlambda) and that there is
c  no pre-stresses, it is obvious that
c
c                 Kgl = 0 and  Kpl= 0
c
c  but usually  Kmnl =\= 0.
c
c  therefore, eqn (18a) can be rewritten as
c
c                       Kl = Kml                       (18c)
c
c  in classical theory, however, the assumption is made of use that
either
c  rotation is zero or neglected, hence
c
c                       Kmnl = 0                       (18d)
c
c
c  therefore, eqn (18b) can be rewritten as
c
c                    Knl = Kgnl + Kpnl                 (18e)
c
c  furthermore, if there is no pressure load, i.e.,
c
c                         Kpnl = 0                     (18f)
c  we have
c                        Knl = Kgnl                    (18g)
c
c-----------------------------------------------------------------------
---
c
c  now we consider how to calculate Kl and Knl.
c
c  in principle, we can linearize Kt as eqn (7) by the assumption that
c  rotation is small. therefore, rotation matix is simplified to
c
c                Lambda = Lambda(theta)
c
c                       = I + skew(theta)               (19)
c
```

```
c   substitute into the exact formula for Km, we get
c
c               Km = Kml(0) + Kmnlt.theta             (20)
c
c   in which  Kmnlt is independent of theta and therefore indepent of
load
c   factor lambda while rotation vector theta is proportational to
lambda.
c   if we represent theta0 correspends to reference load, then
c
c                Kmnlt.theta  = lambda Kmnlt.(theta0)
c                             = lambda Kmnl            (21)
c   where
c                     Kmnl = Kmnlt.theta0             (21a)
c
c   for classical solution, we can simply set Kmnl = 0.
c
c
c   similarly, using linearized formular (19), we can get linearized
formulae
c   for stress resultants and couples:
c
c               n = lambda n_nlt.theta_0 = lambda n_nlt (22)
c
c               m = lambda m_nlt.theta_0 = lambda m_nlt (23)
c
c   where
c                     n_nl = n_nlt.theta_0              (22a)
c
c                     m_nl = m_nlt.theta_0              (23a)
c
c   which are linearized  stress resultants and couples associated
c   with reference load.
c
c   substitute eqns (22a) and (23a) into the formula for Kg, we can get
c   Kgnl.
c
c
c   the same to Kpnl.
c
c
c   as it is messy to get Knl via above approach, here quasi forward
finite
c   difference method is used to get Knl.
c
c
c
c ------------------------------------------------------------------
---
c
c
c    two different approaches to choose Kl and Knl:
c
c    1:  Kl is the tangential stiffness matrix at lambda equals to
zero.
c
```

```
c        Knl is geometric and pressure parts of tangential stiffness
matrix
c        out of first iteration of the first load step. this approach
is
c        classical approach in the sense that rotation is assumed to be
zero
c        or neglible.
c
c        Knl = [Kg(lambda_eps)+Kp(lambda_eps)]/lambda_eps
c
c        lambda_eps is set to 0.0001
c
c
c    2:  Kl is the tangential stiffness matrix at lambda equals to
zero.
c
c        Knl includes all terms of material, geometric and pressure
parts of
c        tangential stiffness matrix out of first iteration of the
first load
c        step.
c
c        Knl = [Km(lambda_eps)+Kg(lambda_eps)+Kp(lambda_eps)-
Kl]/lambda_eps
c
c
c       lambda_eps is set to 0.001
c
c
c
c
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccc


        SUBROUTINE BIFURL(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    wr,wi,z,loc,fv1,iv1,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet,ip2)
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION LM(*),MAXH(*),MHT(*)
      DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(*)
      DIMENSION mhing1(2,*),mhing3(2,*)
      DIMENSION hing1(5,*),idofh1(*)
      DIMENSION CNCM(6,4,*)
      DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
      DIMENSION qext0(*),qint(*)
      DIMENSION da(*),dq(*),da0(*),dq0(*)
      DIMENSION sda(*),sdq(*),sxnode(*)
      DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
      DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
      DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
      DIMENSION atotl(*),atotl0(*)
      DIMENSION WR(*),WI(*),Z(MAXEQN,*),LOC(*)
      DIMENSION FV1(*),IV1(*)
      MAXEQN=MAX-1

      CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
C
C   ASSEMBLY THE TANGENT STIFFNESS MATRIX: AST
C
C
      CALL CLVEC(AST,NWK)

      if(ip2.eq.1) then
        almde = 0.000001
      end if
      if(ip2.eq.2) then
        almde = 0.000001
      end if

      LCODE = 1
      CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                   nhing1,nhing3,
     &                   mhing1,mhing3,
     &                   hing1,idofh1,
     &                   distgg,distnn,distpp,ptload,alambd,
     &                   ttrg,t6rg,omegag,om6gag,da,
     &                   LM,MAXH,MHT,NWK,MK,
     &                   AST,qint,qext0,LCODE)

C
C   ASSEMBLY THE INTITIAL UNBALANCED LOAD,
C               AT CURRENT LOAD LEVEL
C
      CALL CLVEC(dq,MAX)
      LCODE = 2
      CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                   nhing1,nhing3,
     &                   mhing1,mhing3,
     &                   hing1,idofh1,
     &                   distgg,distnn,distpp,ptload,alambd,
     &                   ttrg,t6rg,omegag,om6gag,da,
     &                   LM,MAXH,MHT,NWK,MK,
     &                   AST,qint,qext0,LCODE)
```

```
        write(6,*) 'external load'
        qq=0.
        do  ii=1, maxeqn
          qq=qq+qext0(ii)
        end do

        write(6,*) qq

        alambd=almde
        CALL RESIDL(qext0,qint,maxeqn,alambd,dq)

        do ii = 1,nwk

            AST1(ii) = AST(ii)

        end do

C
C   DECOMPOSITION OF TANGENT STIFFNESS MATRIX
C
        CALL DECOMP1(MAXEQN,MAXEQN,AST,COND,IPVT,NCHG,WORK)

        DO ING=1,MAXEQN
          da(ING)=dq(ING)  ! get the unbalanced load
        END DO


C-----SOLVE THE MATRIX EQUATIONS TO GET DISPLACEMENTS
C     DUE TO INTITIAL UNBALANCED LOAD

        CALL SOLVE1(MAXEQN,MAXEQN,AST,da,IPVT)

C
C     RECOVER daQ
C
        call recda1(da,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
C
C   ASSEMBLY THE TANGENT STIFFNESS MATRIX: AST
C
C
c        CALL CLVEC(AST,NWK)


c       LCODE = 1
c        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
c     &                     nhing1,nhing3,
c     &                     mhing1,mhing3,
c     &                     hing1,idofh1,
c     &                    distgg,distnn,distpp,ptload,alambd,
c     &                     ttrg,t6rg,omegag,om6gag,da,
c     &                     LM,MAXH,MHT,NWK,MK,
c     &                     AST,qint,qext0,LCODE)
        do ii = 1,nwk

            AST(ii) = AST1(ii)
```

```
        end do

C
C     UPDATE xnodel
C

        CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,da)
        CALL upatot(ntnode,NODBC,atotl,da)
        CALL thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)
C
C     UPDATE ttrg & omegag
C
        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,ttrg,omegag)
        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,t6rg,om6gag)
C
C   ASSEMBLY THE Nonlinear TANGENT STIFFNESS MATRIX: AST1
C
C
        CALL CLVEC(AST1,NWK)
        LCODE = 1
        alambd=almde
        if(ip2.eq.1) then
          CALL ASSEMN(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST1,qint,qext0,LCODE)
        end if
        if(ip2.eq.2) then
          CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST1,qint,qext0,LCODE)
        end if

        do ii = 1,nwk
          if(ip2.eq.1) then
            AST1(ii) = -AST1(ii)/almde
          end if
          if(ip2.eq.2) then
            AST1(ii) = (AST(ii)- AST1(ii))/almde
          end if
        end do

        MATZ=2
        call  EIGRGG(maxeqn,maxeqn,AST,AST1,WR,WI,FV1,
     &               MATZ,Z,LOC,Nmin,IERR)
```

```
      DO II=1,Nmin
        imag=0
        imag0=0
        LOCAT=LOC(II)
        do i = 1, locat-1
          if(dabs(WI(i)).gt.0.) then
            imag=imag+1
          end if
        end do

        if(dabs(WI(locat)).gt.0.) then
          imag0=1
        end if
        WRITE(10,*) '   '
        WRITE(10,*) '   '
        WRITE(10,4) II,WR(locat)
 4      FORMAT(2X,'Bifurcation buckling load factor',I2, '
=',E20.10/)
        WRITE(10,*) '   '
        WRITE(10,*) '   '
        WRITE(10,5)
 5      FORMAT(2X,'Buckling Shape=')


        do icol=1, 1+imag0
          do i=1,maxeqn
            FV1(i) = Z(i,locat+imag-icol+1)
          end do
          call recda1(FV1,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
          CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,FV1)
          CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
          do i=1,maxeqn
            Z(i,locat+imag-icol+1)=FV1(i)
          end do
        end do
        do icol=1, 1+imag0
          WRITE(10,6) (Z(i,locat+imag-icol+1), i=1,maxeqn)
        end do

 6      FORMAT(2X,5E20.10)
        WRITE(10,7) IERR
 7      FORMAT(2X,'IERR = ',I8/)
        WRITE(10,*) '   '
        WRITE(10,*) 'Note: only if IERR = 0, '
        WRITE(10,*) '      is eigen analysis successful!!'
      END DO

      return

      END



      SUBROUTINE EIGRG(NM,N,A,WR,WI,MATZ,Z,IV1,FV1,IERR,
     &                 da,CADD,LOC,NZERO)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(NM,N),WR(N),WI(N),Z(NM,N),FV1(N),IV1(N),da(*)
```

```
      DIMENSION LOC(*)
      CALL RG(NM,N,A,WR,WI,MATZ,Z,IV1,FV1,IERR)

      small= 10000000000.
      locat=0
      imag=0
      imag0=0

        do i = 1, n
          small1=dsqrt(wr(i)*wr(i)+wi(i)*wi(i))
          if(small1.lt.small) then
            locat=i
            small=small1
          end if
        end do
        LOC(1) = LOCAT
        NZERO=1
        DO I = 1,N
          IF(I.NE.LOCAT) THEN
            small1=dsqrt(wr(i)*wr(i)+wi(i)*wi(i))
            IF(DABS(SMALL1-SMALL).LT.0.01) THEN
               NZERO=NZERO+1
               LOC(NZERO)=I
            END IF
          END IF
        END DO

      DO II=1,NZERO
        imag=0
        imag0=0
        LOCAT=LOC(II)
        do i = 1, locat-1
          if(dabs(wi(i)).gt.0.) then
            imag=imag+1
          end if
        end do

        if(dabs(wi(locat)).gt.0.) then
          imag0=1
        end if

        WRITE(10,4) II,WR(locat)
 4      FORMAT(2X,'Eigenvalue',I2, ' =',E20.10/)

        WRITE(10,5)
 5      FORMAT(2X,'Eigenvector =')

        do icol=1, 1+imag0
          WRITE(10,6) (Z(i,locat+imag-icol+1), i=1,n)
        end do
 6      FORMAT(2X,5E20.10)
        WRITE(10,7) IERR
 7      FORMAT(2X,'IERR = ',I8/)
c        CALL VECVEC(da,da,dada,N)  ! inner product  da.da
      END DO
c        do iadd=1,N
c         da(iadd)=CADD*Z(iadd,locat+imag-1+1)
```

```
c         end do

          return
          END


          SUBROUTINE EIGRGG(NM,N,A,B,ALFR,ALFI,BETA,MATZ,Z,LOC,Nmin,IERR)
          IMPLICIT REAL*8(A-H,O-Z)
          DIMENSION A(NM,N),B(NM,N),ALFR(N),ALFI(N),BETA(N),Z(NM,N)
          dimension LOC(*)

          CALL RGG(NM,N,A,B,ALFR,ALFI,BETA,MATZ,Z,IERR)
c          write(10,*) 'alfr',(ALFR(ii),ii=1,n)
c         write(10,*) 'beta',(beta(ii),ii=1,n)
          small= 100000000000000.
          locat=0
          imag=0
          imag0=0
            do i = 1,n
              ALFR(i)=ALFR(i)/BETA(i)
              ALFI(i)=ALFI(i)/BETA(i)
            end do

            write(10,*) 'eigenvalues = '
            write(10,*) '  '
            write(10,*) (ALFR(ii),ii=1,n)
            do i = 1, n
              small1=ALFR(i)
              if(small1.lt.small.and.small1.gt.0.) then
                locat=i
                small=small1
              end if
            end do
            LOC(1) = LOCAT
            Nmin=1
            DO I = 1,N
              IF(I.NE.LOCAT) THEN
                small1=ALFR(i)
                IF(DABS(SMALL1-SMALL).LT.0.01) THEN
                   Nmin=Nmin+1
                   LOC(Nmin)=I
                END IF
              END IF
            END DO

          return
          END



          SUBROUTINE DIADET(AST,IPVT,DIAG,MAXEQN,NCHG,NNEGA,NNEGD,DET)
          IMPLICIT REAL*8(A-H,O-Z)
          DIMENSION AST(MAXEQN,MAXEQN),IPVT(MAXEQN), DIAG(MAXEQN)
          NNEGA=0
          DET = 1.
          DO II=1,MAXEQN
           DIAG(II) = AST(II,II)
```

```
         IF(DIAG(II).LT.0.) THEN
          NNEGA= NNEGA+1
         END IF
         DET=DET + DLOG(DABS(DIAG(II)))
        END DO
        idet = IPVT(MAXEQN)*(-1)**NNEGA
        NNEGD=NNEGA-NCHG
        WRITE(10, 5) NNEGA, NCHG, NNEGD
5       FORMAT(2X,'NNEGA =',I3,3X,'NCHG=',I3,3X,'NNEGD=',I3/)

        WRITE(10, *) 'log |DET| =', det, 'the sign of det ', idet

        RETURN
        END
```

```
C  analy5.f:
C
        SUBROUTINE ARCSTP(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,aext0,aint,
     &                    wr,wi,z,loc,fv1,iv1,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,ilast,
     &                    toltld,tolda,toldq,toldadq,cadd,
     &                    alamb0,alinc0,det0,rdet,nodepr,ip)

        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION LM(*),MAXH(*),MHT(*)
        DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(NTNODE,*)
        DIMENSION mhing1(2,*),mhing3(2,*)
        DIMENSION hing1(5,*),idofh1(*)
        DIMENSION CNCM(6,4,*)
        DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
        DIMENSION qext0(*),qint(*)
        DIMENSION da(*),dq(*),da0(*),dq0(*)
        DIMENSION sda(*),sdq(*),sxnode(*)
        DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
        DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
        DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
        DIMENSION ATOTL(*),ATOTL0(*),aext0(*),aint(*)
        DIMENSION aexw0(900),dar(900),ATOTL1(900),ATOTL2(900)
        DIMENSION darw(900), detdat(3),aaccbb(3)
        DIMENSION WR(*),WI(*),Z(MAXEQN,*),LOC(*)
        DIMENSION FV1(*),IV1(*)


        do ii = 1,3
          detdat(ii)=0.
          aaccbb(ii)=0.
        end do

        Npr=1
        nldc=0
        ldown=0
        niter=0
        idesir=6
        if(ip.eq.2) then
         idesir=7
        end if
        idet0 = 1
        alambd=alamb0
```

```
        CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)

18      Npr=Npr+1
        niter=0
        NLD=NLD+1

        IF(NLD.EQ.maxnld) then
           STOP 'MAXIMUM NUMBER OF LOAD STEPS EXCEEDED'
        END IF

        IF(dabs(alamb0).GT.toltld) then
           stop 'MAXIMUM LOAD FACTOR EXCEEDED'
        END IF

c       write(6,*) '   '
c       write(6,*) 'load level NLD=', nld, '  alambd =',alambd
c       write(6,*) '   '

        CALL CLVEC(AST,NWK)
        CALL CLVEC(dq0,MAX)

        MAXEQN=MAX-1


C
C   ASSEMBLY THE TANGENT STIFFNESS MATRIX: AST
C
C

63      continue

        CALL CLVEC(AST,NWK)
        CALL CLVEC(dq0,MAX)

        LCODE = 1
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)

        DO IASSIGN = 1, NWK
          AST1(IASSIGN)=AST(IASSIGN)
        END DO
C
C   DECOMPOSITION OF TANGENT STIFFNESS MATRIX
C
        CALL DECOMP1(MAXEQN,MAXEQN,AST,COND,IPVT,NCHG,WORK)

C
C   ASSEMBLY THE INTITIAL UNBALANCED LOAD,
C               AT CURRENT LOAD LEVEL
C
```

```
         CALL CLVEC(dq,MAX)
         LCODE = 2
         CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                     nhing1,nhing3,
     &                     mhing1,mhing3,
     &                     hing1,idofh1,
     &                     distgg,distnn,distpp,ptload,alambd,
     &                     ttrg,t6rg,omegag,om6gag,da,
     &                     LM,MAXH,MHT,NWK,MK,
     &                     AST,qint,qext0,LCODE)

         CALL RESIDL(qext0,qint,maxeqn,alambd,dq)

c        write(6,*) 'intitial unbalanced load, external load'
c        write(6,*) 'dq=',(dq(iii),iii=1,maxeqn)
c        write(6,*) 'qext0=',(qext0(iii),iii=1,maxeqn)
c        write(6,*) 'atotl=',(atotl(iii),iii=1,maxeqn)

         if(nld.eq.1) then
           DO ING=1,MAXEQN
             dq0(ING)=dq(ING)  ! get the  intitial unbalanced LOAD
           END DO
         end if


64       CONTINUE


         ITR=1     ! number of iterations inside the incremental load
loop

C-----SOLVE THE MATRIX EQUATIONS TO GET DISPLACEMENTS
C     DUE TO INTITIAL UNBALANCED LOAD

         DO ING=1,MAXEQN
           aext0(ING)=qext0(ING)
           dar(ING)=dq(ING)
         END DO

         CALL SOLVE1(MAXEQN,MAXEQN,AST,aext0,IPVT)
         CALL SOLVE1(MAXEQN,MAXEQN,AST,dar,IPVT)
C
C     RECOVER daQ
C
         call recda1(aext0,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
         call recda1(dar,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)

c        write(6,*) 'aext0=',(aext0(iii),iii=1,maxeqn)
         DO ING=1,MAXEQN
           aexw0(ING)=aext0(ING)
         END DO

           call aarotv(ntnode,NODBC,atotl,aexw0)
           CALL VECVEC(aexw0,aexw0,aexaex,MAXEQN)

         IF(NLD.EQ.1) THEN
            c3 = 2.
```

```
           if(ip.eq.1) then
             c3=1.01
           end if
          arclen = alinc0*dsqrt(c3*aexaex)
          c2=(c3-1.)*aexaex
          write(6,*) 'CCCCCCC = ', c2

        ELSE IF(NLD.GT.1) THEN
           faclim=1.1
          if(ip.eq.2) then
            faclim=1.001
          end if
          fac=dsqrt(dfloat(idesir)/dfloat(ILAST))
          if(fac.ge.faclim) then
            fac = faclim
          end if
          if(fac.lt.0.5) then
            fac = 0.5
          end if
          arclen = fac*arcln0
          if(idet.ne.idet0) then
            idet0 = -idet0
          end if
          alinc0 = dfloat(idet0)*arclen/dsqrt(c3*aexaex)
          c2=(c3-1.)*aexaex
        END IF
        write(6,*) 'CCCCCCC = ', c2
        alinc=alinc0

        write(6,*) 'alinc, arclen',alinc, arclen

        do ii = 1,maxeqn
          da(ii) = dar(ii)+alinc*aext0(ii)
        end do

        alambd = alambd+alinc

        IF(NLD.EQ.1) THEN
          DO ING=1,MAXEQN
            da0(ING)=da(ING) !store the displ due to intitial
unbalanced LOAD
          END DO
          CALL VECVEC(dq0,dq0,dq0dq0,MAXEQN)  ! inner product dq0.dq0
          CALL VECVEC(da0,da0,da0da0,MAXEQN)  ! inner product  da0.da0
        END IF


        ITR=1     ! number of iterations inside the incremental load
loop

24      CONTINUE        ! iterations inside a load step start here

        ITR=ITR+1
        niter=niter+1
C       IF(ITR.GT.76) GO TO 28

C
```

```
C     UPDATE xnodel, atotl
C
        CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,da)
        CALL upatot(ntnode,NODBC,atotl,da)
        call thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)

        do ii = 1,maxeqn
           atotl1(ii)=atotl(ii)
           atotl2(ii)=atotl(ii)
        end do
C
C     UPDATE ttrg & omegag
C
        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,ttrg,omegag)
        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,t6rg,om6gag)
C
C   ASSEMBLY THE UPDATED TANGENT STIFFNESS MATRIX: AST
C
C
        CALL CLVEC(AST,NWK)
        LCODE = 1
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)
        DO IASSIGN = 1, NWK
          AST1(IASSIGN)=AST(IASSIGN)
        END DO

C
C   DECOMPOSITION OF TANGENT STIFFNESS MATRIX
C
        CALL DECOMP1(MAXEQN,MAXEQN,AST,COND,IPVT,NCHG,WORK)

C
C   ASSEMBLY THE unbalanced LOAD for updated configuration
C               at current load level
C
        CALL CLVEC(dq,MAX)
        LCODE = 2
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)
        CALL RESIDL(qext0,qint,maxeqn,alambd,dq)

        DO ING=1,MAXEQN
```

```
          aext0(ING)=qext0(ING)
          dar(ING)=dq(ING)
        END DO

c
c-----SOLVE THE MATRIX EQUATIONS TO GET DISPLACEMENTS under unbalanced
force
c

        CALL SOLVE1(MAXEQN,MAXEQN,AST,aext0,IPVT)
        CALL SOLVE1(MAXEQN,MAXEQN,AST,dar,IPVT)
C
C     RECOVER daQ
C
        call recda1(aext0,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
        call recda1(dar,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)

C
C       GET NEW LOADING FACTOR BASED ON ARC LENGTH
C
        do ii = 1, maxeqn
          aexw0(ii) = aext0(ii)
          darw(ii) = dar(ii)
        end do

        call aarotv(ntnode,NODBC,atotl,aexw0)
        call aarotv(ntnode,NODBC,atotl,darw)

        aexaex=0.
        aedain=0.
        dain2=0.

        do ii =1,maxeqn
              aexaex=aexaex+aexw0(ii)*aexw0(ii)
              aedain=aedain+aexw0(ii)
     &          *(atotl(ii)-atotl0(ii)+darw(ii))
              dain2=dain2+
     &          (atotl(ii)-atotl0(ii)+darw(ii))
     &         *(atotl(ii)-atotl0(ii)+darw(ii))
        end do

        aa=aexaex+c2
        bb=2.*aedain+2.*(alambd-alamb0)*c2
        cc=dain2-arclen*arclen+(alambd-alamb0)*(alambd-alamb0)*c2

        crit=bb*bb-4.*aa*cc
        write(6,*) 'crit =', crit

        if(crit.lt.0.) then
            ilast = 100
            write(6,*) 'itr =itr =itr =', itr
            go to 66
        end if


        dalmd1=(-bb+dsqrt(crit))/(2.*aa)
```

```
          dalmd2=(-bb-dsqrt(crit))/(2.*aa)


c         write(6,*) 'aa =',aa , 'bb=', bb, 'cc=', cc

          write(6,*) 'dalmd1 =',dalmd1 , 'dalmd2=', dalmd2

          da1da=0.
          da2da=0.

          do ii = 1,maxeqn
            da(ii) = dar(ii)+dalmd1*aext0(ii)
          end do
          call upatot(ntnode,NODBC,atotl1,da)
          do ii = 1,maxeqn
            da(ii) = dar(ii)+dalmd2*aext0(ii)
          end do
          call upatot(ntnode,NODBC,atotl2,da)

          do ii = 1, maxeqn
           da1da = da1da+(atotl1(ii)-atotl0(ii))*(atotl(ii)-atotl0(ii))
           da2da = da2da+(atotl2(ii)-atotl0(ii))*(atotl(ii)-atotl0(ii))
          end do

c         write(6,*) 'da1da  =',da1da ,'da2da  =',da2da

          if(da1da.ge.0.0.and.da2da.ge.0.0) then
               alinc =dalmd2
             if(dabs(dalmd1+cc/bb).lt.dabs(dalmd2+cc/bb)) then
               alinc =dalmd1
             end if
          else if(da1da.gt.0.0.and.da2da.le.0.0) then
            alinc = dalmd1
          else if(da2da.gt.0.0.and.da1da.le.0.0) then
            alinc = dalmd2
          else
            alinc = almd1
          end if

          do ii = 1,maxeqn
            da(ii) = dar(ii)+alinc*aext0(ii)
          end do

          alambd =alambd+alinc
          write(6,*) 'ARC =', arclen, 'alambd=', alambd
          CALL RESIDL(qext0,qint,maxeqn,alambd,dq)


          CALL VECVEC(dq,dq,dqdq,MAXEQN)  ! inner product dq.dq

          CALL VECVEC(da,da,dada,MAXEQN)  ! inner product  da.da

          dq0dq0=1.d0
          da0da0=1.d0

          RATIOL=DSQRT(dqdq/dq0dq0)
          RATIOD=DSQRT(dada/da0da0)
```

```
        RATLD=DSQRT(dqdq/dada)
        WRITE (16,*) itr,RATIOL
C       WRITE (6,*) 'RATLD=',RATLD

C
C   CONVERGENCE CHECKING FOR LOAD RESIDUAL
C

        IF(RATIOL.LE.toldq) GO TO 28    !CONVERGENCE

C
C   CONVERGENCE CHECKING FOR DISPLACEMENT INCREMENT
C
        IF(RATIOD.LE.tolda) GO TO 28    !CONVERGENCE
c        IF(arclen.LE.tolda) GO TO 28    !CONVERGENCE

C
C   CONVERGENCE CHECKING FOR LOAD RESIDUAL AND DISPLACEMENT INCREMENT
C
C       IF(RATIOD.LE.toldadq.AND.RATIOL.LE.toldadq) GO TO 28
!CONVERGENCE

C
C   CHECK MAXIMUM NUMBER OF ITERATIONS INSIDE OF CURRENT LOAD LOOP
C

        IF(ITR.GT.MAXITR) GO TO 66           ! EXCEED


        GO TO 24                          ! GO TO ANOTHER ITR LOOP

28      CONTINUE                          ! GET HERE IF CONVERGENCE
ACHIEVES.
        nldc=nldc+1
        write(6,*) 'nldc =', nldc
        WRITE(6,*) 'The number of iteration itr =',itr
c        WRITE(6,*) 'dada=',dada, 'dqdq=',dqdq
c        WRITE (6,20003) RATIOD,RATIOL
20003   FORMAT(//2X,'RATIOS: RATIOD =',1PE20.10,'RATIOL =',1PE20.10)
c        WRITE (6,*) 'RATLD=',RATLD

        WRITE(10,*) '    '
        WRITE(10,*) '============================================='
        write(10,*) 'load level NLD=', nld, '     alambd=',alambd
C
C
C
        CALL DIADET(AST,IPVT,DIAG,MAXEQN,NCHG,NNEGA,NNEGD,DET)
        idet = IPVT(MAXEQN)*(-1)**NNEGA

        if(nld.eq.1) then
          det0=det
          bbb=alambd
          fbbb=1.
        else if (nld.gt.1) then

          rdet=dfloat(idet)*dexp(det-det0)
```

```
      write(10,*) 'Relative Det =', rdet
      write(12,*) alambd, rdet
      if(ip.eq.2) then
        aaa=bbb
        faaa=fbbb
        bbb=alambd
        fbbb=rdet
        detdat(1) = detdat(2)
        detdat(2) = detdat(3)
        detdat(3) = dsqrt(dsqrt(rdet*rdet))
        aaccbb(1) = aaccbb(2)
        aaccbb(2) = aaccbb(3)
        aaccbb(3) = alambd
        if(nldc.gt.2) then
          if(faaa*fbbb.le.0.) then
        write(10,*) '====== Using Bisection Approach ======  '
        write(10,*) '                                '
        write(10,*) '====== Using Bisection Approach ======  '

            call BIFUR2(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
&                     LM,MAXH,MHT,
&                     X0,XYZN,xnodel,melm,NODBC,
&                     nhing1,nhing3,
&                     mhing1,mhing3,
&                     hing1,idofh1,
&                     cncm,
&                     ptload,distgg,distnn,distpp,
&                     qext0,qint,da,dq,da0,dq0,
&                     sda,sdq,sxnode,
&                     ttrg,t6rg,omegag,om6gag,
&                     sttrg,st6rg,somegg,som6gg,
&                     AST,AST1,diag,IPVT,WORK,
&                     atotl,atotl0,
&                     maxnld,maxldn,maxitr,
&                     NLD,ldown,niter,
&                     toltld,tolda,toldq,toldadq,
&                     alambd,alamb0,det0,rdet,
&                     aaa,bbb,faaa,fbbb)
      write(10,*) '    '
      write(10,*) '    '
      write(10,*) '====== Using Bisection Approach ======  '
      write(10,*) '    '
      write(10,*) '    '
      write(10,*) ' bifurcation buclking load factor:',alambd
            nldc =0
            write(12,*) alambd, rdet
        MATZ=2
        CALL EIGRG(MAXEQN,MAXEQN,AST1,WR,WI,MATZ,Z,IV1,FV1,IERR,
&                  da,CADD,LOC,Nmin)


     DO II=1,Nmin
       imag=0
       imag0=0
       LOCAT=LOC(II)
       do i = 1, locat-1
         if(dabs(WI(i)).gt.0.) then
```

```
            imag=imag+1
          end if
        end do

        if(dabs(WI(locat)).gt.0.) then
          imag0=1
        end if
        write(10,*) '    '
        write(10,*) '    '
        WRITE(10,500)
500     FORMAT(2X,'Buckling Shape=')
        write(10,*) '    '
        do icol=1, 1+imag0
          do i=1,maxeqn
            FV1(i) = Z(i,locat+imag-icol+1)
          end do
          call recda1(FV1,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
          CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,FV1)
          CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
          do i=1,maxeqn
            Z(i,locat+imag-icol+1)=FV1(i)
          end do
        end do
        do icol=1, 1+imag0
          WRITE(10,600) (Z(i,locat+imag-icol+1), i=1,maxeqn)
        end do

600     FORMAT(2X,5E20.10)
        WRITE(10,*) '   '
        WRITE(10,700) IERR
700     FORMAT(2X,'IERR = ',I8/)
        WRITE(10,*) '   '
        WRITE(10,*) 'Note: only if IERR = 0, '
        WRITE(10,*) '      is eigen analysis successful!!'
      END DO
            return
          end if
        end if

          if(nldc.gt.3) then
            if((detdat(3)-detdat(2)).gt.0.
     &        .and.(detdat(1)-detdat(2)).ge.0.
     &        .and.dabs(detdat(2)).le.0.5) then
      alamb0 = alambd
      alambd = alamb0 - 0.25*alinc0
      write(10,*) '====== Using 3 Points Interplotion ======  '
        write(10,*) '       '
        write(10,*) '      '
      write(10,*) '====== Using 3 Points Interplotion ======  '
      CALL BIFUR3(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                   LM,MAXH,MHT,
     &                   X0,XYZN,xnodel,melm,NODBC,
     &                   nhing1,nhing3,
     &                   mhing1,mhing3,
     &                   hing1,idofh1,
     &                   cncm,
     &                   ptload,distgg,distnn,distpp,
```

```
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)
          write(10,*) '       '
          write(10,*) '      '
          write(10,*) '====== Using 3 Points Interplotion ======  '
          write(10,*) '    '
          write(10,*) '    '
          write(10,*) 'Bifurcation point is : alambd = ', alambd
                nldc = 0
c              write(12,*) alambd, rdet
          MATZ=2
          call EIGRG(MAXEQN,MAXEQN,AST1,WR,WI,MATZ,Z,IV1,FV1,IERR,
     &                  da,CADD,LOC,Nmin)

         DO II=1,Nmin
           imag=0
           imag0=0
           LOCAT=LOC(II)
           do i = 1, locat-1
             if(dabs(WI(i)).gt.0.) then
               imag=imag+1
             end if
           end do

           if(dabs(WI(locat)).gt.0.) then
             imag0=1
           end if
           write(10,*) '    '
           write(10,*) '    '
           WRITE(10,5)
 5         FORMAT(2X,'Buckling Shape =')
           write(10,*) '    '
           do icol=1, 1+imag0
             do i=1,maxeqn
               FV1(i) = Z(i,locat+imag-icol+1)
             end do
             call recda1(FV1,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
             CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,FV1)
             CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
             do i=1,maxeqn
               Z(i,locat+imag-icol+1)=FV1(i)
             end do
           end do
           do icol=1, 1+imag0
             WRITE(10,6) (Z(i,locat+imag-icol+1), i=1,maxeqn)
           end do

 6         FORMAT(2X,5E20.10)
           WRITE(10,*) '   '
```

```
          WRITE(10,7) IERR
 7        FORMAT(2X,'IERR = ',I8/)
          WRITE(10,*) '   '
          WRITE(10,*) 'Note: only if IERR = 0, '
          WRITE(10,*) '      is eigen analysis successful!!'
        END DO

        return

C
C     RECOVER daQ
C
c         call recda1(da,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
C
C     UPDATE xnodel, atotl
C
c        CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,da)
c       CALL upatot(ntnode,NODBC,atotl,da)
c       call thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)
c
c       do ii = 1,maxeqn
c           atotl1(ii)=atotl(ii)
c           atotl2(ii)=atotl(ii)
c       end do
C
C     UPDATE ttrg & omegag
C
c        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
c     &              x0,da,ttrg,omegag)
c        CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
c     &              x0,da,t6rg,om6gag)

             end if
           end if
          end if

        end if

C
C    GET AND PRINT THE NEW NODAL POSITION VECTOR XYZN
C
c        if(npr.eq.1) then
          write(6,*) 'load level NLD=', nld, '     alambd=',alambd
          CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
          CALL PRDISP(NTNODE,NODBC,atotl,nodepr,alambd)

          npr=0
c        end if


        do ii = 1, maxeqn
         atotl0(ii) = atotl(ii)
        end do


        call tempsv(maxeqn,nelm,
     &              xnodel,ttrg,omegag,da,dq,
```

```
     &              sxnode,sttrg,somegg,sda,sdq)

         call tempsv(maxeqn,nelm,
     &              xnodel,t6rg,om6gag,da,dq,
     &              sxnode,st6rg,som6gg,sda,sdq)

         alamb0 = alambd
         arcln0 = arclen
         ilast=niter
         ldown=0
         go to 18        ! go to new load loop
C
C
C
66       CONTINUE
C
C     CONVERGENCE NOT ACHIEVED. REDUCE THE LOAD STEP BY HALF.
C
c        write(6,*) 'Load increment is reduced by half'

         ilast = 100
         ldown=ldown+1
         alambd=alamb0
         arcln0 = arclen
         idet = idet0
         write(6,*) 'Load level is reduced to alambd= ', alambd

         call tempgt(maxeqn,nelm,
     &             xnodel,ttrg,omegag,da,dq,
     &             sxnode,sttrg,somegg,sda,sdq)
         call tempgt(maxeqn,nelm,
     &             xnodel,t6rg,om6gag,da,dq,
     &             sxnode,st6rg,som6gg,sda,sdq)
         do ii = 1, maxeqn
          atotl(ii) = atotl0(ii)
         end do
         call thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)

         IF(ldown.GT.maxldn) then
           write(6,*) 'idown = ', ldown
           stop 'no convergence can be achieved'
         end if

         GO TO 18

         END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
c
c   transfer rotational components of aext0 n aint into rotation vector
form
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc
         subroutine aarotv(ntnode,NODBC,atotl,daw)
```

```
IMPLICIT REAL*8 (A-H,O-Z)
dimension atotl(*),daw(*)
dimension dwvec(3),theta(3),dthet(3)
dimension NODBC(ntnode,6)

do ii=1,ntnode

    dwvec(1)=0.
    dwvec(2)=0.
    dwvec(3)=0.

    theta(1)=0.
    theta(2)=0.
    theta(3)=0.

  do jj=1,3
    ndof=NODBC(II,JJ+3)
    if(ndof.ne.0) then
      theta(jj)=atotl(ndof)
      dwvec(jj)= daw(ndof)
    end if
  end do

  call dwthet(dwvec,theta,dthet)

  do jj=1,3
    ndof=NODBC(II,JJ+3)
    if(ndof.ne.0) then
      daw(ndof)=dthet(jj)
    end if
  end do

end do


return
end
```

```
C  analy6.f:
C
        SUBROUTINE LODSTP(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                    LM,MAXH,MHT,
     &                    X0,XYZN,xnodel,melm,NODBC,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,aext0,aint,
     &                    wr,wi,z,loc,fv1,iv1,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,ilast,
     &                    toltld,tolda,toldq,toldadq,
     &                    alamb0,alinc0,det0,rdet,nodepr,ip)
c
c   ip = 1, dont seek the range in which, bifurcation point is
c
c   ip = 2, seek the range in which, bifurcation point is
c

        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION LM(*),MAXH(*),MHT(*)
        DIMENSION X0(*),XYZN(*),xnodel(*),MELM(*),NODBC(NTNODE,*)
        DIMENSION mhing1(2,*),mhing3(2,*)
        DIMENSION hing1(5,*),idofh1(*)
        DIMENSION CNCM(6,4,*)
        DIMENSION PTLOAD(*),DISTGG(*),DISTNN(*),DISTPP(*)
        DIMENSION qext0(*),qint(*)
        DIMENSION da(*),dq(*),da0(*),dq0(*)
        DIMENSION sda(*),sdq(*),sxnode(*)
        DIMENSION ttrg(*),t6rg(*),omegag(*),om6gag(*)
        DIMENSION sttrg(*),st6rg(*),somegg(*),som6gg(*)
        DIMENSION AST(*),AST1(*),diag(*),IPVT(*),WORK(*)
        DIMENSION ATOTL(*),ATOTL0(*),aext0(*),aint(*)
        DIMENSION aexw0(900),dar(900),ATOTL1(900),ATOTL2(900)
        DIMENSION darw(900), detdat(3),aaccbb(3)
        DIMENSION WR(*),WI(*),Z(MAXEQN,*),LOC(*)
        DIMENSION FV1(*),IV1(*)

        do ii = 1,3
          detdat(ii)=0.
          aaccbb(ii)=0.
        end do

        Npr=1
        nldc=0
        ldown=0
        niter=0
        IDEDIR=10
        idet0 = 1
```

```
        alambd=alamb0

        CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)

18      Npr=Npr+1


        MAXEQN=MAX-1


C
C   ASSEMBLY THE TANGENT STIFFNESS MATRIX: AST
C
C

63      continue


        CALL CLVEC(AST,NWK)
        CALL CLVEC(dq0,MAX)

        LCODE = 1

        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)

        DO IASSIGN = 1, NWK
          AST1(IASSIGN)=AST(IASSIGN)
        END DO
C
C   DECOMPOSITION OF TANGENT STIFFNESS MATRIX
C
        CALL DECOMP1(MAXEQN,MAXEQN,AST,COND,IPVT,NCHG,WORK)
c       call praskd(ast,maxeqn)
C
C   ASSEMBLY THE INTITIAL UNBALANCED LOAD,
C               AT CURRENT LOAD LEVEL
C
        CALL CLVEC(dq,MAX)
        LCODE = 2
        CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                    nhing1,nhing3,
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    distgg,distnn,distpp,ptload,alambd,
     &                    ttrg,t6rg,omegag,om6gag,da,
     &                    LM,MAXH,MHT,NWK,MK,
     &                    AST,qint,qext0,LCODE)

        CALL RESIDL(qext0,qint,maxeqn,alambd,dq)
```

```
c       write(6,*) 'intitial unbalanced load, external load'
c       write(6,*) 'dq=',(dq(iii),iii=1,maxeqn)
c       write(6,*) 'qext0=',(qext0(iii),iii=1,maxeqn)
c       write(6,*) 'atotl=',(atotl(iii),iii=1,maxeqn)

        if(nld.eq.1) then
          DO ING=1,MAXEQN
            dq0(ING)=dq(ING)  ! get the  intitial unbalanced LOAD
          END DO
        end if


64      CONTINUE

        niter=0
        NLD=NLD+1

        IF(NLD.EQ.maxnld) then
           STOP 'MAXIMUM NUMBER OF LOAD STEPS EXCEEDED'
        END IF

        IF(dabs(alamb0).GT.toltld) then
           stop 'MAXIMUM LOAD FACTOR EXCEEDED'
        END IF

c       write(6,*) '   '
c       write(6,*) 'load level NLD=', nld, '  alambd =',alambd
c       write(6,*) '   '


        ITR=1     ! number of iterations inside the incremental load
loop

C-----SOLVE THE MATRIX EQUATIONS TO GET DISPLACEMENTS
C     DUE TO INTITIAL UNBALANCED LOAD

        DO ING=1,MAXEQN
          aext0(ING)=qext0(ING)
          dar(ING)=dq(ING)
        END DO
c        write(6,*) 'qext0=',(qext0(iii),iii=1,maxeqn)
        CALL SOLVE1(MAXEQN,MAXEQN,AST,aext0,IPVT)
        CALL SOLVE1(MAXEQN,MAXEQN,AST,dar,IPVT)
c        write(6,*) 'aext0=',(aext0(iii),iii=1,maxeqn)
c        write(6,*) 'dar1=',(dar(iii),iii=1,maxeqn)
C
C     RECOVER daQ
C

        call recda1(aext0,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
        call recda1(dar,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
c        write(6,*) 'aext2=',(aext0(iii),iii=1,maxeqn)
c       write(6,*) 'dar2=',(dar(iii),iii=1,maxeqn)
c        IF(NLD.GT.1) THEN

c         fac=dsqrt(dfloat(IDEDIR)/dfloat(ILAST))
c         if(fac.ge.1.0) then
```

```
c           fac = 1.0
c         end if
c         if(fac.lt.0.5) then
c           fac = 0.5
c         end if

c         alinc0 = fac*alinc0

c        END IF

         alinc=alinc0
c         write(6,*) 'alinc0= ', alinc0
         do ii = 1,maxeqn
           da(ii) = dar(ii)+alinc*aext0(ii)
         end do
c        write(6,*) 'da=',(da(iii),iii=1,maxeqn)
         alambd = alambd+alinc
         write(6,*) 'alambd = ',alambd

         IF(NLD.EQ.1) THEN
           DO ING=1,MAXEQN
             da0(ING)=da(ING) !store the displ due to intitial
unbalanced LOAD
           END DO
           CALL VECVEC(dq0,dq0,dq0dq0,MAXEQN)  ! inner product dq0.dq0
           CALL VECVEC(da0,da0,da0da0,MAXEQN)  ! inner product  da0.da0
         END IF


         ITR=1     ! number of iterations inside the incremental load
loop

24       CONTINUE        ! iterations inside a load step start here

         ITR=ITR+1
         niter=niter+1
C        IF(ITR.GT.76) GO TO 28
C
C     UPDATE xnodel, atotl
C

         CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,da)
         CALL upatot(ntnode,NODBC,atotl,da)

         call thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)
c        write(6,*) 'atotl=',(atotl(iii),iii=1,maxeqn)
C
C     UPDATE ttrg & omegag
C
         CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,ttrg,omegag)
         CALL ttromg(NTNODE,NELM,MELM,NODBC,max,
     &              x0,da,t6rg,om6gag)
C
C   ASSEMBLY THE UPDATED TANGENT STIFFNESS MATRIX: AST
C
C
```

```
         CALL CLVEC(AST,NWK)
         LCODE = 1
         CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                     nhing1,nhing3,
     &                     mhing1,mhing3,
     &                     hing1,idofh1,
     &                     distgg,distnn,distpp,ptload,alambd,
     &                     ttrg,t6rg,omegag,om6gag,da,
     &                     LM,MAXH,MHT,NWK,MK,
     &                     AST,qint,qext0,LCODE)

         DO IASSIGN = 1, NWK
           AST1(IASSIGN)=AST(IASSIGN)
         END DO
c        write(6,*) 'qext0=',(qext0(iii),iii=1,maxeqn)
C
C   DECOMPOSITION OF TANGENT STIFFNESS MATRIX
C

         CALL DECOMP1(MAXEQN,MAXEQN,AST,COND,IPVT,NCHG,WORK)

C
C   ASSEMBLY THE unbalanced LOAD for updated configuration
C                at current load level
C
         CALL CLVEC(dq,MAX)
         LCODE = 2
         CALL ASSEMB(NTNODE,NELM,melm,x0,xnodel,cncm,NODBC,MAXEQN,
     &                     nhing1,nhing3,
     &                     mhing1,mhing3,
     &                     hing1,idofh1,
     &                     distgg,distnn,distpp,ptload,alambd,
     &                     ttrg,t6rg,omegag,om6gag,da,
     &                     LM,MAXH,MHT,NWK,MK,
     &                     AST,qint,qext0,LCODE)
         CALL RESIDL(qext0,qint,maxeqn,alambd,dq)

         DO ING=1,MAXEQN
           dar(ING)=dq(ING)
         END DO
c         write(6,*) 'qext3=',(qext0(iii),iii=1,maxeqn)
c         write(6,*) 'qint3=',(qint(iii),iii=1,maxeqn)
c         write(6,*) 'dq3=',(dar(iii),iii=1,maxeqn)
c
c-----SOLVE THE MATRIX EQUATIONS TO GET DISPLACEMENTS under unbalanced
force
c

         CALL SOLVE1(MAXEQN,MAXEQN,AST,dar,IPVT)

c         write(6,*) 'dar3=',(dar(iii),iii=1,maxeqn)
C
C     RECOVER daQ
C
         call recda1(dar,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
```

```
        do ii = 1,maxeqn
          da(ii) = dar(ii)
        end do

        CALL VECVEC(dq,dq,dqdq,MAXEQN)  ! inner product dq.dq

        CALL VECVEC(da,da,dada,MAXEQN)  ! inner product  da.da

c        dq0dq0=1.d0
c        da0da0=1.d0

        RATIOL=DSQRT(dqdq/dq0dq0)
        RATIOD=DSQRT(dada/da0da0)
        RATLD=DSQRT(dqdq/dada)
        WRITE (16,*) itr,RATIOL
C       WRITE (6,*) 'RATLD=',RATLD

C
C   CONVERGENCE CHECKING FOR LOAD RESIDUAL
C

        IF(RATIOL.LE.toldq) GO TO 28    !CONVERGENCE

C
C   CONVERGENCE CHECKING FOR DISPLACEMENT INCREMENT
C
         IF(RATIOD.LE.tolda) GO TO 28    !CONVERGENCE

C
C   CONVERGENCE CHECKING FOR LOAD RESIDUAL AND DISPLACEMENT INCREMENT
C
C       IF(RATIOD.LE.toldadq.AND.RATIOL.LE.toldadq) GO TO 28
!CONVERGENCE

C
C   CHECK MAXIMUM NUMBER OF ITERATIONS INSIDE OF CURRENT LOAD LOOP
C

        IF(ITR.GT.MAXITR) GO TO 66           ! EXCEED


        GO TO 24                          ! GO TO ANOTHER ITR LOOP

28      CONTINUE                          ! GET HERE IF CONVERGENCE
ACHIEVES.

        nldc=nldc+1
        WRITE(6,*) 'The number of iteration itr =',itr
c        WRITE(6,*) 'dada=',dada, 'dqdq=',dqdq
c        WRITE (6,20003) RATIOD,RATIOL
20003   FORMAT(//2X,'RATIOS: RATIOD =',1PE20.10,'RATIOL =',1PE20.10)
c        WRITE (6,*) 'RATLD=',RATLD

        WRITE(10,*) '    '
        WRITE(10,*) '============================================='
        write(10,*) 'load level NLD=', nld, '     alambd=',alambd
C
```

```
C
C
      CALL DIADET(AST,IPVT,DIAG,MAXEQN,NCHG,NNEGA,NNEGD,DET)
      idet = IPVT(MAXEQN)*(-1)**NNEGA

      if(nld.eq.1) then
        det0=det
        bbb=alambd
        fbbb=1.
      else if (nld.gt.1) then
        rdet=dfloat(idet)*dexp(det-det0)

        write(10,*) 'Relative Det =', rdet
        write(12,*) alambd, rdet
        if(ip.eq.2) then
          aaa=bbb
          faaa=fbbb
          bbb=alambd
          fbbb=rdet
          detdat(1) = detdat(2)
          detdat(2) = detdat(3)
          detdat(3) =dsqrt(dsqrt(rdet*rdet))
          if(nldc.gt.2) then
            if(faaa*fbbb.le.0.) then
       write(10,*) '====== Using Bisection Approach ======  '
       write(10,*) '    '
       write(10,*) '====== Using Bisection Approach ======  '
       write(10,*) '    '
             call BIFUR2(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                   LM,MAXH,MHT,
     &                   X0,XYZN,xnodel,melm,NODBC,
     &                   nhing1,nhing3,
     &                   mhing1,mhing3,
     &                   hing1,idofh1,
     &                   cncm,
     &                   ptload,distgg,distnn,distpp,
     &                   qext0,qint,da,dq,da0,dq0,
     &                   sda,sdq,sxnode,
     &                   ttrg,t6rg,omegag,om6gag,
     &                   sttrg,st6rg,somegg,som6gg,
     &                   AST,AST1,diag,IPVT,WORK,
     &                   atotl,atotl0,
     &                   maxnld,maxldn,maxitr,
     &                   NLD,ldown,niter,
     &                   toltld,tolda,toldq,toldadq,
     &                   alambd,alamb0,det0,rdet,
     &                   aaa,bbb,faaa,fbbb)

       write(10,*) '====== Using Bisection Approach ======  '
       write(10,*) '    '
       write(10,*) '    '
       write(10,*) ' bifurcation buclking load factor:',alambd
             nldc =0
             write(12,*) alambd, rdet
          MATZ=2
          CALL EIGRG(MAXEQN,MAXEQN,AST1,WR,WI,MATZ,Z,IV1,FV1,IERR,
     &                  da,CADD,LOC,Nmin)
```

```
      DO II=1,Nmin
        imag=0
        imag0=0
        LOCAT=LOC(II)
        do i = 1, locat-1
          if(dabs(WI(i)).gt.0.) then
            imag=imag+1
          end if
        end do

        if(dabs(WI(locat)).gt.0.) then
          imag0=1
        end if
        write(10,*) '    '
        write(10,*) '    '
        WRITE(10,500)
500     FORMAT(2X,'Buckling Shape =')

        do icol=1, 1+imag0
          do i=1,maxeqn
            FV1(i) = Z(i,locat+imag-icol+1)
          end do
          call recda1(FV1,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
          CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,FV1)
          CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
          do i=1,maxeqn
            Z(i,locat+imag-icol+1)=FV1(i)
          end do
        end do
        do icol=1, 1+imag0
          WRITE(10,600) (Z(i,locat+imag-icol+1), i=1,maxeqn)
        end do

600     FORMAT(2X,5E20.10)
        write(10,*) '    '
        WRITE(10,700) IERR
700     FORMAT(2X,'IERR = ',I8/)
        WRITE(10,*) '   '
        WRITE(10,*) 'Note: only if IERR = 0, '
        WRITE(10,*) '      is eigen analysis successful!!'
      END DO
             return
           end if
         end if
         if(nldc.gt.3) then
            if((detdat(3)-detdat(2)).gt.0.
     &        .and.(detdat(1)-detdat(2)).ge.0.
     &        .and.dabs(detdat(2)).le.0.1) then
      alamb0 = alambd
      alambd = alamb0 - 0.2*alinc
       write(10,*) '====== Using 3 Points Interplotion ======  '
      CALL BIFUR3(NTNODE,NELM,NWK,MK,MAX,MAXEQN,
     &                  LM,MAXH,MHT,
     &                  X0,XYZN,xnodel,melm,NODBC,
     &                  nhing1,nhing3,
```

```
     &                    mhing1,mhing3,
     &                    hing1,idofh1,
     &                    cncm,
     &                    ptload,distgg,distnn,distpp,
     &                    qext0,qint,da,dq,da0,dq0,
     &                    sda,sdq,sxnode,
     &                    ttrg,t6rg,omegag,om6gag,
     &                    sttrg,st6rg,somegg,som6gg,
     &                    AST,AST1,diag,IPVT,WORK,
     &                    atotl,atotl0,
     &                    maxnld,maxldn,maxitr,
     &                    NLD,ldown,niter,
     &                    toltld,tolda,toldq,toldadq,
     &                    alambd,alamb0,det0,rdet)

          write(10,*) '====== Using 3 Points Interplotion ======  '
          write(10,*) '    '
          write(10,*) '    '
          write(10,*) ' bifurcation buclking load factor:',alambd
                nldc =0
                write(12,*) alambd, rdet
            MATZ=2
            CALL EIGRG(MAXEQN,MAXEQN,AST1,WR,WI,MATZ,Z,IV1,FV1,IERR,
     &                  da,CADD,LOC,Nmin)


         DO II=1,Nmin
           imag=0
           imag0=0
           LOCAT=LOC(II)
           do i = 1, locat-1
             if(dabs(WI(i)).gt.0.) then
               imag=imag+1
             end if
           end do

           if(dabs(WI(locat)).gt.0.) then
             imag0=1
           end if
           write(10,*) '    '
           write(10,*) '    '
           WRITE(10,5)
5          FORMAT(2X,'Buckling Shape =')

           do icol=1, 1+imag0
             do i=1,maxeqn
               FV1(i) = Z(i,locat+imag-icol+1)
             end do
             call recda1(FV1,nhing1,mhing1,hing1,idofh1,NODBC,ntnode)
             CALL upxnod(ntnode,nelm,MELM,NODBC,xnodel,FV1)
             CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
             do i=1,maxeqn
               Z(i,locat+imag-icol+1)=FV1(i)
             end do
           end do
           do icol=1, 1+imag0
             WRITE(10,6) (Z(i,locat+imag-icol+1), i=1,maxeqn)
```

```
          end do

 6        FORMAT(2X,5E20.10)
          write(10,*) '    '
          WRITE(10,7) IERR
 7        FORMAT(2X,'IERR = ',I8/)
          WRITE(10,*) '   '
          WRITE(10,*) 'Note: only if IERR = 0, '
          WRITE(10,*) '      is eigen analysis successful!!'
        END DO

              return
            end if
          end if
          end if


        end if

C
C    GET AND PRINT THE NEW NODAL POSITION VECTOR XYZN
C
c        if(npr.eq.1) then
          write(6,*) 'load level NLD=', nld, '     alambd=',alambd
          CALL NXYZ(NTNODE,NELM,MELM,XNODEL,XYZN,alambd,itr)
          CALL PRDISP(NTNODE,NODBC,atotl,nodepr,alambd)

          npr=0
c        end if


        do ii = 1, maxeqn
         atotl0(ii) = atotl(ii)
        end do


        call tempsv(maxeqn,nelm,
     &              xnodel,ttrg,omegag,da,dq,
     &              sxnode,sttrg,somegg,sda,sdq)

        call tempsv(maxeqn,nelm,
     &              xnodel,t6rg,om6gag,da,dq,
     &              sxnode,st6rg,som6gg,sda,sdq)

        alamb0 = alambd
        ilast=niter
        ldown=0
        go to 64        ! go to new load loop
C
C
C
66      CONTINUE
C
C     CONVERGENCE NOT ACHIEVED. REDUCE THE LOAD STEP BY HALF.
C

        ldown=ldown+1
```

```
      alambd=alamb0
      alinc0=0.5*alinc0

      idet = idet0
      write(6,*) 'Load level is reduced to alambd= ', alambd

      call tempgt(maxeqn,nelm,
     &            xnodel,ttrg,omegag,da,dq,
     &            sxnode,sttrg,somegg,sda,sdq)
      call tempgt(maxeqn,nelm,
     &            xnodel,t6rg,om6gag,da,dq,
     &            sxnode,st6rg,som6gg,sda,sdq)
      do ii = 1, maxeqn
       atotl(ii) = atotl0(ii)
      end do
      call thetah(atotl,nhing1,mhing1,hing1,idofh1,NODBC,NTNODE)

      IF(ldown.GT.maxldn) then
        write(6,*) 'idown = ', ldown
       stop 'too small a load step!! Limit point maybe here.'
      end if

      GO TO 18

      END

      subroutine praskd(ast,maxeqn)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Ast(maxeqn,maxeqn)

      write(22,*) 'ast2=',(ast(ii,ii), ii=1, maxeqn)

      return
      end
```

```
C  eigenrgrgg.f:
C
      SUBROUTINE RG(NM,N,A,WR,WI,MATZ,Z,IV1,FV1,IERR)
C
      INTEGER N,NM,IS1,IS2,IERR,MATZ
      DOUBLE PRECISION A(NM,N),WR(N),WI(N),Z(NM,N),FV1(N)
      INTEGER IV1(N)
C
C     THIS SUBROUTINE CALLS THE RECOMMENDED SEQUENCE OF
C     SUBROUTINES FROM THE EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)
C     TO FIND THE EIGENVALUES AND EIGENVECTORS (IF DESIRED)
C     OF A REAL GENERAL MATRIX.
C
C     ON INPUT
C
C        NM  MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL
C        ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C        DIMENSION STATEMENT.
C
C        N  IS THE ORDER OF THE MATRIX  A.
C
C        A  CONTAINS THE REAL GENERAL MATRIX.
C
C        MATZ  IS AN INTEGER VARIABLE SET EQUAL TO ZERO IF
C        ONLY EIGENVALUES ARE DESIRED.  OTHERWISE IT IS SET TO
C        ANY NON-ZERO INTEGER FOR BOTH EIGENVALUES AND EIGENVECTORS.
C
C     ON OUTPUT
C
C        WR  AND  WI  CONTAIN THE REAL AND IMAGINARY PARTS,
C        RESPECTIVELY, OF THE EIGENVALUES.  COMPLEX CONJUGATE
C        PAIRS OF EIGENVALUES APPEAR CONSECUTIVELY WITH THE
C        EIGENVALUE HAVING THE POSITIVE IMAGINARY PART FIRST.
C
C        Z  CONTAINS THE REAL AND IMAGINARY PARTS OF THE EIGENVECTORS
C        IF MATZ IS NOT ZERO.  IF THE J-TH EIGENVALUE IS REAL, THE
C        J-TH COLUMN OF  Z  CONTAINS ITS EIGENVECTOR.  IF THE J-TH
C        EIGENVALUE IS COMPLEX WITH POSITIVE IMAGINARY PART, THE
C        J-TH AND (J+1)-TH COLUMNS OF  Z  CONTAIN THE REAL AND
C        IMAGINARY PARTS OF ITS EIGENVECTOR.  THE CONJUGATE OF THIS
C        VECTOR IS THE EIGENVECTOR FOR THE CONJUGATE EIGENVALUE.
C
C        IERR  IS AN INTEGER OUTPUT VARIABLE SET EQUAL TO AN ERROR
C           COMPLETION CODE DESCRIBED IN THE DOCUMENTATION FOR HQR
C           AND HQR2.  THE NORMAL COMPLETION CODE IS ZERO.
C
C        IV1  AND  FV1  ARE TEMPORARY STORAGE ARRAYS.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      IF (N .LE. NM) GO TO 10
```

```
      IERR = 10 * N
      GO TO 50
C
   10 CALL  BALANC(NM,N,A,IS1,IS2,FV1)
      CALL  ELMHES(NM,N,IS1,IS2,A,IV1)
      IF (MATZ .NE. 0) GO TO 20
C     ..... FIND EIGENVALUES ONLY .....
      CALL  HQR(NM,N,IS1,IS2,A,WR,WI,IERR)
      GO TO 50
C     ..... FIND BOTH EIGENVALUES AND EIGENVECTORS .....
   20 CALL  ELTRAN(NM,N,IS1,IS2,A,IV1,Z)
      CALL  HQR2(NM,N,IS1,IS2,A,WR,WI,Z,IERR)
      IF (IERR .NE. 0) GO TO 50
      CALL  BALBAK(NM,N,IS1,IS2,FV1,N,Z)
   50 RETURN
      END

      SUBROUTINE BALANC(NM,N,A,LOW,IGH,SCALE)
C
      INTEGER I,J,K,L,M,N,JJ,NM,IGH,LOW,IEXC
      DOUBLE PRECISION A(NM,N),SCALE(N)
      DOUBLE PRECISION C,F,G,R,S,B2,RADIX
      LOGICAL NOCONV
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE BALANCE,
C     NUM. MATH. 13, 293-304(1969) BY PARLETT AND REINSCH.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 315-326(1971).
C
C     THIS SUBROUTINE BALANCES A REAL MATRIX AND ISOLATES
C     EIGENVALUES WHENEVER POSSIBLE.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        A CONTAINS THE INPUT MATRIX TO BE BALANCED.
C
C     ON OUTPUT
C
C        A CONTAINS THE BALANCED MATRIX.
C
C        LOW AND IGH ARE TWO INTEGERS SUCH THAT A(I,J)
C          IS EQUAL TO ZERO IF
C           (1) I IS GREATER THAN J AND
C           (2) J=1,...,LOW-1 OR I=IGH+1,...,N.
C
C        SCALE CONTAINS INFORMATION DETERMINING THE
C           PERMUTATIONS AND SCALING FACTORS USED.
C
C     SUPPOSE THAT THE PRINCIPAL SUBMATRIX IN ROWS LOW THROUGH IGH
C     HAS BEEN BALANCED, THAT P(J) DENOTES THE INDEX INTERCHANGED
C     WITH J DURING THE PERMUTATION STEP, AND THAT THE ELEMENTS
C     OF THE DIAGONAL MATRIX USED ARE DENOTED BY D(I,J).  THEN
```

```
C        SCALE(J) = P(J),    FOR J = 1,...,LOW-1
C                 = D(J,J),      J = LOW,...,IGH
C                 = P(J)         J = IGH+1,...,N.
C     THE ORDER IN WHICH THE INTERCHANGES ARE MADE IS N TO IGH+1,
C     THEN 1 TO LOW-1.
C
C     NOTE THAT 1 IS RETURNED FOR IGH IF IGH IS ZERO FORMALLY.
C
C     THE ALGOL PROCEDURE EXC CONTAINED IN BALANCE APPEARS IN
C     BALANC  IN LINE.  (NOTE THAT THE ALGOL ROLES OF IDENTIFIERS
C     K,L HAVE BEEN REVERSED.)
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      RADIX = 16.0D0
C
      B2 = RADIX * RADIX
      K = 1
      L = N
      GO TO 100
C     ..... IN-LINE PROCEDURE FOR ROW AND
C                 COLUMN EXCHANGE .....
   20 SCALE(M) = J
      IF (J .EQ. M) GO TO 50
C
      DO 30 I = 1, L
         F = A(I,J)
         A(I,J) = A(I,M)
         A(I,M) = F
   30 CONTINUE
C
      DO 40 I = K, N
         F = A(J,I)
         A(J,I) = A(M,I)
         A(M,I) = F
   40 CONTINUE
C
   50 GO TO (80,130), IEXC
C     ..... SEARCH FOR ROWS ISOLATING AN EIGENVALUE
C                 AND PUSH THEM DOWN .....
   80 IF (L .EQ. 1) GO TO 280
      L = L - 1
C     ..... FOR J=L STEP -1 UNTIL 1 DO -- .....
  100 DO 120 JJ = 1, L
         J = L + 1 - JJ
C
         DO 110 I = 1, L
            IF (I .EQ. J) GO TO 110
            IF (A(J,I) .NE. 0.0D0) GO TO 120
  110    CONTINUE
C
```

```
         M = L
         IEXC = 1
         GO TO 20
  120 CONTINUE
C
      GO TO 140
C     ......... SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE
C                AND PUSH THEM LEFT .........
  130 K = K + 1
C
  140 DO 170 J = K, L
C
         DO 150 I = K, L
            IF (I .EQ. J) GO TO 150
            IF (A(I,J) .NE. 0.0D0) GO TO 170
  150    CONTINUE
C
         M = K
         IEXC = 2
         GO TO 20
  170 CONTINUE
C     ......... NOW BALANCE THE SUBMATRIX IN ROWS K TO L .........
      DO 180 I = K, L
  180 SCALE(I) = 1.0D0
C     ......... ITERATIVE LOOP FOR NORM REDUCTION .........
  190 NOCONV = .FALSE.
C
      DO 270 I = K, L
         C = 0.0D0
         R = 0.0D0
C
         DO 200 J = K, L
            IF (J .EQ. I) GO TO 200
            C = C + DABS(A(J,I))
            R = R + DABS(A(I,J))
  200    CONTINUE
C     ......... GUARD AGAINST ZERO C OR R DUE TO UNDERFLOW .........
         IF (C .EQ. 0.0D0 .OR. R .EQ. 0.0D0) GO TO 270
         G = R / RADIX
         F = 1.0D0
         S = C + R
  210    IF (C .GE. G) GO TO 220
         F = F * RADIX
         C = C * B2
         GO TO 210
  220    G = R * RADIX
  230    IF (C .LT. G) GO TO 240
         F = F / RADIX
         C = C / B2
         GO TO 230
C     ......... NOW BALANCE .........
  240    IF ((C + R) / F .GE. 0.95D0 * S) GO TO 270
         G = 1.0D0 / F
         SCALE(I) = SCALE(I) * F
         NOCONV = .TRUE.
C
         DO 250 J = K, N
```

```
  250    A(I,J) = A(I,J) * G
C
         DO 260 J = 1, L
  260    A(J,I) = A(J,I) * F
C
  270 CONTINUE
C
      IF (NOCONV) GO TO 190
C
  280 LOW = K
      IGH = L
      RETURN
      END

      SUBROUTINE ELMHES(NM,N,LOW,IGH,A,INT)
C
      INTEGER I,J,M,N,LA,NM,IGH,KP1,LOW,MM1,MP1
      DOUBLE PRECISION A(NM,N)
      DOUBLE PRECISION X,Y
      INTEGER INT(IGH)
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE ELMHES,
C     NUM. MATH. 12, 349-368(1968) BY MARTIN AND WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 339-358(1971).
C
C     GIVEN A REAL GENERAL MATRIX, THIS SUBROUTINE
C     REDUCES A SUBMATRIX SITUATED IN ROWS AND COLUMNS
C     LOW THROUGH IGH TO UPPER HESSENBERG FORM BY
C     STABILIZED ELEMENTARY SIMILARITY TRANSFORMATIONS.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        LOW AND IGH ARE INTEGERS DETERMINED BY THE BALANCING
C          SUBROUTINE  BALANC.  IF  BALANC  HAS NOT BEEN USED,
C          SET LOW=1, IGH=N.
C
C        A CONTAINS THE INPUT MATRIX.
C
C     ON OUTPUT
C
C        A CONTAINS THE HESSENBERG MATRIX.  THE MULTIPLIERS
C          WHICH WERE USED IN THE REDUCTION ARE STORED IN THE
C          REMAINING TRIANGLE UNDER THE HESSENBERG MATRIX.
C
C        INT CONTAINS INFORMATION ON THE ROWS AND COLUMNS
C          INTERCHANGED IN THE REDUCTION.
C          ONLY ELEMENTS LOW THROUGH IGH ARE USED.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
```

```
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C
      DO 180 M = KP1, LA
         MM1 = M - 1
         X = 0.0D0
         I = M
C
         DO 100 J = M, IGH
            IF (DABS(A(J,MM1)) .LE. DABS(X)) GO TO 100
            X = A(J,MM1)
            I = J
  100    CONTINUE
C
         INT(M) = I
         IF (I .EQ. M) GO TO 130
C     .......... INTERCHANGE ROWS AND COLUMNS OF A ..........
         DO 110 J = MM1, N
            Y = A(I,J)
            A(I,J) = A(M,J)
            A(M,J) = Y
  110    CONTINUE
C
         DO 120 J = 1, IGH
            Y = A(J,I)
            A(J,I) = A(J,M)
            A(J,M) = Y
  120    CONTINUE
C     .......... END INTERCHANGE ..........
  130    IF (X .EQ. 0.0D0) GO TO 180
         MP1 = M + 1
C
         DO 160 I = MP1, IGH
            Y = A(I,MM1)
            IF (Y .EQ. 0.0D0) GO TO 160
            Y = Y / X
            A(I,MM1) = Y
C
            DO 140 J = M, N
  140       A(I,J) = A(I,J) - Y * A(M,J)
C
            DO 150 J = 1, IGH
  150       A(J,M) = A(J,M) + Y * A(J,I)
C
  160    CONTINUE
C
  180 CONTINUE
C
  200 RETURN
      END
      SUBROUTINE ELTRAN(NM,N,LOW,IGH,A,INT,Z)
```

```
C
      INTEGER I,J,N,KL,MM,MP,NM,IGH,LOW,MP1
      DOUBLE PRECISION A(NM,IGH),Z(NM,N)
      INTEGER INT(IGH)
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE ELMTRANS,
C     NUM. MATH. 16, 181-204(1970) BY PETERS AND WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 372-395(1971).
C
C     THIS SUBROUTINE ACCUMULATES THE STABILIZED ELEMENTARY
C     SIMILARITY TRANSFORMATIONS USED IN THE REDUCTION OF A
C     REAL GENERAL MATRIX TO UPPER HESSENBERG FORM BY  ELMHES.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        LOW AND IGH ARE INTEGERS DETERMINED BY THE BALANCING
C          SUBROUTINE  BALANC.  IF  BALANC  HAS NOT BEEN USED,
C          SET LOW=1, IGH=N.
C
C        A CONTAINS THE MULTIPLIERS WHICH WERE USED IN THE
C          REDUCTION BY  ELMHES  IN ITS LOWER TRIANGLE
C          BELOW THE SUBDIAGONAL.
C
C        INT CONTAINS INFORMATION ON THE ROWS AND COLUMNS
C          INTERCHANGED IN THE REDUCTION BY  ELMHES.
C          ONLY ELEMENTS LOW THROUGH IGH ARE USED.
C
C     ON OUTPUT
C
C        Z CONTAINS THE TRANSFORMATION MATRIX PRODUCED IN THE
C          REDUCTION BY  ELMHES.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
C     .......... INITIALIZE Z TO IDENTITY MATRIX ..........
      DO 80 J = 1, N
C
         DO 60 I = 1, N
   60    Z(I,J) = 0.0D0
C
         Z(J,J) = 1.0D0
   80 CONTINUE
C
      KL = IGH - LOW - 1
      IF (KL .LT. 1) GO TO 200
```

```
C     ......... FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- .........
      DO 140 MM = 1, KL
         MP = IGH - MM
         MP1 = MP + 1
C
         DO 100 I = MP1, IGH
  100    Z(I,MP) = A(I,MP-1)
C
         I = INT(MP)
         IF (I .EQ. MP) GO TO 140
C
         DO 130 J = MP, IGH
            Z(MP,J) = Z(I,J)
            Z(I,J) = 0.0D0
  130    CONTINUE
C
         Z(I,MP) = 1.0D0
  140 CONTINUE
C
  200 RETURN
      END

      SUBROUTINE HQR(NM,N,LOW,IGH,H,WR,WI,IERR)
C
      INTEGER I,J,K,L,M,N,EN,LL,MM,NA,NM,IGH,ITN,ITS,LOW,MP2,ENM2,IERR
      DOUBLE PRECISION H(NM,N),WR(N),WI(N)
      DOUBLE PRECISION P,Q,R,S,T,W,X,Y,ZZ,NORM,TST1,TST2
      LOGICAL NOTLAS
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE HQR,
C     NUM. MATH. 14, 219-231(1970) BY MARTIN, PETERS, AND WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 359-371(1971).
C
C     THIS SUBROUTINE FINDS THE EIGENVALUES OF A REAL
C     UPPER HESSENBERG MATRIX BY THE QR METHOD.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        LOW AND IGH ARE INTEGERS DETERMINED BY THE BALANCING
C          SUBROUTINE  BALANC.  IF  BALANC  HAS NOT BEEN USED,
C          SET LOW=1, IGH=N.
C
C        H CONTAINS THE UPPER HESSENBERG MATRIX.  INFORMATION ABOUT
C          THE TRANSFORMATIONS USED IN THE REDUCTION TO HESSENBERG
C          FORM BY  ELMHES  OR  ORTHES, IF PERFORMED, IS STORED
C          IN THE REMAINING TRIANGLE UNDER THE HESSENBERG MATRIX.
C
C     ON OUTPUT
C
C        H HAS BEEN DESTROYED.  THEREFORE, IT MUST BE SAVED
C          BEFORE CALLING  HQR  IF SUBSEQUENT CALCULATION AND
```

```
C          BACK TRANSFORMATION OF EIGENVECTORS IS TO BE PERFORMED.
C
C        WR AND WI CONTAIN THE REAL AND IMAGINARY PARTS,
C          RESPECTIVELY, OF THE EIGENVALUES.  THE EIGENVALUES
C          ARE UNORDERED EXCEPT THAT COMPLEX CONJUGATE PAIRS
C          OF VALUES APPEAR CONSECUTIVELY WITH THE EIGENVALUE
C          HAVING THE POSITIVE IMAGINARY PART FIRST.  IF AN
C          ERROR EXIT IS MADE, THE EIGENVALUES SHOULD BE CORRECT
C          FOR INDICES IERR+1,...,N.
C
C        IERR IS SET TO
C          ZERO       FOR NORMAL RETURN,
C          J          IF THE LIMIT OF 30*N ITERATIONS IS EXHAUSTED
C                     WHILE THE J-TH EIGENVALUE IS BEING SOUGHT.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      IERR = 0
      NORM = 0.0D0
      K = 1
C     ..... STORE ROOTS ISOLATED BY BALANC
C                AND COMPUTE MATRIX NORM .....
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    NORM = NORM + DABS(H(I,J))
C
         K = I
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
         WR(I) = H(I,I)
         WI(I) = 0.0D0
   50 CONTINUE
C
      EN = IGH
      T = 0.0D0
      ITN = 30*N
C     ..... SEARCH FOR NEXT EIGENVALUES .....
   60 IF (EN .LT. LOW) GO TO 1001
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C     ..... LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- .....
   70 DO 80 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 100
         S = DABS(H(L-1,L-1)) + DABS(H(L,L))
         IF (S .EQ. 0.0D0) S = NORM
         TST1 = S
         TST2 = TST1 + DABS(H(L,L-1))
         IF (TST2 .EQ. TST1) GO TO 100
```

```
   80 CONTINUE
C     ..........  FORM SHIFT  ..........
  100 X = H(EN,EN)
      IF (L .EQ. EN) GO TO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L .EQ. NA) GO TO 280
      IF (ITN .EQ. 0) GO TO 1000
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C     ..........  FORM EXCEPTIONAL SHIFT  ..........
      T = T + X
C
      DO 120 I = LOW, EN
  120 H(I,I) = H(I,I) - X
C
      S = DABS(H(EN,NA)) + DABS(H(NA,ENM2))
      X = 0.75D0 * S
      Y = X
      W = -0.4375D0 * S * S
  130 ITS = ITS + 1
      ITN = ITN - 1
C     ..........  LOOK FOR TWO CONSECUTIVE SMALL
C                 SUB-DIAGONAL ELEMENTS.
C                 FOR M=EN-2 STEP -1 UNTIL L DO --  ..........
      DO 140 MM = L, ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
         S = Y - ZZ
         P = (R * S - W) / H(M+1,M) + H(M,M+1)
         Q = H(M+1,M+1) - ZZ - R - S
         R = H(M+2,M+1)
         S = DABS(P) + DABS(Q) + DABS(R)
         P = P / S
         Q = Q / S
         R = R / S
         IF (M .EQ. L) GO TO 150
         TST1 = DABS(P)*(DABS(H(M-1,M-1)) + DABS(ZZ) +
DABS(H(M+1,M+1)))
         TST2 = TST1 + DABS(H(M,M-1))*(DABS(Q) + DABS(R))
         IF (TST2 .EQ. TST1) GO TO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2, EN
         H(I,I-2) = 0.0D0
         IF (I .EQ. MP2) GO TO 160
         H(I,I-3) = 0.0D0
  160 CONTINUE
C     ..........  DOUBLE QR STEP INVOLVING ROWS L TO EN AND
C                 COLUMNS M TO EN  ..........
      DO 260 K = M, NA
         NOTLAS = K .NE. NA
         IF (K .EQ. M) GO TO 170
         P = H(K,K-1)
         Q = H(K+1,K-1)
```

```
         R = 0.0D0
         IF (NOTLAS) R = H(K+2,K-1)
         X = DABS(P) + DABS(Q) + DABS(R)
         IF (X .EQ. 0.0D0) GO TO 260
         P = P / X
         Q = Q / X
         R = R / X
  170    S = DSIGN(DSQRT(P*P+Q*Q+R*R),P)
         IF (K .EQ. M) GO TO 180
         H(K,K-1) = -S * X
         GO TO 190
  180    IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190    P = P + S
         X = P / S
         Y = Q / S
         ZZ = R / S
         Q = Q / P
         R = R / P
         IF (NOTLAS) GO TO 225
C     .......... ROW MODIFICATION ..........
         DO 200 J = K, N
            P = H(K,J) + Q * H(K+1,J)
            H(K,J) = H(K,J) - P * X
            H(K+1,J) = H(K+1,J) - P * Y
  200    CONTINUE
C
         J = MIN0(EN,K+3)
C     .......... COLUMN MODIFICATION ..........
         DO 210 I = 1, J
            P = X * H(I,K) + Y * H(I,K+1)
            H(I,K) = H(I,K) - P
            H(I,K+1) = H(I,K+1) - P * Q
  210    CONTINUE
         GO TO 255
  225    CONTINUE
C     .......... ROW MODIFICATION ..........
         DO 230 J = K, N
            P = H(K,J) + Q * H(K+1,J) + R * H(K+2,J)
            H(K,J) = H(K,J) - P * X
            H(K+1,J) = H(K+1,J) - P * Y
            H(K+2,J) = H(K+2,J) - P * ZZ
  230    CONTINUE
C
         J = MIN0(EN,K+3)
C     .......... COLUMN MODIFICATION ..........
         DO 240 I = 1, J
            P = X * H(I,K) + Y * H(I,K+1) + ZZ * H(I,K+2)
            H(I,K) = H(I,K) - P
            H(I,K+1) = H(I,K+1) - P * Q
            H(I,K+2) = H(I,K+2) - P * R
  240    CONTINUE
  255    CONTINUE
C
  260 CONTINUE
C
       GO TO 70
C     .......... ONE ROOT FOUND ..........
```

```
  270 WR(EN) = X + T
      WI(EN) = 0.0D0
      EN = NA
      GO TO 60
C     ..... TWO ROOTS FOUND .....
  280 P = (Y - X) / 2.0D0
      Q = P * P + W
      ZZ = DSQRT(DABS(Q))
      X = X + T
      IF (Q .LT. 0.0D0) GO TO 320
C     ..... REAL PAIR .....
      ZZ = P + DSIGN(ZZ,P)
      WR(NA) = X + ZZ
      WR(EN) = WR(NA)
      IF (ZZ .NE. 0.0D0) WR(EN) = X - W / ZZ
      WI(NA) = 0.0D0
      WI(EN) = 0.0D0
      GO TO 330
C     ..... COMPLEX PAIR .....
  320 WR(NA) = X + P
      WR(EN) = X + P
      WI(NA) = ZZ
      WI(EN) = -ZZ
  330 EN = ENM2
      GO TO 60
C     ..... SET ERROR -- ALL EIGENVALUES HAVE NOT
C                 CONVERGED AFTER 30*N ITERATIONS .....
 1000 IERR = EN
 1001 RETURN
      END
      SUBROUTINE HQR2(NM,N,LOW,IGH,H,WR,WI,Z,IERR)
C
      INTEGER I,J,K,L,M,N,EN,II,JJ,LL,MM,NA,NM,NN,
     X        IGH,ITN,ITS,LOW,MP2,ENM2,IERR
      DOUBLE PRECISION H(NM,N),WR(N),WI(N),Z(NM,N)
      DOUBLE PRECISION P,Q,R,S,T,W,X,Y,RA,SA,VI,VR,ZZ,NORM,TST1,TST2
      LOGICAL NOTLAS
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE HQR2,
C     NUM. MATH. 16, 181-204(1970) BY PETERS AND WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 372-395(1971).
C
C     THIS SUBROUTINE FINDS THE EIGENVALUES AND EIGENVECTORS
C     OF A REAL UPPER HESSENBERG MATRIX BY THE QR METHOD.  THE
C     EIGENVECTORS OF A REAL GENERAL MATRIX CAN ALSO BE FOUND
C     IF  ELMHES  AND  ELTRAN  OR  ORTHES  AND  ORTRAN  HAVE
C     BEEN USED TO REDUCE THIS GENERAL MATRIX TO HESSENBERG FORM
C     AND TO ACCUMULATE THE SIMILARITY TRANSFORMATIONS.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
```

```
C        LOW AND IGH ARE INTEGERS DETERMINED BY THE BALANCING
C          SUBROUTINE  BALANC.  IF  BALANC  HAS NOT BEEN USED,
C          SET LOW=1, IGH=N.
C
C        H CONTAINS THE UPPER HESSENBERG MATRIX.
C
C        Z CONTAINS THE TRANSFORMATION MATRIX PRODUCED BY  ELTRAN
C          AFTER THE REDUCTION BY  ELMHES, OR BY  ORTRAN  AFTER THE
C          REDUCTION BY  ORTHES, IF PERFORMED.  IF THE EIGENVECTORS
C          OF THE HESSENBERG MATRIX ARE DESIRED, Z MUST CONTAIN THE
C          IDENTITY MATRIX.
C
C     ON OUTPUT
C
C        H HAS BEEN DESTROYED.
C
C        WR AND WI CONTAIN THE REAL AND IMAGINARY PARTS,
C          RESPECTIVELY, OF THE EIGENVALUES.  THE EIGENVALUES
C          ARE UNORDERED EXCEPT THAT COMPLEX CONJUGATE PAIRS
C          OF VALUES APPEAR CONSECUTIVELY WITH THE EIGENVALUE
C          HAVING THE POSITIVE IMAGINARY PART FIRST.  IF AN
C          ERROR EXIT IS MADE, THE EIGENVALUES SHOULD BE CORRECT
C          FOR INDICES IERR+1,...,N.
C
C        Z CONTAINS THE REAL AND IMAGINARY PARTS OF THE EIGENVECTORS.
C          IF THE I-TH EIGENVALUE IS REAL, THE I-TH COLUMN OF Z
C          CONTAINS ITS EIGENVECTOR.  IF THE I-TH EIGENVALUE IS COMPLEX
C          WITH POSITIVE IMAGINARY PART, THE I-TH AND (I+1)-TH
C          COLUMNS OF Z CONTAIN THE REAL AND IMAGINARY PARTS OF ITS
C          EIGENVECTOR.  THE EIGENVECTORS ARE UNNORMALIZED.  IF AN
C          ERROR EXIT IS MADE, NONE OF THE EIGENVECTORS HAS BEEN FOUND.
C
C        IERR IS SET TO
C          ZERO       FOR NORMAL RETURN,
C          J          IF THE LIMIT OF 30*N ITERATIONS IS EXHAUSTED
C                     WHILE THE J-TH EIGENVALUE IS BEING SOUGHT.
C
C     CALLS CDIV FOR COMPLEX DIVISION.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      IERR = 0
      NORM = 0.0D0
      K = 1
C     .......... STORE ROOTS ISOLATED BY BALANC
C                AND COMPUTE MATRIX NORM ..........
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    NORM = NORM + DABS(H(I,J))
C
```

```
         K = I
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
         WR(I) = H(I,I)
         WI(I) = 0.0D0
   50 CONTINUE
C
      EN = IGH
      T = 0.0D0
      ITN = 30*N
C     .......... SEARCH FOR NEXT EIGENVALUES ..........
   60 IF (EN .LT. LOW) GO TO 340
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C     .......... LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- ..........
   70 DO 80 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 100
         S = DABS(H(L-1,L-1)) + DABS(H(L,L))
         IF (S .EQ. 0.0D0) S = NORM
         TST1 = S
         TST2 = TST1 + DABS(H(L,L-1))
         IF (TST2 .EQ. TST1) GO TO 100
   80 CONTINUE
C     .......... FORM SHIFT ..........
  100 X = H(EN,EN)
      IF (L .EQ. EN) GO TO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L .EQ. NA) GO TO 280
      IF (ITN .EQ. 0) GO TO 1000
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C     .......... FORM EXCEPTIONAL SHIFT ..........
      T = T + X
C
      DO 120 I = LOW, EN
  120 H(I,I) = H(I,I) - X
C
      S = DABS(H(EN,NA)) + DABS(H(NA,ENM2))
      X = 0.75D0 * S
      Y = X
      W = -0.4375D0 * S * S
  130 ITS = ITS + 1
      ITN = ITN - 1
C     .......... LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS.
C                FOR M=EN-2 STEP -1 UNTIL L DO -- ..........
      DO 140 MM = L, ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
         S = Y - ZZ
         P = (R * S - W) / H(M+1,M) + H(M,M+1)
         Q = H(M+1,M+1) - ZZ - R - S
         R = H(M+2,M+1)
         S = DABS(P) + DABS(Q) + DABS(R)
```

```
         P = P / S
         Q = Q / S
         R = R / S
         IF (M .EQ. L) GO TO 150
         TST1 = DABS(P)*(DABS(H(M-1,M-1)) + DABS(ZZ) +
DABS(H(M+1,M+1)))
         TST2 = TST1 + DABS(H(M,M-1))*(DABS(Q) + DABS(R))
         IF (TST2 .EQ. TST1) GO TO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2, EN
         H(I,I-2) = 0.0D0
         IF (I .EQ. MP2) GO TO 160
         H(I,I-3) = 0.0D0
  160 CONTINUE
C     .......... DOUBLE QR STEP INVOLVING ROWS L TO EN AND
C                COLUMNS M TO EN ..........
      DO 260 K = M, NA
         NOTLAS = K .NE. NA
         IF (K .EQ. M) GO TO 170
         P = H(K,K-1)
         Q = H(K+1,K-1)
         R = 0.0D0
         IF (NOTLAS) R = H(K+2,K-1)
         X = DABS(P) + DABS(Q) + DABS(R)
         IF (X .EQ. 0.0D0) GO TO 260
         P = P / X
         Q = Q / X
         R = R / X
  170    S = DSIGN(DSQRT(P*P+Q*Q+R*R),P)
         IF (K .EQ. M) GO TO 180
         H(K,K-1) = -S * X
         GO TO 190
  180    IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190    P = P + S
         X = P / S
         Y = Q / S
         ZZ = R / S
         Q = Q / P
         R = R / P
         IF (NOTLAS) GO TO 225
C     .......... ROW MODIFICATION ..........
         DO 200 J = K, N
            P = H(K,J) + Q * H(K+1,J)
            H(K,J) = H(K,J) - P * X
            H(K+1,J) = H(K+1,J) - P * Y
  200    CONTINUE
C
         J = MIN0(EN,K+3)
C     .......... COLUMN MODIFICATION ..........
         DO 210 I = 1, J
            P = X * H(I,K) + Y * H(I,K+1)
            H(I,K) = H(I,K) - P
            H(I,K+1) = H(I,K+1) - P * Q
  210    CONTINUE
```

```
C     .......... ACCUMULATE TRANSFORMATIONS ..........
         DO 220 I = LOW, IGH
            P = X * Z(I,K) + Y * Z(I,K+1)
            Z(I,K) = Z(I,K) - P
            Z(I,K+1) = Z(I,K+1) - P * Q
  220    CONTINUE
         GO TO 255
  225    CONTINUE
C     .......... ROW MODIFICATION ..........
         DO 230 J = K, N
            P = H(K,J) + Q * H(K+1,J) + R * H(K+2,J)
            H(K,J) = H(K,J) - P * X
            H(K+1,J) = H(K+1,J) - P * Y
            H(K+2,J) = H(K+2,J) - P * ZZ
  230    CONTINUE
C
         J = MIN0(EN,K+3)
C     .......... COLUMN MODIFICATION ..........
         DO 240 I = 1, J
            P = X * H(I,K) + Y * H(I,K+1) + ZZ * H(I,K+2)
            H(I,K) = H(I,K) - P
            H(I,K+1) = H(I,K+1) - P * Q
            H(I,K+2) = H(I,K+2) - P * R
  240    CONTINUE
C     .......... ACCUMULATE TRANSFORMATIONS ..........
         DO 250 I = LOW, IGH
            P = X * Z(I,K) + Y * Z(I,K+1) + ZZ * Z(I,K+2)
            Z(I,K) = Z(I,K) - P
            Z(I,K+1) = Z(I,K+1) - P * Q
            Z(I,K+2) = Z(I,K+2) - P * R
  250    CONTINUE
  255    CONTINUE
C
  260 CONTINUE
C
      GO TO 70
C     .......... ONE ROOT FOUND ..........
  270 H(EN,EN) = X + T
      WR(EN) = H(EN,EN)
      WI(EN) = 0.0D0
      EN = NA
      GO TO 60
C     .......... TWO ROOTS FOUND ..........
  280 P = (Y - X) / 2.0D0
      Q = P * P + W
      ZZ = DSQRT(DABS(Q))
      H(EN,EN) = X + T
      X = H(EN,EN)
      H(NA,NA) = Y + T
      IF (Q .LT. 0.0D0) GO TO 320
C     .......... REAL PAIR ..........
      ZZ = P + DSIGN(ZZ,P)
      WR(NA) = X + ZZ
      WR(EN) = WR(NA)
      IF (ZZ .NE. 0.0D0) WR(EN) = X - W / ZZ
      WI(NA) = 0.0D0
      WI(EN) = 0.0D0
```

```
      X = H(EN,NA)
      S = DABS(X) + DABS(ZZ)
      P = X / S
      Q = ZZ / S
      R = DSQRT(P*P+Q*Q)
      P = P / R
      Q = Q / R
C     .......... ROW MODIFICATION ..........
      DO 290 J = NA, N
         ZZ = H(NA,J)
         H(NA,J) = Q * ZZ + P * H(EN,J)
         H(EN,J) = Q * H(EN,J) - P * ZZ
  290 CONTINUE
C     .......... COLUMN MODIFICATION ..........
      DO 300 I = 1, EN
         ZZ = H(I,NA)
         H(I,NA) = Q * ZZ + P * H(I,EN)
         H(I,EN) = Q * H(I,EN) - P * ZZ
  300 CONTINUE
C     .......... ACCUMULATE TRANSFORMATIONS ..........
      DO 310 I = LOW, IGH
         ZZ = Z(I,NA)
         Z(I,NA) = Q * ZZ + P * Z(I,EN)
         Z(I,EN) = Q * Z(I,EN) - P * ZZ
  310 CONTINUE
C
      GO TO 330
C     .......... COMPLEX PAIR ..........
  320 WR(NA) = X + P
      WR(EN) = X + P
      WI(NA) = ZZ
      WI(EN) = -ZZ
  330 EN = ENM2
      GO TO 60
C     .......... ALL ROOTS FOUND.  BACKSUBSTITUTE TO FIND
C                VECTORS OF UPPER TRIANGULAR FORM ..........
  340 IF (NORM .EQ. 0.0D0) GO TO 1001
C     .......... FOR EN=N STEP -1 UNTIL 1 DO -- ..........
      DO 800 NN = 1, N
         EN = N + 1 - NN
         P = WR(EN)
         Q = WI(EN)
         NA = EN - 1
         IF (Q) 710, 600, 800
C     .......... REAL VECTOR ..........
  600    M = EN
         H(EN,EN) = 1.0D0
         IF (NA .EQ. 0) GO TO 800
C     .......... FOR I=EN-1 STEP -1 UNTIL 1 DO -- ..........
         DO 700 II = 1, NA
            I = EN - II
            W = H(I,I) - P
            R = 0.0D0
C
            DO 610 J = M, EN
  610       R = R + H(I,J) * H(J,EN)
C
```

```
            IF (WI(I) .GE. 0.0D0) GO TO 630
            ZZ = W
            S = R
            GO TO 700
  630       M = I
            IF (WI(I) .NE. 0.0D0) GO TO 640
            T = W
            IF (T .NE. 0.0D0) GO TO 635
               TST1 = NORM
               T = TST1
  632          T = 0.01D0 * T
               TST2 = NORM + T
               IF (TST2 .GT. TST1) GO TO 632
  635       H(I,EN) = -R / T
            GO TO 680
C     ..... SOLVE REAL EQUATIONS .....
  640       X = H(I,I+1)
            Y = H(I+1,I)
            Q = (WR(I) - P) * (WR(I) - P) + WI(I) * WI(I)
            T = (X * S - ZZ * R) / Q
            H(I,EN) = T
            IF (DABS(X) .LE. DABS(ZZ)) GO TO 650
            H(I+1,EN) = (-R - W * T) / X
            GO TO 680
  650       H(I+1,EN) = (-S - Y * T) / ZZ
C
C     ..... OVERFLOW CONTROL .....
  680       T = DABS(H(I,EN))
            IF (T .EQ. 0.0D0) GO TO 700
            TST1 = T
            TST2 = TST1 + 1.0D0/TST1
            IF (TST2 .GT. TST1) GO TO 700
            DO 690 J = I, EN
               H(J,EN) = H(J,EN)/T
  690       CONTINUE
C
  700    CONTINUE
C     ..... END REAL VECTOR .....
         GO TO 800
C     ..... COMPLEX VECTOR .....
  710    M = NA
C     ..... LAST VECTOR COMPONENT CHOSEN IMAGINARY SO THAT
C                EIGENVECTOR MATRIX IS TRIANGULAR .....
         IF (DABS(H(EN,NA)) .LE. DABS(H(NA,EN))) GO TO 720
         H(NA,NA) = Q / H(EN,NA)
         H(NA,EN) = -(H(EN,EN) - P) / H(EN,NA)
         GO TO 730
  720    CALL CDIV(0.0D0,-H(NA,EN),H(NA,NA)-P,Q,H(NA,NA),H(NA,EN))
  730    H(EN,NA) = 0.0D0
         H(EN,EN) = 1.0D0
         ENM2 = NA - 1
         IF (ENM2 .EQ. 0) GO TO 800
C     ..... FOR I=EN-2 STEP -1 UNTIL 1 DO -- .....
         DO 795 II = 1, ENM2
            I = NA - II
            W = H(I,I) - P
            RA = 0.0D0
```

```
            SA = 0.0D0
C
            DO 760 J = M, EN
               RA = RA + H(I,J) * H(J,NA)
               SA = SA + H(I,J) * H(J,EN)
  760       CONTINUE
C
            IF (WI(I) .GE. 0.0D0) GO TO 770
            ZZ = W
            R = RA
            S = SA
            GO TO 795
  770       M = I
            IF (WI(I) .NE. 0.0D0) GO TO 780
            CALL CDIV(-RA,-SA,W,Q,H(I,NA),H(I,EN))
            GO TO 790
C     ..... SOLVE COMPLEX EQUATIONS .....
  780       X = H(I,I+1)
            Y = H(I+1,I)
            VR = (WR(I) - P) * (WR(I) - P) + WI(I) * WI(I) - Q * Q
            VI = (WR(I) - P) * 2.0D0 * Q
            IF (VR .NE. 0.0D0 .OR. VI .NE. 0.0D0) GO TO 784
               TST1 = NORM * (DABS(W) + DABS(Q) + DABS(X)
     X                      + DABS(Y) + DABS(ZZ))
               VR = TST1
  783          VR = 0.01D0 * VR
               TST2 = TST1 + VR
               IF (TST2 .GT. TST1) GO TO 783
  784       CALL CDIV(X*R-ZZ*RA+Q*SA,X*S-ZZ*SA-Q*RA,VR,VI,
     X                H(I,NA),H(I,EN))
            IF (DABS(X) .LE. DABS(ZZ) + DABS(Q)) GO TO 785
            H(I+1,NA) = (-RA - W * H(I,NA) + Q * H(I,EN)) / X
            H(I+1,EN) = (-SA - W * H(I,EN) - Q * H(I,NA)) / X
            GO TO 790
  785       CALL CDIV(-R-Y*H(I,NA),-S-Y*H(I,EN),ZZ,Q,
     X                H(I+1,NA),H(I+1,EN))
C
C     ..... OVERFLOW CONTROL .....
  790       T = DMAX1(DABS(H(I,NA)), DABS(H(I,EN)))
            IF (T .EQ. 0.0D0) GO TO 795
            TST1 = T
            TST2 = TST1 + 1.0D0/TST1
            IF (TST2 .GT. TST1) GO TO 795
            DO 792 J = I, EN
               H(J,NA) = H(J,NA)/T
               H(J,EN) = H(J,EN)/T
  792       CONTINUE
C
  795    CONTINUE
C     ..... END COMPLEX VECTOR .....
  800 CONTINUE
C     ..... END BACK SUBSTITUTION.
C                VECTORS OF ISOLATED ROOTS .....
      DO 840 I = 1, N
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
C
         DO 820 J = I, N
```

```
  820    Z(I,J) = H(I,J)
C
  840 CONTINUE
C     .......... MULTIPLY BY TRANSFORMATION MATRIX TO GIVE
C                VECTORS OF ORIGINAL FULL MATRIX.
C                FOR J=N STEP -1 UNTIL LOW DO -- ..........
      DO 880 JJ = LOW, N
         J = N + LOW - JJ
         M = MIN0(J,IGH)
C
         DO 880 I = LOW, IGH
            ZZ = 0.0D0
C
            DO 860 K = LOW, M
  860       ZZ = ZZ + Z(I,K) * H(K,J)
C
            Z(I,J) = ZZ
  880 CONTINUE
C
      GO TO 1001
C     .......... SET ERROR -- ALL EIGENVALUES HAVE NOT
C                CONVERGED AFTER 30*N ITERATIONS ..........
 1000 IERR = EN
 1001 RETURN
      END

      SUBROUTINE BALBAK(NM,N,LOW,IGH,SCALE,M,Z)
C
      INTEGER I,J,K,M,N,II,NM,IGH,LOW
      DOUBLE PRECISION SCALE(N),Z(NM,M)
      DOUBLE PRECISION S
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE BALBAK,
C     NUM. MATH. 13, 293-304(1969) BY PARLETT AND REINSCH.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 315-326(1971).
C
C     THIS SUBROUTINE FORMS THE EIGENVECTORS OF A REAL GENERAL
C     MATRIX BY BACK TRANSFORMING THOSE OF THE CORRESPONDING
C     BALANCED MATRIX DETERMINED BY  BALANC.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        LOW AND IGH ARE INTEGERS DETERMINED BY  BALANC.
C
C        SCALE CONTAINS INFORMATION DETERMINING THE PERMUTATIONS
C          AND SCALING FACTORS USED BY  BALANC.
C
C        M IS THE NUMBER OF COLUMNS OF Z TO BE BACK TRANSFORMED.
C
C        Z CONTAINS THE REAL AND IMAGINARY PARTS OF THE EIGEN-
C          VECTORS TO BE BACK TRANSFORMED IN ITS FIRST M COLUMNS.
```

```
C
C     ON OUTPUT
C
C        Z CONTAINS THE REAL AND IMAGINARY PARTS OF THE
C          TRANSFORMED EIGENVECTORS IN ITS FIRST M COLUMNS.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      IF (M .EQ. 0) GO TO 200
      IF (IGH .EQ. LOW) GO TO 120
C
      DO 110 I = LOW, IGH
         S = SCALE(I)
C     .......... LEFT HAND EIGENVECTORS ARE BACK TRANSFORMED
C                IF THE FOREGOING STATEMENT IS REPLACED BY
C                S=1.0D0/SCALE(I). ..........
         DO 100 J = 1, M
  100    Z(I,J) = Z(I,J) * S
C
  110 CONTINUE
C     ......... FOR I=LOW-1 STEP -1 UNTIL 1,
C               IGH+1 STEP 1 UNTIL N DO -- ..........
  120 DO 140 II = 1, N
         I = II
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 140
         IF (I .LT. LOW) I = LOW - II
         K = SCALE(I)
         IF (K .EQ. I) GO TO 140
C
         DO 130 J = 1, M
            S = Z(I,J)
            Z(I,J) = Z(K,J)
            Z(K,J) = S
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
      SUBROUTINE CDIV(AR,AI,BR,BI,CR,CI)
      DOUBLE PRECISION AR,AI,BR,BI,CR,CI
C
C     COMPLEX DIVISION, (CR,CI) = (AR,AI)/(BR,BI)
C
      DOUBLE PRECISION S,ARS,AIS,BRS,BIS
      S = DABS(BR) + DABS(BI)
      ARS = AR/S
      AIS = AI/S
      BRS = BR/S
      BIS = BI/S
      S = BRS**2 + BIS**2
```

```
      CR = (ARS*BRS + AIS*BIS)/S
      CI = (AIS*BRS - ARS*BIS)/S
      RETURN
      END
      SUBROUTINE CSROOT(XR,XI,YR,YI)
      DOUBLE PRECISION XR,XI,YR,YI
C
C     (YR,YI) = COMPLEX DSQRT(XR,XI)
C     BRANCH CHOSEN SO THAT YR .GE. 0.0 AND SIGN(YI) .EQ. SIGN(XI)
C
      DOUBLE PRECISION S,TR,TI,PYTHAG
      TR = XR
      TI = XI
      S = DSQRT(0.5D0*(PYTHAG(TR,TI) + DABS(TR)))
      IF (TR .GE. 0.0D0) YR = S
      IF (TI .LT. 0.0D0) S = -S
      IF (TR .LE. 0.0D0) YI = S
      IF (TR .LT. 0.0D0) YR = 0.5D0*(TI/YI)
      IF (TR .GT. 0.0D0) YI = 0.5D0*(TI/YR)
      RETURN
      END
      DOUBLE PRECISION FUNCTION EPSLON (X)
      DOUBLE PRECISION X
C
C     ESTIMATE UNIT ROUNDOFF IN QUANTITIES OF SIZE X.
C
      DOUBLE PRECISION A,B,C,EPS
C
C     THIS PROGRAM SHOULD FUNCTION PROPERLY ON ALL SYSTEMS
C     SATISFYING THE FOLLOWING TWO ASSUMPTIONS,
C        1.  THE BASE USED IN REPRESENTING FLOATING POINT
C            NUMBERS IS NOT A POWER OF THREE.
C        2.  THE QUANTITY  A  IN STATEMENT 10 IS REPRESENTED TO
C            THE ACCURACY USED IN FLOATING POINT VARIABLES
C            THAT ARE STORED IN MEMORY.
C     THE STATEMENT NUMBER 10 AND THE GO TO 10 ARE INTENDED TO
C     FORCE OPTIMIZING COMPILERS TO GENERATE CODE SATISFYING
C     ASSUMPTION 2.
C     UNDER THESE ASSUMPTIONS, IT SHOULD BE TRUE THAT,
C            A  IS NOT EXACTLY EQUAL TO FOUR-THIRDS,
C            B  HAS A ZERO FOR ITS LAST BIT OR DIGIT,
C            C  IS NOT EXACTLY EQUAL TO ONE,
C            EPS  MEASURES THE SEPARATION OF 1.0 FROM
C                 THE NEXT LARGER FLOATING POINT NUMBER.
C     THE DEVELOPERS OF EISPACK WOULD APPRECIATE BEING INFORMED
C     ABOUT ANY SYSTEMS WHERE THESE ASSUMPTIONS DO NOT HOLD.
C
C     THIS VERSION DATED 4/6/83.
C
      A = 4.0D0/3.0D0
   10 B = A - 1.0D0
      C = B + B + B
      EPS = DABS(C-1.0D0)
      IF (EPS .EQ. 0.0D0) GO TO 10
      EPSLON = EPS*DABS(X)
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION PYTHAG(A,B)
      DOUBLE PRECISION A,B
C
C     FINDS DSQRT(A**2+B**2) WITHOUT OVERFLOW OR DESTRUCTIVE UNDERFLOW
C
      DOUBLE PRECISION P,R,S,T,U
      P = DMAX1(DABS(A),DABS(B))
      IF (P .EQ. 0.0D0) GO TO 20
      R = (DMIN1(DABS(A),DABS(B))/P)**2
   10 CONTINUE
         T = 4.0D0 + R
         IF (T .EQ. 4.0D0) GO TO 20
         S = R/T
         U = 1.0D0 + 2.0D0*S
         P = U*P
         R = (S/U)**2 * R
      GO TO 10
   20 PYTHAG = P
      RETURN
      END
      SUBROUTINE RGG(NM,N,A,B,ALFR,ALFI,BETA,MATZ,Z,IERR)
C
      INTEGER N,NM,IERR,MATZ
      DOUBLE PRECISION A(NM,N),B(NM,N),ALFR(N),ALFI(N),BETA(N),Z(NM,N)
      LOGICAL TF
C
C     THIS SUBROUTINE CALLS THE RECOMMENDED SEQUENCE OF
C     SUBROUTINES FROM THE EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)
C     TO FIND THE EIGENVALUES AND EIGENVECTORS (IF DESIRED)
C     FOR THE REAL GENERAL GENERALIZED EIGENPROBLEM  AX = (LAMBDA)BX.
C
C     ON INPUT
C
C        NM  MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL
C        ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C        DIMENSION STATEMENT.
C
C        N  IS THE ORDER OF THE MATRICES  A  AND  B.
C
C        A  CONTAINS A REAL GENERAL MATRIX.
C
C        B  CONTAINS A REAL GENERAL MATRIX.
C
C        MATZ  IS AN INTEGER VARIABLE SET EQUAL TO ZERO IF
C        ONLY EIGENVALUES ARE DESIRED.  OTHERWISE IT IS SET TO
C        ANY NON-ZERO INTEGER FOR BOTH EIGENVALUES AND EIGENVECTORS.
C
C     ON OUTPUT
C
C        ALFR  AND  ALFI  CONTAIN THE REAL AND IMAGINARY PARTS,
C        RESPECTIVELY, OF THE NUMERATORS OF THE EIGENVALUES.
C
C        BETA  CONTAINS THE DENOMINATORS OF THE EIGENVALUES,
C        WHICH ARE THUS GIVEN BY THE RATIOS  (ALFR+I*ALFI)/BETA.
C        COMPLEX CONJUGATE PAIRS OF EIGENVALUES APPEAR CONSECUTIVELY
C        WITH THE EIGENVALUE HAVING THE POSITIVE IMAGINARY PART FIRST.
C
```

```
C        Z  CONTAINS THE REAL AND IMAGINARY PARTS OF THE EIGENVECTORS
C        IF MATZ IS NOT ZERO.  IF THE J-TH EIGENVALUE IS REAL, THE
C        J-TH COLUMN OF  Z  CONTAINS ITS EIGENVECTOR.  IF THE J-TH
C        EIGENVALUE IS COMPLEX WITH POSITIVE IMAGINARY PART, THE
C        J-TH AND (J+1)-TH COLUMNS OF  Z  CONTAIN THE REAL AND
C        IMAGINARY PARTS OF ITS EIGENVECTOR.  THE CONJUGATE OF THIS
C        VECTOR IS THE EIGENVECTOR FOR THE CONJUGATE EIGENVALUE.
C
C        IERR  IS AN INTEGER OUTPUT VARIABLE SET EQUAL TO AN ERROR
C           COMPLETION CODE DESCRIBED IN THE DOCUMENTATION FOR QZIT.
C           THE NORMAL COMPLETION CODE IS ZERO.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      IF (N .LE. NM) GO TO 10
      IERR = 10 * N
      GO TO 50
C
   10 IF (MATZ .NE. 0) GO TO 20
C     ..... FIND EIGENVALUES ONLY .....
      TF = .FALSE.
      CALL  QZHES(NM,N,A,B,TF,Z)
      CALL  QZIT(NM,N,A,B,0.0D0,TF,Z,IERR)
      CALL  QZVAL(NM,N,A,B,ALFR,ALFI,BETA,TF,Z)
      GO TO 50
C     ..... FIND BOTH EIGENVALUES AND EIGENVECTORS .....
   20 TF = .TRUE.
      CALL  QZHES(NM,N,A,B,TF,Z)
      CALL  QZIT(NM,N,A,B,0.0D0,TF,Z,IERR)
      CALL  QZVAL(NM,N,A,B,ALFR,ALFI,BETA,TF,Z)
      IF (IERR .NE. 0) GO TO 50
      CALL  QZVEC(NM,N,A,B,ALFR,ALFI,BETA,Z)
   50 RETURN
      END

      SUBROUTINE QZHES(NM,N,A,B,MATZ,Z)
C
      INTEGER I,J,K,L,N,LB,L1,NM,NK1,NM1,NM2
      DOUBLE PRECISION A(NM,N),B(NM,N),Z(NM,N)
      DOUBLE PRECISION R,S,T,U1,U2,V1,V2,RHO
      LOGICAL MATZ
C
C     THIS SUBROUTINE IS THE FIRST STEP OF THE QZ ALGORITHM
C     FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS,
C     SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART.
C
C     THIS SUBROUTINE ACCEPTS A PAIR OF REAL GENERAL MATRICES AND
C     REDUCES ONE OF THEM TO UPPER HESSENBERG FORM AND THE OTHER
C     TO UPPER TRIANGULAR FORM USING ORTHOGONAL TRANSFORMATIONS.
C     IT IS USUALLY FOLLOWED BY  QZIT,  QZVAL  AND, POSSIBLY,  QZVEC.
C
```

```
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRICES.
C
C        A CONTAINS A REAL GENERAL MATRIX.
C
C        B CONTAINS A REAL GENERAL MATRIX.
C
C        MATZ SHOULD BE SET TO .TRUE. IF THE RIGHT HAND TRANSFORMATIONS
C          ARE TO BE ACCUMULATED FOR LATER USE IN COMPUTING
C          EIGENVECTORS, AND TO .FALSE. OTHERWISE.
C
C     ON OUTPUT
C
C        A HAS BEEN REDUCED TO UPPER HESSENBERG FORM.  THE ELEMENTS
C          BELOW THE FIRST SUBDIAGONAL HAVE BEEN SET TO ZERO.
C
C        B HAS BEEN REDUCED TO UPPER TRIANGULAR FORM.  THE ELEMENTS
C          BELOW THE MAIN DIAGONAL HAVE BEEN SET TO ZERO.
C
C        Z CONTAINS THE PRODUCT OF THE RIGHT HAND TRANSFORMATIONS IF
C          MATZ HAS BEEN SET TO .TRUE.  OTHERWISE, Z IS NOT REFERENCED.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
C     .......... INITIALIZE Z ..........
      IF (.NOT. MATZ) GO TO 10
C
      DO 3 J = 1, N
C
         DO 2 I = 1, N
            Z(I,J) = 0.0D0
    2    CONTINUE
C
         Z(J,J) = 1.0D0
    3 CONTINUE
C     .......... REDUCE B TO UPPER TRIANGULAR FORM ..........
   10 IF (N .LE. 1) GO TO 170
      NM1 = N - 1
C
      DO 100 L = 1, NM1
         L1 = L + 1
         S = 0.0D0
C
         DO 20 I = L1, N
            S = S + DABS(B(I,L))
   20    CONTINUE
```

```
C
      IF (S .EQ. 0.0D0) GO TO 100
      S = S + DABS(B(L,L))
      R = 0.0D0
C
      DO 25 I = L, N
         B(I,L) = B(I,L) / S
         R = R + B(I,L)**2
25    CONTINUE
C
      R = DSIGN(DSQRT(R),B(L,L))
      B(L,L) = B(L,L) + R
      RHO = R * B(L,L)
C
      DO 50 J = L1, N
         T = 0.0D0
C
         DO 30 I = L, N
            T = T + B(I,L) * B(I,J)
30       CONTINUE
C
         T = -T / RHO
C
         DO 40 I = L, N
            B(I,J) = B(I,J) + T * B(I,L)
40       CONTINUE
C
50    CONTINUE
C
      DO 80 J = 1, N
         T = 0.0D0
C
         DO 60 I = L, N
            T = T + B(I,L) * A(I,J)
60       CONTINUE
C
         T = -T / RHO
C
         DO 70 I = L, N
            A(I,J) = A(I,J) + T * B(I,L)
70       CONTINUE
C
80    CONTINUE
C
      B(L,L) = -S * R
C
      DO 90 I = L1, N
         B(I,L) = 0.0D0
90    CONTINUE
C
100 CONTINUE
C     ..... REDUCE A TO UPPER HESSENBERG FORM, WHILE
C                KEEPING B TRIANGULAR .....
      IF (N .EQ. 2) GO TO 170
      NM2 = N - 2
C
      DO 160 K = 1, NM2
```

```
         NK1 = NM1 - K
C     ..... FOR L=N-1 STEP -1 UNTIL K+1 DO -- .....
         DO 150 LB = 1, NK1
            L = N - LB
            L1 = L + 1
C     ..... ZERO A(L+1,K) .....
            S = DABS(A(L,K)) + DABS(A(L1,K))
            IF (S .EQ. 0.0D0) GO TO 150
            U1 = A(L,K) / S
            U2 = A(L1,K) / S
            R = DSIGN(DSQRT(U1*U1+U2*U2),U1)
            V1 =  -(U1 + R) / R
            V2 = -U2 / R
            U2 = V2 / V1
C
            DO 110 J = K, N
               T = A(L,J) + U2 * A(L1,J)
               A(L,J) = A(L,J) + T * V1
               A(L1,J) = A(L1,J) + T * V2
  110       CONTINUE
C
            A(L1,K) = 0.0D0
C
            DO 120 J = L, N
               T = B(L,J) + U2 * B(L1,J)
               B(L,J) = B(L,J) + T * V1
               B(L1,J) = B(L1,J) + T * V2
  120       CONTINUE
C     ..... ZERO B(L+1,L) .....
            S = DABS(B(L1,L1)) + DABS(B(L1,L))
            IF (S .EQ. 0.0D0) GO TO 150
            U1 = B(L1,L1) / S
            U2 = B(L1,L) / S
            R = DSIGN(DSQRT(U1*U1+U2*U2),U1)
            V1 =  -(U1 + R) / R
            V2 = -U2 / R
            U2 = V2 / V1
C
            DO 130 I = 1, L1
               T = B(I,L1) + U2 * B(I,L)
               B(I,L1) = B(I,L1) + T * V1
               B(I,L) = B(I,L) + T * V2
  130       CONTINUE
C
            B(L1,L) = 0.0D0
C
            DO 140 I = 1, N
               T = A(I,L1) + U2 * A(I,L)
               A(I,L1) = A(I,L1) + T * V1
               A(I,L) = A(I,L) + T * V2
  140       CONTINUE
C
            IF (.NOT. MATZ) GO TO 150
C
            DO 145 I = 1, N
               T = Z(I,L1) + U2 * Z(I,L)
               Z(I,L1) = Z(I,L1) + T * V1
```

```
                Z(I,L) = Z(I,L) + T * V2
  145       CONTINUE
C
  150    CONTINUE
C
  160 CONTINUE
C
  170 RETURN
      END

      SUBROUTINE QZIT(NM,N,A,B,EPS1,MATZ,Z,IERR)
C
      INTEGER I,J,K,L,N,EN,K1,K2,LD,LL,L1,NA,NM,ISH,ITN,ITS,KM1,LM1,
     X        ENM2,IERR,LOR1,ENORN
      DOUBLE PRECISION A(NM,N),B(NM,N),Z(NM,N)
      DOUBLE PRECISION R,S,T,A1,A2,A3,EP,SH,U1,U2,U3,V1,V2,V3,ANI,A11,
     X       A12,A21,A22,A33,A34,A43,A44,BNI,B11,B12,B22,B33,B34,
     X       B44,EPSA,EPSB,EPS1,ANORM,BNORM,EPSLON
      LOGICAL MATZ,NOTLAS
C
C     THIS SUBROUTINE IS THE SECOND STEP OF THE QZ ALGORITHM
C     FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS,
C     SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART,
C     AS MODIFIED IN TECHNICAL NOTE NASA TN D-7305(1973) BY WARD.
C
C     THIS SUBROUTINE ACCEPTS A PAIR OF REAL MATRICES, ONE OF THEM
C     IN UPPER HESSENBERG FORM AND THE OTHER IN UPPER TRIANGULAR FORM.
C     IT REDUCES THE HESSENBERG MATRIX TO QUASI-TRIANGULAR FORM USING
C     ORTHOGONAL TRANSFORMATIONS WHILE MAINTAINING THE TRIANGULAR FORM
C     OF THE OTHER MATRIX.  IT IS USUALLY PRECEDED BY  QZHES  AND
C     FOLLOWED BY  QZVAL  AND, POSSIBLY,  QZVEC.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRICES.
C
C        A CONTAINS A REAL UPPER HESSENBERG MATRIX.
C
C        B CONTAINS A REAL UPPER TRIANGULAR MATRIX.
C
C        EPS1 IS A TOLERANCE USED TO DETERMINE NEGLIGIBLE ELEMENTS.
C          EPS1 = 0.0 (OR NEGATIVE) MAY BE INPUT, IN WHICH CASE AN
C          ELEMENT WILL BE NEGLECTED ONLY IF IT IS LESS THAN ROUNDOFF
C          ERROR TIMES THE NORM OF ITS MATRIX.  IF THE INPUT EPS1 IS
C          POSITIVE, THEN AN ELEMENT WILL BE CONSIDERED NEGLIGIBLE
C          IF IT IS LESS THAN EPS1 TIMES THE NORM OF ITS MATRIX.  A
C          POSITIVE VALUE OF EPS1 MAY RESULT IN FASTER EXECUTION,
C          BUT LESS ACCURATE RESULTS.
C
C        MATZ SHOULD BE SET TO .TRUE. IF THE RIGHT HAND TRANSFORMATIONS
C          ARE TO BE ACCUMULATED FOR LATER USE IN COMPUTING
C          EIGENVECTORS, AND TO .FALSE. OTHERWISE.
C
```

```
C        Z CONTAINS, IF MATZ HAS BEEN SET TO .TRUE., THE
C          TRANSFORMATION MATRIX PRODUCED IN THE REDUCTION
C          BY  QZHES, IF PERFORMED, OR ELSE THE IDENTITY MATRIX.
C          IF MATZ HAS BEEN SET TO .FALSE., Z IS NOT REFERENCED.
C
C     ON OUTPUT
C
C        A HAS BEEN REDUCED TO QUASI-TRIANGULAR FORM.  THE ELEMENTS
C          BELOW THE FIRST SUBDIAGONAL ARE STILL ZERO AND NO TWO
C          CONSECUTIVE SUBDIAGONAL ELEMENTS ARE NONZERO.
C
C        B IS STILL IN UPPER TRIANGULAR FORM, ALTHOUGH ITS ELEMENTS
C          HAVE BEEN ALTERED.  THE LOCATION B(N,1) IS USED TO STORE
C          EPS1 TIMES THE NORM OF B FOR LATER USE BY  QZVAL  AND
QZVEC.
C
C        Z CONTAINS THE PRODUCT OF THE RIGHT HAND TRANSFORMATIONS
C          (FOR BOTH STEPS) IF MATZ HAS BEEN SET TO .TRUE..
C
C        IERR IS SET TO
C          ZERO       FOR NORMAL RETURN,
C          J          IF THE LIMIT OF 30*N ITERATIONS IS EXHAUSTED
C                     WHILE THE J-TH EIGENVALUE IS BEING SOUGHT.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      IERR = 0
C     ........... COMPUTE EPSA,EPSB ..........
      ANORM = 0.0D0
      BNORM = 0.0D0
C
      DO 30 I = 1, N
         ANI = 0.0D0
         IF (I .NE. 1) ANI = DABS(A(I,I-1))
         BNI = 0.0D0
C
         DO 20 J = I, N
            ANI = ANI + DABS(A(I,J))
            BNI = BNI + DABS(B(I,J))
   20    CONTINUE
C
         IF (ANI .GT. ANORM) ANORM = ANI
         IF (BNI .GT. BNORM) BNORM = BNI
   30 CONTINUE
C
      IF (ANORM .EQ. 0.0D0) ANORM = 1.0D0
      IF (BNORM .EQ. 0.0D0) BNORM = 1.0D0
      EP = EPS1
      IF (EP .GT. 0.0D0) GO TO 50
C     .......... USE ROUNDOFF LEVEL IF EPS1 IS ZERO ..........
      EP = EPSLON(1.0D0)
```

```
   50 EPSA = EP * ANORM
      EPSB = EP * BNORM
C     ..... REDUCE A TO QUASI-TRIANGULAR FORM, WHILE
C                KEEPING B TRIANGULAR .....
      LOR1 = 1
      ENORN = N
      EN = N
      ITN = 30*N
C     ..... BEGIN QZ STEP .....
   60 IF (EN .LE. 2) GO TO 1001
      IF (.NOT. MATZ) ENORN = EN
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
   70 ISH = 2
C     ..... CHECK FOR CONVERGENCE OR REDUCIBILITY.
C                FOR L=EN STEP -1 UNTIL 1 DO -- .....
      DO 80 LL = 1, EN
         LM1 = EN - LL
         L = LM1 + 1
         IF (L .EQ. 1) GO TO 95
         IF (DABS(A(L,LM1)) .LE. EPSA) GO TO 90
   80 CONTINUE
C
   90 A(L,LM1) = 0.0D0
      IF (L .LT. NA) GO TO 95
C     ..... 1-BY-1 OR 2-BY-2 BLOCK ISOLATED .....
      EN = LM1
      GO TO 60
C     ..... CHECK FOR SMALL TOP OF B .....
   95 LD = L
  100 L1 = L + 1
      B11 = B(L,L)
      IF (DABS(B11) .GT. EPSB) GO TO 120
      B(L,L) = 0.0D0
      S = DABS(A(L,L)) + DABS(A(L1,L))
      U1 = A(L,L) / S
      U2 = A(L1,L) / S
      R = DSIGN(DSQRT(U1*U1+U2*U2),U1)
      V1 = -(U1 + R) / R
      V2 = -U2 / R
      U2 = V2 / V1
C
      DO 110 J = L, ENORN
         T = A(L,J) + U2 * A(L1,J)
         A(L,J) = A(L,J) + T * V1
         A(L1,J) = A(L1,J) + T * V2
         T = B(L,J) + U2 * B(L1,J)
         B(L,J) = B(L,J) + T * V1
         B(L1,J) = B(L1,J) + T * V2
  110 CONTINUE
C
      IF (L .NE. 1) A(L,LM1) = -A(L,LM1)
      LM1 = L
      L = L1
      GO TO 90
  120 A11 = A(L,L) / B11
```

```
      A21 = A(L1,L) / B11
      IF (ISH .EQ. 1) GO TO 140
C     .......... ITERATION STRATEGY ..........
      IF (ITN .EQ. 0) GO TO 1000
      IF (ITS .EQ. 10) GO TO 155
C     .......... DETERMINE TYPE OF SHIFT ..........
      B22 = B(L1,L1)
      IF (DABS(B22) .LT. EPSB) B22 = EPSB
      B33 = B(NA,NA)
      IF (DABS(B33) .LT. EPSB) B33 = EPSB
      B44 = B(EN,EN)
      IF (DABS(B44) .LT. EPSB) B44 = EPSB
      A33 = A(NA,NA) / B33
      A34 = A(NA,EN) / B44
      A43 = A(EN,NA) / B33
      A44 = A(EN,EN) / B44
      B34 = B(NA,EN) / B44
      T = 0.5D0 * (A43 * B34 - A33 - A44)
      R = T * T + A34 * A43 - A33 * A44
      IF (R .LT. 0.0D0) GO TO 150
C     .......... DETERMINE SINGLE SHIFT ZEROTH COLUMN OF A ..........
      ISH = 1
      R = DSQRT(R)
      SH = -T + R
      S = -T - R
      IF (DABS(S-A44) .LT. DABS(SH-A44)) SH = S
C     .......... LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS OF A.
C                FOR L=EN-2 STEP -1 UNTIL LD DO -- ..........
      DO 130 LL = LD, ENM2
         L = ENM2 + LD - LL
         IF (L .EQ. LD) GO TO 140
         LM1 = L - 1
         L1 = L + 1
         T = A(L,L)
         IF (DABS(B(L,L)) .GT. EPSB) T = T - SH * B(L,L)
         IF (DABS(A(L,LM1)) .LE. DABS(T/A(L1,L)) * EPSA) GO TO 100
  130 CONTINUE
C
  140 A1 = A11 - SH
      A2 = A21
      IF (L .NE. LD) A(L,LM1) = -A(L,LM1)
      GO TO 160
C     .......... DETERMINE DOUBLE SHIFT ZEROTH COLUMN OF A ..........
  150 A12 = A(L,L1) / B22
      A22 = A(L1,L1) / B22
      B12 = B(L,L1) / B22
      A1 = ((A33 - A11) * (A44 - A11) - A34 * A43 + A43 * B34 * A11)
     X     / A21 + A12 - A11 * B12
      A2 = (A22 - A11) - A21 * B12 - (A33 - A11) - (A44 - A11)
     X     + A43 * B34
      A3 = A(L1+1,L1) / B22
      GO TO 160
C     .......... AD HOC SHIFT ..........
  155 A1 = 0.0D0
      A2 = 1.0D0
      A3 = 1.1605D0
```

```
  160 ITS = ITS + 1
      ITN = ITN - 1
      IF (.NOT. MATZ) LOR1 = LD
C     ........... MAIN LOOP ..........
      DO 260 K = L, NA
         NOTLAS = K .NE. NA .AND. ISH .EQ. 2
         K1 = K + 1
         K2 = K + 2
         KM1 = MAX0(K-1,L)
         LL = MIN0(EN,K1+ISH)
         IF (NOTLAS) GO TO 190
C     ........... ZERO A(K+1,K-1) ..........
         IF (K .EQ. L) GO TO 170
         A1 = A(K,KM1)
         A2 = A(K1,KM1)
  170    S = DABS(A1) + DABS(A2)
         IF (S .EQ. 0.0D0) GO TO 70
         U1 = A1 / S
         U2 = A2 / S
         R = DSIGN(DSQRT(U1*U1+U2*U2),U1)
         V1 = -(U1 + R) / R
         V2 = -U2 / R
         U2 = V2 / V1
C
         DO 180 J = KM1, ENORN
            T = A(K,J) + U2 * A(K1,J)
            A(K,J) = A(K,J) + T * V1
            A(K1,J) = A(K1,J) + T * V2
            T = B(K,J) + U2 * B(K1,J)
            B(K,J) = B(K,J) + T * V1
            B(K1,J) = B(K1,J) + T * V2
  180    CONTINUE
C
         IF (K .NE. L) A(K1,KM1) = 0.0D0
         GO TO 240
C     ........... ZERO A(K+1,K-1) AND A(K+2,K-1) ..........
  190    IF (K .EQ. L) GO TO 200
         A1 = A(K,KM1)
         A2 = A(K1,KM1)
         A3 = A(K2,KM1)
  200    S = DABS(A1) + DABS(A2) + DABS(A3)
         IF (S .EQ. 0.0D0) GO TO 260
         U1 = A1 / S
         U2 = A2 / S
         U3 = A3 / S
         R = DSIGN(DSQRT(U1*U1+U2*U2+U3*U3),U1)
         V1 = -(U1 + R) / R
         V2 = -U2 / R
         V3 = -U3 / R
         U2 = V2 / V1
         U3 = V3 / V1
C
         DO 210 J = KM1, ENORN
            T = A(K,J) + U2 * A(K1,J) + U3 * A(K2,J)
            A(K,J) = A(K,J) + T * V1
            A(K1,J) = A(K1,J) + T * V2
            A(K2,J) = A(K2,J) + T * V3
```

```
            T = B(K,J) + U2 * B(K1,J) + U3 * B(K2,J)
            B(K,J) = B(K,J) + T * V1
            B(K1,J) = B(K1,J) + T * V2
            B(K2,J) = B(K2,J) + T * V3
  210    CONTINUE
C
         IF (K .EQ. L) GO TO 220
         A(K1,KM1) = 0.0D0
         A(K2,KM1) = 0.0D0
C     .......... ZERO B(K+2,K+1) AND B(K+2,K) ..........
  220    S = DABS(B(K2,K2)) + DABS(B(K2,K1)) + DABS(B(K2,K))
         IF (S .EQ. 0.0D0) GO TO 240
         U1 = B(K2,K2) / S
         U2 = B(K2,K1) / S
         U3 = B(K2,K) / S
         R = DSIGN(DSQRT(U1*U1+U2*U2+U3*U3),U1)
         V1 = -(U1 + R) / R
         V2 = -U2 / R
         V3 = -U3 / R
         U2 = V2 / V1
         U3 = V3 / V1
C
         DO 230 I = LOR1, LL
            T = A(I,K2) + U2 * A(I,K1) + U3 * A(I,K)
            A(I,K2) = A(I,K2) + T * V1
            A(I,K1) = A(I,K1) + T * V2
            A(I,K) = A(I,K) + T * V3
            T = B(I,K2) + U2 * B(I,K1) + U3 * B(I,K)
            B(I,K2) = B(I,K2) + T * V1
            B(I,K1) = B(I,K1) + T * V2
            B(I,K) = B(I,K) + T * V3
  230    CONTINUE
C
         B(K2,K) = 0.0D0
         B(K2,K1) = 0.0D0
         IF (.NOT. MATZ) GO TO 240
C
         DO 235 I = 1, N
            T = Z(I,K2) + U2 * Z(I,K1) + U3 * Z(I,K)
            Z(I,K2) = Z(I,K2) + T * V1
            Z(I,K1) = Z(I,K1) + T * V2
            Z(I,K) = Z(I,K) + T * V3
  235    CONTINUE
C     .......... ZERO B(K+1,K) ..........
  240    S = DABS(B(K1,K1)) + DABS(B(K1,K))
         IF (S .EQ. 0.0D0) GO TO 260
         U1 = B(K1,K1) / S
         U2 = B(K1,K) / S
         R = DSIGN(DSQRT(U1*U1+U2*U2),U1)
         V1 = -(U1 + R) / R
         V2 = -U2 / R
         U2 = V2 / V1
C
         DO 250 I = LOR1, LL
            T = A(I,K1) + U2 * A(I,K)
            A(I,K1) = A(I,K1) + T * V1
            A(I,K) = A(I,K) + T * V2
```

```
            T = B(I,K1) + U2 * B(I,K)
            B(I,K1) = B(I,K1) + T * V1
            B(I,K) = B(I,K) + T * V2
  250    CONTINUE
C
         B(K1,K) = 0.0D0
         IF (.NOT. MATZ) GO TO 260
C
         DO 255 I = 1, N
            T = Z(I,K1) + U2 * Z(I,K)
            Z(I,K1) = Z(I,K1) + T * V1
            Z(I,K) = Z(I,K) + T * V2
  255    CONTINUE
C
  260 CONTINUE
C     .......... END QZ STEP ..........
      GO TO 70
C     .......... SET ERROR -- ALL EIGENVALUES HAVE NOT
C                CONVERGED AFTER 30*N ITERATIONS ..........
 1000 IERR = EN
C     .......... SAVE EPSB FOR USE BY QZVAL AND QZVEC ..........
 1001 IF (N .GT. 1) B(N,1) = EPSB
      RETURN
      END
      SUBROUTINE QZVAL(NM,N,A,B,ALFR,ALFI,BETA,MATZ,Z)
C
      INTEGER I,J,N,EN,NA,NM,NN,ISW
      DOUBLE PRECISION A(NM,N),B(NM,N),ALFR(N),ALFI(N),BETA(N),Z(NM,N)
      DOUBLE PRECISION C,D,E,R,S,T,AN,A1,A2,BN,CQ,CZ,DI,DR,EI,TI,TR,U1,
     X       U2,V1,V2,A1I,A11,A12,A2I,A21,A22,B11,B12,B22,SQI,SQR,
     X       SSI,SSR,SZI,SZR,A11I,A11R,A12I,A12R,A22I,A22R,EPSB
      LOGICAL MATZ
C
C     THIS SUBROUTINE IS THE THIRD STEP OF THE QZ ALGORITHM
C     FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS,
C     SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART.
C
C     THIS SUBROUTINE ACCEPTS A PAIR OF REAL MATRICES, ONE OF THEM
C     IN QUASI-TRIANGULAR FORM AND THE OTHER IN UPPER TRIANGULAR FORM.
C     IT REDUCES THE QUASI-TRIANGULAR MATRIX FURTHER, SO THAT ANY
C     REMAINING 2-BY-2 BLOCKS CORRESPOND TO PAIRS OF COMPLEX
C     EIGENVALUES, AND RETURNS QUANTITIES WHOSE RATIOS GIVE THE
C     GENERALIZED EIGENVALUES.  IT IS USUALLY PRECEDED BY  QZHES
C     AND  QZIT  AND MAY BE FOLLOWED BY  QZVEC.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRICES.
C
C        A CONTAINS A REAL UPPER QUASI-TRIANGULAR MATRIX.
C
C        B CONTAINS A REAL UPPER TRIANGULAR MATRIX.  IN ADDITION,
C          LOCATION B(N,1) CONTAINS THE TOLERANCE QUANTITY (EPSB)
```

```
C          COMPUTED AND SAVED IN  QZIT.
C
C        MATZ SHOULD BE SET TO .TRUE. IF THE RIGHT HAND TRANSFORMATIONS
C          ARE TO BE ACCUMULATED FOR LATER USE IN COMPUTING
C          EIGENVECTORS, AND TO .FALSE. OTHERWISE.
C
C        Z CONTAINS, IF MATZ HAS BEEN SET TO .TRUE., THE
C          TRANSFORMATION MATRIX PRODUCED IN THE REDUCTIONS BY QZHES
C          AND QZIT, IF PERFORMED, OR ELSE THE IDENTITY MATRIX.
C          IF MATZ HAS BEEN SET TO .FALSE., Z IS NOT REFERENCED.
C
C     ON OUTPUT
C
C        A HAS BEEN REDUCED FURTHER TO A QUASI-TRIANGULAR MATRIX
C          IN WHICH ALL NONZERO SUBDIAGONAL ELEMENTS CORRESPOND TO
C          PAIRS OF COMPLEX EIGENVALUES.
C
C        B IS STILL IN UPPER TRIANGULAR FORM, ALTHOUGH ITS ELEMENTS
C          HAVE BEEN ALTERED.  B(N,1) IS UNALTERED.
C
C        ALFR AND ALFI CONTAIN THE REAL AND IMAGINARY PARTS OF THE
C          DIAGONAL ELEMENTS OF THE TRIANGULAR MATRIX THAT WOULD BE
C          OBTAINED IF A WERE REDUCED COMPLETELY TO TRIANGULAR FORM
C          BY UNITARY TRANSFORMATIONS.  NON-ZERO VALUES OF ALFI OCCUR
C          IN PAIRS, THE FIRST MEMBER POSITIVE AND THE SECOND NEGATIVE.
C
C        BETA CONTAINS THE DIAGONAL ELEMENTS OF THE CORRESPONDING B,
C          NORMALIZED TO BE REAL AND NON-NEGATIVE.  THE GENERALIZED
C          EIGENVALUES ARE THEN THE RATIOS ((ALFR+I*ALFI)/BETA).
C
C        Z CONTAINS THE PRODUCT OF THE RIGHT HAND TRANSFORMATIONS
C          (FOR ALL THREE STEPS) IF MATZ HAS BEEN SET TO .TRUE.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      EPSB = B(N,1)
      ISW = 1
C     .......... FIND EIGENVALUES OF QUASI-TRIANGULAR MATRICES.
C                FOR EN=N STEP -1 UNTIL 1 DO -- ..........
      DO 510 NN = 1, N
         EN = N + 1 - NN
         NA = EN - 1
         IF (ISW .EQ. 2) GO TO 505
         IF (EN .EQ. 1) GO TO 410
         IF (A(EN,NA) .NE. 0.0D0) GO TO 420
C     .......... 1-BY-1 BLOCK, ONE REAL ROOT ..........
  410    ALFR(EN) = A(EN,EN)
         IF (B(EN,EN) .LT. 0.0D0) ALFR(EN) = -ALFR(EN)
         BETA(EN) = DABS(B(EN,EN))
         ALFI(EN) = 0.0D0
         GO TO 510
```

```
C     .......... 2-BY-2 BLOCK ..........
  420    IF (DABS(B(NA,NA)) .LE. EPSB) GO TO 455
         IF (DABS(B(EN,EN)) .GT. EPSB) GO TO 430
         A1 = A(EN,EN)
         A2 = A(EN,NA)
         BN = 0.0D0
         GO TO 435
  430    AN = DABS(A(NA,NA)) + DABS(A(NA,EN)) + DABS(A(EN,NA))
     X      + DABS(A(EN,EN))
         BN = DABS(B(NA,NA)) + DABS(B(NA,EN)) + DABS(B(EN,EN))
         A11 = A(NA,NA) / AN
         A12 = A(NA,EN) / AN
         A21 = A(EN,NA) / AN
         A22 = A(EN,EN) / AN
         B11 = B(NA,NA) / BN
         B12 = B(NA,EN) / BN
         B22 = B(EN,EN) / BN
         E = A11 / B11
         EI = A22 / B22
         S = A21 / (B11 * B22)
         T = (A22 - E * B22) / B22
         IF (DABS(E) .LE. DABS(EI)) GO TO 431
         E = EI
         T = (A11 - E * B11) / B11
  431    C = 0.5D0 * (T - S * B12)
         D = C * C + S * (A12 - E * B12)
         IF (D .LT. 0.0D0) GO TO 480
C     .......... TWO REAL ROOTS.
C                ZERO BOTH A(EN,NA) AND B(EN,NA) ..........
         E = E + (C + DSIGN(DSQRT(D),C))
         A11 = A11 - E * B11
         A12 = A12 - E * B12
         A22 = A22 - E * B22
         IF (DABS(A11) + DABS(A12) .LT.
     X       DABS(A21) + DABS(A22)) GO TO 432
         A1 = A12
         A2 = A11
         GO TO 435
  432    A1 = A22
         A2 = A21
C     .......... CHOOSE AND APPLY REAL Z ..........
  435    S = DABS(A1) + DABS(A2)
         U1 = A1 / S
         U2 = A2 / S
         R = DSIGN(DSQRT(U1*U1+U2*U2),U1)
         V1 = -(U1 + R) / R
         V2 = -U2 / R
         U2 = V2 / V1
C
         DO 440 I = 1, EN
            T = A(I,EN) + U2 * A(I,NA)
            A(I,EN) = A(I,EN) + T * V1
            A(I,NA) = A(I,NA) + T * V2
            T = B(I,EN) + U2 * B(I,NA)
            B(I,EN) = B(I,EN) + T * V1
            B(I,NA) = B(I,NA) + T * V2
  440    CONTINUE
```

```
C
        IF (.NOT. MATZ) GO TO 450
C
        DO 445 I = 1, N
           T = Z(I,EN) + U2 * Z(I,NA)
           Z(I,EN) = Z(I,EN) + T * V1
           Z(I,NA) = Z(I,NA) + T * V2
  445   CONTINUE
C
  450   IF (BN .EQ. 0.0D0) GO TO 475
        IF (AN .LT. DABS(E) * BN) GO TO 455
        A1 = B(NA,NA)
        A2 = B(EN,NA)
        GO TO 460
  455   A1 = A(NA,NA)
        A2 = A(EN,NA)
C     ..... CHOOSE AND APPLY REAL Q .....
  460   S = DABS(A1) + DABS(A2)
        IF (S .EQ. 0.0D0) GO TO 475
        U1 = A1 / S
        U2 = A2 / S
        R = DSIGN(DSQRT(U1*U1+U2*U2),U1)
        V1 = -(U1 + R) / R
        V2 = -U2 / R
        U2 = V2 / V1
C
        DO 470 J = NA, N
           T = A(NA,J) + U2 * A(EN,J)
           A(NA,J) = A(NA,J) + T * V1
           A(EN,J) = A(EN,J) + T * V2
           T = B(NA,J) + U2 * B(EN,J)
           B(NA,J) = B(NA,J) + T * V1
           B(EN,J) = B(EN,J) + T * V2
  470   CONTINUE
C
  475   A(EN,NA) = 0.0D0
        B(EN,NA) = 0.0D0
        ALFR(NA) = A(NA,NA)
        ALFR(EN) = A(EN,EN)
        IF (B(NA,NA) .LT. 0.0D0) ALFR(NA) = -ALFR(NA)
        IF (B(EN,EN) .LT. 0.0D0) ALFR(EN) = -ALFR(EN)
        BETA(NA) = DABS(B(NA,NA))
        BETA(EN) = DABS(B(EN,EN))
        ALFI(EN) = 0.0D0
        ALFI(NA) = 0.0D0
        GO TO 505
C     ..... TWO COMPLEX ROOTS .....
  480   E = E + C
        EI = DSQRT(-D)
        A11R = A11 - E * B11
        A11I = EI * B11
        A12R = A12 - E * B12
        A12I = EI * B12
        A22R = A22 - E * B22
        A22I = EI * B22
        IF (DABS(A11R) + DABS(A11I) + DABS(A12R) + DABS(A12I) .LT.
     X       DABS(A21) + DABS(A22R) + DABS(A22I)) GO TO 482
```

```
         A1 = A12R
         A1I = A12I
         A2 = -A11R
         A2I = -A11I
         GO TO 485
  482    A1 = A22R
         A1I = A22I
         A2 = -A21
         A2I = 0.0D0
C     .......... CHOOSE COMPLEX Z ..........
  485    CZ = DSQRT(A1*A1+A1I*A1I)
         IF (CZ .EQ. 0.0D0) GO TO 487
         SZR = (A1 * A2 + A1I * A2I) / CZ
         SZI = (A1 * A2I - A1I * A2) / CZ
         R = DSQRT(CZ*CZ+SZR*SZR+SZI*SZI)
         CZ = CZ / R
         SZR = SZR / R
         SZI = SZI / R
         GO TO 490
  487    SZR = 1.0D0
         SZI = 0.0D0
  490    IF (AN .LT. (DABS(E) + EI) * BN) GO TO 492
         A1 = CZ * B11 + SZR * B12
         A1I = SZI * B12
         A2 = SZR * B22
         A2I = SZI * B22
         GO TO 495
  492    A1 = CZ * A11 + SZR * A12
         A1I = SZI * A12
         A2 = CZ * A21 + SZR * A22
         A2I = SZI * A22
C     .......... CHOOSE COMPLEX Q ..........
  495    CQ = DSQRT(A1*A1+A1I*A1I)
         IF (CQ .EQ. 0.0D0) GO TO 497
         SQR = (A1 * A2 + A1I * A2I) / CQ
         SQI = (A1 * A2I - A1I * A2) / CQ
         R = DSQRT(CQ*CQ+SQR*SQR+SQI*SQI)
         CQ = CQ / R
         SQR = SQR / R
         SQI = SQI / R
         GO TO 500
  497    SQR = 1.0D0
         SQI = 0.0D0
C     .......... COMPUTE DIAGONAL ELEMENTS THAT WOULD RESULT
C                IF TRANSFORMATIONS WERE APPLIED ..........
  500    SSR = SQR * SZR + SQI * SZI
         SSI = SQR * SZI - SQI * SZR
         I = 1
         TR = CQ * CZ * A11 + CQ * SZR * A12 + SQR * CZ * A21
     X      + SSR * A22
         TI = CQ * SZI * A12 - SQI * CZ * A21 + SSI * A22
         DR = CQ * CZ * B11 + CQ * SZR * B12 + SSR * B22
         DI = CQ * SZI * B12 + SSI * B22
         GO TO 503
  502    I = 2
         TR = SSR * A11 - SQR * CZ * A12 - CQ * SZR * A21
     X      + CQ * CZ * A22
```

```
         TI = -SSI * A11 - SQI * CZ * A12 + CQ * SZI * A21
         DR = SSR * B11 - SQR * CZ * B12 + CQ * CZ * B22
         DI = -SSI * B11 - SQI * CZ * B12
  503    T = TI * DR - TR * DI
         J = NA
         IF (T .LT. 0.0D0) J = EN
         R = DSQRT(DR*DR+DI*DI)
         BETA(J) = BN * R
         ALFR(J) = AN * (TR * DR + TI * DI) / R
         ALFI(J) = AN * T / R
         IF (I .EQ. 1) GO TO 502
  505    ISW = 3 - ISW
  510 CONTINUE
      B(N,1) = EPSB
C
      RETURN
      END

      SUBROUTINE QZVEC(NM,N,A,B,ALFR,ALFI,BETA,Z)
C
      INTEGER I,J,K,M,N,EN,II,JJ,NA,NM,NN,ISW,ENM2
      DOUBLE PRECISION A(NM,N),B(NM,N),ALFR(N),ALFI(N),BETA(N),Z(NM,N)
      DOUBLE PRECISION
     D,Q,R,S,T,W,X,Y,DI,DR,RA,RR,SA,TI,TR,T1,T2,W1,X1,
     X       ZZ,Z1,ALFM,ALMI,ALMR,BETM,EPSB
C
C     THIS SUBROUTINE IS THE OPTIONAL FOURTH STEP OF THE QZ ALGORITHM
C     FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS,
C     SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART.
C
C     THIS SUBROUTINE ACCEPTS A PAIR OF REAL MATRICES, ONE OF THEM IN
C     QUASI-TRIANGULAR FORM (IN WHICH EACH 2-BY-2 BLOCK CORRESPONDS TO
C     A PAIR OF COMPLEX EIGENVALUES) AND THE OTHER IN UPPER TRIANGULAR
C     FORM.  IT COMPUTES THE EIGENVECTORS OF THE TRIANGULAR PROBLEM AND
C     TRANSFORMS THE RESULTS BACK TO THE ORIGINAL COORDINATE SYSTEM.
C     IT IS USUALLY PRECEDED BY  QZHES,  QZIT, AND  QZVAL.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRICES.
C
C        A CONTAINS A REAL UPPER QUASI-TRIANGULAR MATRIX.
C
C        B CONTAINS A REAL UPPER TRIANGULAR MATRIX.  IN ADDITION,
C          LOCATION B(N,1) CONTAINS THE TOLERANCE QUANTITY (EPSB)
C          COMPUTED AND SAVED IN  QZIT.
C
C        ALFR, ALFI, AND BETA  ARE VECTORS WITH COMPONENTS WHOSE
C          RATIOS ((ALFR+I*ALFI)/BETA) ARE THE GENERALIZED
C          EIGENVALUES.  THEY ARE USUALLY OBTAINED FROM  QZVAL.
C
C        Z CONTAINS THE TRANSFORMATION MATRIX PRODUCED IN THE
C          REDUCTIONS BY  QZHES,  QZIT, AND  QZVAL, IF PERFORMED.
```

```
C          IF THE EIGENVECTORS OF THE TRIANGULAR PROBLEM ARE
C          DESIRED, Z MUST CONTAIN THE IDENTITY MATRIX.
C
C     ON OUTPUT
C
C        A IS UNALTERED.  ITS SUBDIAGONAL ELEMENTS PROVIDE INFORMATION
C           ABOUT THE STORAGE OF THE COMPLEX EIGENVECTORS.
C
C        B HAS BEEN DESTROYED.
C
C        ALFR, ALFI, AND BETA ARE UNALTERED.
C
C        Z CONTAINS THE REAL AND IMAGINARY PARTS OF THE EIGENVECTORS.
C          IF ALFI(I) .EQ. 0.0, THE I-TH EIGENVALUE IS REAL AND
C            THE I-TH COLUMN OF Z CONTAINS ITS EIGENVECTOR.
C          IF ALFI(I) .NE. 0.0, THE I-TH EIGENVALUE IS COMPLEX.
C            IF ALFI(I) .GT. 0.0, THE EIGENVALUE IS THE FIRST OF
C              A COMPLEX PAIR AND THE I-TH AND (I+1)-TH COLUMNS
C              OF Z CONTAIN ITS EIGENVECTOR.
C            IF ALFI(I) .LT. 0.0, THE EIGENVALUE IS THE SECOND OF
C              A COMPLEX PAIR AND THE (I-1)-TH AND I-TH COLUMNS
C              OF Z CONTAIN THE CONJUGATE OF ITS EIGENVECTOR.
C          EACH EIGENVECTOR IS NORMALIZED SO THAT THE MODULUS
C          OF ITS LARGEST COMPONENT IS 1.0 .
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
-
C
      EPSB = B(N,1)
      ISW = 1
C     .......... FOR EN=N STEP -1 UNTIL 1 DO -- ..........
      DO 800 NN = 1, N
         EN = N + 1 - NN
         NA = EN - 1
         IF (ISW .EQ. 2) GO TO 795
         IF (ALFI(EN) .NE. 0.0D0) GO TO 710
C     .......... REAL VECTOR ..........
         M = EN
         B(EN,EN) = 1.0D0
         IF (NA .EQ. 0) GO TO 800
         ALFM = ALFR(M)
         BETM = BETA(M)
C     .......... FOR I=EN-1 STEP -1 UNTIL 1 DO -- ..........
         DO 700 II = 1, NA
            I = EN - II
            W = BETM * A(I,I) - ALFM * B(I,I)
            R = 0.0D0
C
            DO 610 J = M, EN
  610       R = R + (BETM * A(I,J) - ALFM * B(I,J)) * B(J,EN)
C
            IF (I .EQ. 1 .OR. ISW .EQ. 2) GO TO 630
```

```
            IF (BETM * A(I,I-1) .EQ. 0.0D0) GO TO 630
            ZZ = W
            S = R
            GO TO 690
  630       M = I
            IF (ISW .EQ. 2) GO TO 640
C     .......... REAL 1-BY-1 BLOCK ..........
            T = W
            IF (W .EQ. 0.0D0) T = EPSB
            B(I,EN) = -R / T
            GO TO 700
C     .......... REAL 2-BY-2 BLOCK ..........
  640       X = BETM * A(I,I+1) - ALFM * B(I,I+1)
            Y = BETM * A(I+1,I)
            Q = W * ZZ - X * Y
            T = (X * S - ZZ * R) / Q
            B(I,EN) = T
            IF (DABS(X) .LE. DABS(ZZ)) GO TO 650
            B(I+1,EN) = (-R - W * T) / X
            GO TO 690
  650       B(I+1,EN) = (-S - Y * T) / ZZ
  690       ISW = 3 - ISW
  700    CONTINUE
C     .......... END REAL VECTOR ..........
         GO TO 800
C     .......... COMPLEX VECTOR ..........
  710    M = NA
         ALMR = ALFR(M)
         ALMI = ALFI(M)
         BETM = BETA(M)
C     .......... LAST VECTOR COMPONENT CHOSEN IMAGINARY SO THAT
C                EIGENVECTOR MATRIX IS TRIANGULAR ..........
         Y = BETM * A(EN,NA)
         B(NA,NA) = -ALMI * B(EN,EN) / Y
         B(NA,EN) = (ALMR * B(EN,EN) - BETM * A(EN,EN)) / Y
         B(EN,NA) = 0.0D0
         B(EN,EN) = 1.0D0
         ENM2 = NA - 1
         IF (ENM2 .EQ. 0) GO TO 795
C     .......... FOR I=EN-2 STEP -1 UNTIL 1 DO -- ..........
         DO 790 II = 1, ENM2
            I = NA - II
            W = BETM * A(I,I) - ALMR * B(I,I)
            W1 = -ALMI * B(I,I)
            RA = 0.0D0
            SA = 0.0D0
C
            DO 760 J = M, EN
               X = BETM * A(I,J) - ALMR * B(I,J)
               X1 = -ALMI * B(I,J)
               RA = RA + X * B(J,NA) - X1 * B(J,EN)
               SA = SA + X * B(J,EN) + X1 * B(J,NA)
  760       CONTINUE
C
            IF (I .EQ. 1 .OR. ISW .EQ. 2) GO TO 770
            IF (BETM * A(I,I-1) .EQ. 0.0D0) GO TO 770
            ZZ = W
```

```
            Z1 = W1
            R = RA
            S = SA
            ISW = 2
            GO TO 790
  770       M = I
            IF (ISW .EQ. 2) GO TO 780
C     .......... COMPLEX 1-BY-1 BLOCK ..........
            TR = -RA
            TI = -SA
  773       DR = W
            DI = W1
C     .......... COMPLEX DIVIDE (T1,T2) = (TR,TI) / (DR,DI) ..........
  775       IF (DABS(DI) .GT. DABS(DR)) GO TO 777
            RR = DI / DR
            D = DR + DI * RR
            T1 = (TR + TI * RR) / D
            T2 = (TI - TR * RR) / D
            GO TO (787,782), ISW
  777       RR = DR / DI
            D = DR * RR + DI
            T1 = (TR * RR + TI) / D
            T2 = (TI * RR - TR) / D
            GO TO (787,782), ISW
C     .......... COMPLEX 2-BY-2 BLOCK ..........
  780       X = BETM * A(I,I+1) - ALMR * B(I,I+1)
            X1 = -ALMI * B(I,I+1)
            Y = BETM * A(I+1,I)
            TR = Y * RA - W * R + W1 * S
            TI = Y * SA - W * S - W1 * R
            DR = W * ZZ - W1 * Z1 - X * Y
            DI = W * Z1 + W1 * ZZ - X1 * Y
            IF (DR .EQ. 0.0D0 .AND. DI .EQ. 0.0D0) DR = EPSB
            GO TO 775
  782       B(I+1,NA) = T1
            B(I+1,EN) = T2
            ISW = 1
            IF (DABS(Y) .GT. DABS(W) + DABS(W1)) GO TO 785
            TR = -RA - X * B(I+1,NA) + X1 * B(I+1,EN)
            TI = -SA - X * B(I+1,EN) - X1 * B(I+1,NA)
            GO TO 773
  785       T1 = (-R - ZZ * B(I+1,NA) + Z1 * B(I+1,EN)) / Y
            T2 = (-S - ZZ * B(I+1,EN) - Z1 * B(I+1,NA)) / Y
  787       B(I,NA) = T1
            B(I,EN) = T2
  790    CONTINUE
C     .......... END COMPLEX VECTOR ..........
  795    ISW = 3 - ISW
  800 CONTINUE
C     .......... END BACK SUBSTITUTION.
C                TRANSFORM TO ORIGINAL COORDINATE SYSTEM.
C                FOR J=N STEP -1 UNTIL 1 DO -- ..........
      DO 880 JJ = 1, N
         J = N + 1 - JJ
C
         DO 880 I = 1, N
            ZZ = 0.0D0
```

```
C
           DO 860 K = 1, J
  860      ZZ = ZZ + Z(I,K) * B(K,J)
C
           Z(I,J) = ZZ
  880 CONTINUE
C     ..... NORMALIZE SO THAT MODULUS OF LARGEST
C                COMPONENT OF EACH VECTOR IS 1.
C                (ISW IS 1 INITIALLY FROM BEFORE) .....
      DO 950 J = 1, N
         D = 0.0D0
         IF (ISW .EQ. 2) GO TO 920
         IF (ALFI(J) .NE. 0.0D0) GO TO 945
C
         DO 890 I = 1, N
            IF (DABS(Z(I,J)) .GT. D) D = DABS(Z(I,J))
  890    CONTINUE
C
         DO 900 I = 1, N
  900    Z(I,J) = Z(I,J) / D
C
         GO TO 950
C
  920    DO 930 I = 1, N
            R = DABS(Z(I,J-1)) + DABS(Z(I,J))
            IF (R .NE. 0.0D0) R = R * DSQRT((Z(I,J-1)/R)**2
     X                                     +(Z(I,J)/R)**2)
            IF (R .GT. D) D = R
  930    CONTINUE
C
         DO 940 I = 1, N
            Z(I,J-1) = Z(I,J-1) / D
            Z(I,J) = Z(I,J) / D
  940    CONTINUE
C
  945    ISW = 3 - ISW
  950 CONTINUE
C
      RETURN
      END
```

References

Li, Jing; Kapania, R.K. 2000: A geometrically exact curved beam theory and its finite element formulation/implementation. *Thesis for Master of Science in Aerospace Engineering*. Virginia Polytechnic Institute and State University

Kapania, R.K., Li, J. 1998: A Four Noded 3-D geometrically nonlinear curved beam element with large displacements/rotations, in Modeling and Simulation Based Engineering, S.N. Atluri and P.E. O'Donoghue (ed.), Technology Science Press, 679-784