```
C      Program Name: FEM1D          Length(INCLUDINMG BLANKS):2440 lines
C
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C      *                       Program  FEM1D                          *
C      *         (A FINITE ELEMENT ANALYSIS COMPUTER PROGRAM)          *
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C      |                                                               |
C      |  This is a finite element computer program for the analysis   |
C      |  of following three model equations and others:               |
C      |                                                               |
C      |  1. Heat transfer, fluid mechanics, bars, and cables:         |
C      |                                                               |
C      |            CT.u* + CT.u** - (AX.u')' + CX.u = FX              |
C      |                                                               |
C      |  2. The Timoshenko beam and circular plate theory:            |
C      |                                                               |
C      |             CT0.w** - [AX.(w' + s)]' + CX.w = FX              |
C      |             CT1.s** - (BX.s')' + AX.(w' + s) = 0              |
C      |                                                               |
C      |  3. The Euler-Bernoulli beam and circular plate theory:       |
C      |                                                               |
C      |               CT.w** + (BX.w'')'' + CX.w = FX                 |
C      |                                                               |
C      | In the above equations  (')  and (*) denote  differentiations |
C      | with respect to space x and time t,  and  AX, BX, CX, CT, and |
C      | FX are functions of x only:                                   |
C      |                                                               |
C      |    AX = AX0 + AX1.X,  BX = BX0 + BX1.X,  CX = CX0 + CX1.X     |
C      |         CT = CT0 + CT1.X,  FX = FX0 + FX1.X + FX2.X.X         |
C      |                                                               |
C      |    In addition to the three model equations, other equations  |
C      | (for example, disks, trusses, and frames) can be analyzed by  |
C      | the program.                                                  |
C      |                                                               |
C
C
C      .                                                               .
C      .              KEY VARIABLES USED IN THE PROGRAM                .
C      . See Table 7.3.2 of the BOOK for a description of the variables.
C      .                                                               .
C      . NDF....... Number of degrees of freedom per node              .
C      . NEQ....... Number of equations in the model (before B. C.)    .
C      . NGP....... Number of Gauss points used in the evaluation of   .
C      .            the element coeffcients,  ELK ,  ELF ,  ELM        .
C      . NHBW...... Half bandwidth of global coefficient matrix  GLK   .
C      . NN ....... Number of total degrees of freedom in the element  .
C      . NPE....... Number of nodes per element                        .
C
C      .                                                               .
C      .          DIMENSIONS OF VARIOUS ARRAYS IN THE PROGRAM          .
C      .                                                               .
C      . Values of MXELM,MXNOD, etc. in the PARAMETER statement should .
C      .        be changed to meet the requirements of problem:        .
C      .                                                               .
C      . MXELM..... Maximum number of elements in the mesh:            .
C      . MXEBC..... Maximum number of speci. primary deg. of freedom   .
C      . MXMBC..... Maximum number of speci. mixed boundary conditions .
C      . MXNBC..... Maximum number of speci. secondary deg. of freedom .
C      . MXNEQ..... Maximum number of equations in the FE model        .
C      . MXNOD..... Maximum number of nodes in the mesh                .
C      .                                                               .
C      . NOTE:  The following dimension statement in subroutine JACOBI .
C      .        should be modified when MXNEQ is greater than 500:     .
C      .         DIMENSION  V(500,500),VT(500,500),W(500,500),IH(500)  .
C      .        The value of MXNEQ should be used in place of '500'    .
C
C      .                                                               .
C      .               SUBROUTINES USED IN THE PROGRAM                 .
C      .                                                               .
C      .  ASSEMBLE, BOUNDARY, COEFFCNT, CONSTRNT, ECHODATA, EQNSOLVR,  .
C      .    EIGNSLVR, JACOBI, MATRXMLT, MESH1D, POSTPROC, REACTION,    .
C      .             SHAPE1D, TIMFORCE, TIMSTRES, TRANSFRM             .
C
C
      IMPLICIT REAL*8(A-H,O-Z)
```

```
      PARAMETER  (MXELM=250,MXNEQ=500,MXEBC=20,MXNBC=20,MXMBC=20,
     *            MXNOD=250,MXMPC=5)
      DIMENSION  DCAX(MXELM,2),DCBX(MXELM,2),DCCX(MXELM,2),DCFX(MXELM,3)
      DIMENSION  GU0(MXNEQ),GU1(MXNEQ),GU2(MXNEQ),GPU(MXNEQ),DX(MXNOD)
      DIMENSION  IBDY(MXEBC),ISPV(MXEBC,2),ISSV(MXNBC,2),INBC(MXMBC,2)
      DIMENSION  IMC1(MXMPC,2),IMC2(MXMPC,2),VMPC(MXMPC,4)
      DIMENSION  ICON(9),VCON(9),TRM(MXNEQ,MXNEQ)
      DIMENSION  GLM(MXNEQ,MXNEQ),GLF(MXNEQ),GLX(MXNOD),NOD(MXELM,4)
      DIMENSION  CS(MXELM),SN(MXELM),CNT(MXELM),SNT(MXELM),XB(MXELM)
      DIMENSION  EGNVAL(MXNEQ),EGNVEC(MXNEQ,MXNEQ),GLK(MXNEQ,MXNEQ)
      DIMENSION  PR(MXELM),SE(MXELM),SL(MXELM),SA(MXELM),SI(MXELM)
      DIMENSION  HF(MXELM),VF(MXELM),PF(MXELM),F3(MXELM),TITLE(20)
      DIMENSION  UREF(MXMBC),VSPV(MXEBC),VSSV(MXNBC),VNBC(MXMBC)
      COMMON/STF1/ELK(9,9),ELM(9,9),ELF(9),ELX(4),ELU(9),ELV(9),ELA(9)
      COMMON/STF2/A1,A2,A3,A4,A5,AX0,AX1,BX0,BX1,CX0,CX1,CT0,CT1,FX0,
     *            FX1,FX2
      COMMON/IO/IN,IT
C
C          |                                                 |
C          |        P R E P R O C E S S O R   U N I T        |
C          |                                                 |
C
      IN=5
      IT=6
      open (in,file=' ')
      open (it,file=' ')
      NT=0
      NSSV=0
      JVEC=1
      TIME=0.0D0
      TOLRNS=1.0D-06
      CALL ECHODATA(IN,IT)
C
      READ(IN,300) TITLE
      READ(IN,*) MODEL,NTYPE,ITEM
      READ(IN,*) IELEM,NEM
      READ(IN,*) ICONT,NPRNT
C
      IF(MODEL.GE.3)THEN
         NPE=2
         IF(MODEL.EQ.4 .AND. NTYPE.GE.1)THEN
            NDF=3
         ELSE
            NDF=2
         ENDIF
         IF(MODEL.EQ.4 .AND. NTYPE.EQ.2)THEN
            IELEM=1
         ELSE
            IELEM=0
         ENDIF
      ELSE
         IF(MODEL.EQ.2)THEN
            NDF=2
            IF(NTYPE.GT.1)IELEM=1
         ELSE
            NDF=1
         ENDIF
         NPE=IELEM+1
      ENDIF
C
C     Data input for BAR-LIKE and BEAM problems (MODEL=1,2, AND 3)
C
      IF(MODEL.NE.4)THEN

         IF(ICONT.NE.0)THEN
            NNM = NEM*(NPE-1)+1
            NEM1=NEM + 1
            READ(IN,*) (DX(I), I=1,NEM1)
            CALL MESH1D(NEM,NPE,NOD,MXELM,MXNOD,DX,GLX)
            READ(IN,*) AX0,AX1
            READ(IN,*) BX0,BX1
            READ(IN,*) CX0,CX1
            IF(ITEM.LT.3)THEN
               READ(IN,*) FX0,FX1,FX2
            ENDIF
         ELSE
```

```
C
C     Read GLX, NOD, and element-wise continuous coefficients [DC.X]
C
            READ(IN,*)NNM
            DO 10 N=1,NEM
            READ(IN,*) (NOD(N,I),I=1,NPE), GLX(N)
            READ(IN,*) (DCAX(N,I),I=1,2)
            READ(IN,*) (DCBX(N,I),I=1,2)
            READ(IN,*) (DCCX(N,I),I=1,2)
   10       READ(IN,*) (DCFX(N,I),I=1,3)
         ENDIF
      ELSE
C
C     Input data for plane TRUSS or FRAME structures (MODEL=4)
C
          READ(IN,*)NNM
          IF(NTYPE.NE.0)THEN
             DO 20 N=1,NEM
             READ(IN,*) PR(N),SE(N),SL(N),SA(N),SI(N),CS(N),SN(N)
             READ(IN,*) HF(N),VF(N),PF(N),XB(N),CNT(N),SNT(N)
   20        READ(IN,*) (NOD(N,I),I=1,2)
          ELSE
             DO 30 N=1,NEM
             READ(IN,*) SE(N),SL(N),SA(N),CS(N),SN(N),HF(N)
   30        READ(IN,*) (NOD(N,I),I=1,2)
          ENDIF
          READ(IN,*) NCON
          IF(NCON.NE.0)THEN
             DO 35 I=1, NCON
   35        READ(IN,*) ICON(I),VCON(I)
          ENDIF
      ENDIF
      NEQ=NNM*NDF
C
C     Read data on BOUNDARY CONDITIONS of three kinds: Dirichlet (PV)
C               Neumann (SV), and Newton's (MIXED) types
C
      READ(IN,*) NSPV
      IF(NSPV.NE.0)THEN
          DO 40 NB=1,NSPV
          IF(ITEM.GT.2)THEN
             READ(IN,*) (ISPV(NB,J),J=1,2)
          ELSE
             READ(IN,*) (ISPV(NB,J),J=1,2),VSPV(NB)
          ENDIF
   40     CONTINUE
      ENDIF
C
      IF(ITEM.LE.2)THEN
         READ(IN,*) NSSV
         IF(NSSV.NE.0)THEN
            DO 50 IB=1,NSSV
   50       READ(IN,*) (ISSV(IB,J),J=1,2),VSSV(IB)
         ENDIF
      ENDIF
C
      READ(IN,*) NNBC
      IF(NNBC.NE.0)THEN
         DO 60 I=1, NNBC
   60    READ(IN,*) (INBC(I,J),J=1,2),VNBC(I),UREF(I)
      ENDIF
C
C     Read data on multi-point constraints
C
      READ(IN,*) NMPC
      IF(NMPC.NE.0)THEN
         DO 65 I=1, NMPC
   65    READ(IN,*)(IMC1(I,J),J=1,2),(IMC2(I,J),J=1,2),(VMPC(I,J),J=1,4)
      ENDIF
C
      IF(ITEM .NE. 0)THEN
C
C     Input data here for TIME-DEPENDENT problems
C
          IF(ITEM.LE.3)THEN
             READ(IN,*) CT0,CT1
```

```
          ENDIF
          IF(ITEM.LE.2)THEN
             READ(IN,*) DT,ALFA,GAMA
             READ(IN,*) INCOND,NTIME,INTVL
             A1=ALFA*DT
             A2=(1.0-ALFA)*DT
             IF(INCOND.NE.0)THEN
                READ(IN,*) (GU0(I),I=1,NEQ)
             ELSE
                DO 70 I=1,NEQ
   70           GU0(I)=0.0
             ENDIF
             IF(ITEM.EQ.2)THEN
                A3=2.0/GAMA/(DT*DT)
                A4=A3*DT
                A5=1.0/GAMA-1.0
                IF(INCOND.NE.0)THEN
                   READ(IN,*) (GU1(I),I=1,NEQ)
                ELSE
                   DO 80 I=1,NEQ
                   GU1(I)=0.0
   80              GU2(I)=0.0
                ENDIF
             ENDIF
         ENDIF
      ENDIF
C
C     ----------------------------------------------------------------
C      E  N  D     O  F     T  H  E     I  N  P  U  T     D  A  T  A
C     ----------------------------------------------------------------
C     Compute the half BANDWIDTH of the coefficient matrix GLK
C
      NHBW=0.0
      DO 90 N=1,NEM
      DO 90 I=1,NPE
      DO 90 J=1,NPE
      NW=(IABS(NOD(N,I)-NOD(N,J))+1)*NDF
   90 IF(NHBW.LT.NW) NHBW=NW
C
C     ----------------------------------------------------------------
C               P R I N T   T H E   I N P U T   D A T A
C     ----------------------------------------------------------------
C
      WRITE(IT,530)
      WRITE(IT,310)
      WRITE(IT,530)
      WRITE(IT,300) TITLE
      WRITE(IT,320) MODEL,NTYPE
      WRITE(IT,350) IELEM,NDF,NEM,NEQ,NHBW,NSPV,NSSV,NNBC,NMPC
C
      IF(ITEM.NE.0)THEN
         IF(ITEM.LE.2)THEN
            WRITE(IT,330)
            WRITE(IT,390) CT0,CT1,ALFA,GAMA,DT,NTIME,INTVL
            IF(INCOND.NE.0)THEN
               WRITE(IT,370)
               WRITE(IT,540) (GU0(I),I=1,NEQ)
               IF(ITEM.EQ.2)THEN
                  WRITE(IT,380)
                  WRITE(IT,540) (GU1(I),I=1,NEQ)
               ENDIF
            ENDIF
         ELSE
            WRITE(IT,340)
            IF(ITEM.LE.3)THEN
               WRITE(IT,400) CT0,CT1
            ENDIF
         ENDIF
      ENDIF
C
      IF(NSPV.NE.0)THEN
         WRITE(IT,480)
         DO 100 IB=1,NSPV
         IF(ITEM.LE.2)THEN
            WRITE(IT,490) (ISPV(IB,J),J=1,2),VSPV(IB)
         ELSE
```

```
            WRITE(IT,490) (ISPV(IB,J),J=1,2)
         ENDIF
  100    CONTINUE
      ENDIF
C
      IF(NSSV.NE.0)THEN
         WRITE(IT,500)
         DO 110 IB=1,NSSV
  110    WRITE(IT,490) (ISSV(IB,J),J=1,2),VSSV(IB)
      ENDIF
C
      IF(NNBC.NE.0)THEN
         WRITE(IT,510)
         DO 120 I=1,NNBC
  120    WRITE(IT,490) (INBC(I,J),J=1,2),VNBC(I),UREF(I)
      ENDIF
C
      IF(NMPC.NE.0)THEN
         WRITE(IT,515)
         DO 125 I=1, NMPC
  125    WRITE(IT,495)(IMC1(I,J),J=1,2),(IMC2(I,J),J=1,2),
     *                      (VMPC(I,J),J=1,4)
      ENDIF
C

      IF(MODEL.NE.4)THEN
         IF(ICONT.EQ.1)THEN
            WRITE(IT,410)
            WRITE(IT,540) (GLX(I),I=1,NNM)
            WRITE(IT,420)
            IF(MODEL.NE.3)THEN
               WRITE(IT,440) AX0,AX1,BX0,BX1,CX0,CX1,FX0,FX1,FX2
            ELSE
               WRITE(IT,445) AX0,AX1,BX0,BX1,CX0,CX1
            ENDIF
         ELSE
            DO 130 N=1,NEM
            WRITE(IT,430) N,GLX(N)
  130       WRITE(IT,440) (DCAX(N,I),I=1,2),(DCBX(N,I),I=1,2),
     *                    (DCCX(N,I),I=1,2),(DCFX(N,I),I=1,3)
         ENDIF
      ELSE
         DO 140 N=1,NEM
         WRITE(IT,460) N
         IF(NTYPE.NE.0)THEN
            WRITE(IT,450) PR(N),SE(N),SL(N),SA(N),SI(N),CS(N),SN(N),
     *                    HF(N),VF(N),PF(N),XB(N),CNT(N),SNT(N),
     *                    (NOD(N,I),I=1,2)
         ELSE
            WRITE(IT,470) SE(N),SL(N),SA(N),CS(N),SN(N),HF(N),
     *                    (NOD(N,I),I=1,2)
         ENDIF
  140    CONTINUE
      ENDIF
C
C          |                                                |
C          |          P R O C E S S O R   U N I T           |
C          |                                                |
C
C     TIME MARCHING scheme begins here. For  ITEM=2, initial conditions
C     on second derivatives of the solution are computed in the program
C
      IF(ITEM.NE.0)THEN
         IF(ITEM.EQ.1)THEN
            NT=NT+1
            TIME=TIME+DT
         ENDIF
      ENDIF
C
      IF(ITEM.GE.3)NHBW=NEQ
C
C     Initialize global matrices and vectors
C
  150 DO 160 I=1,NEQ
      GLF(I)=0.0
      DO 160 J=1,NHBW
```

```
      IF(ITEM.GE.3)THEN
         GLM(I,J)=0.0
      ENDIF
  160 GLK(I,J)=0.0
C
C     Do-loop for ELEMENT CALCULATIONS and ASSEMBLY
C
      DO 200 NE = 1, NEM
      IF(MODEL.NE.4)THEN
         IF(ICONT.NE.1) THEN
            AX0=DCAX(NE,1)
            AX1=DCAX(NE,2)
            BX0=DCBX(NE,1)
            BX1=DCBX(NE,2)
            CX0=DCCX(NE,1)
            CX1=DCCX(NE,2)
            FX0=DCFX(NE,1)
            FX1=DCFX(NE,2)
            FX2=DCFX(NE,3)
         ENDIF
C
         L=0
         DO 180 I=1,NPE
         NI=NOD(NE,I)
         IF(ICONT.EQ.1)THEN
            ELX(I)=GLX(NI)
         ELSE
            ELX(1)=0.0
            ELX(2)=0.5*GLX(NE)
            ELX(NPE)=GLX(NE)
         ENDIF
         IF(ITEM.EQ.1 .OR. ITEM.EQ.2)THEN
            LI=(NI-1)*NDF
            DO 170 J=1,NDF
            LI=LI+1
            L=L+1
            ELU(L)=GU0(LI)
            IF(ITEM.EQ.2 .AND. NT.GT.0)THEN
               ELV(L)=GU1(LI)
               ELA(L)=GU2(LI)
            ENDIF
  170       CONTINUE
         ENDIF
  180    CONTINUE
C
         CALL COEFFCNT(IELEM,ITEM,MODEL,NDF,NPE,TIME,NTYPE,NE,F3,MXELM)
      ELSE
         CALL TRANSFRM(MXELM,NE,NTYPE,PR,SE,SL,SA,SI,CS,SN,CNT,SNT,
     *                 HF,VF,PF,XB)
      ENDIF
C
      IF(NPRNT .NE.0)THEN
         NN = NPE*NDF

         IF(NPRNT .LE.2)THEN
            IF(NE.LE.5 .AND. NT.LE.1)THEN
               WRITE(IT,550)
               DO 190 I=1,NN
  190          WRITE(IT,540) (ELK(I,J),J=1,NN)
               IF(ITEM.GE.3)THEN
                  WRITE(IT,360)
                  DO 195 I=1,NN
  195             WRITE(IT,540) (ELM(I,J),J=1,NN)
               ELSE
                  WRITE(IT,560)
                  WRITE(IT,540) (ELF(I),I=1,NN)
               ENDIF
            ENDIF
         ENDIF
      ENDIF
C
C     Assemble element matrices
C
      CALL ASSEMBLE(NOD,MXELM,MXNEQ,NDF,NPE,NE,ITEM,GLK,GLM,GLF)
C
  200 CONTINUE
```

```
C
C     Call subroutine CONSTRNT to impose constraint boundary conditions,
C     for example, inclined support conditions
C
      IF(MODEL.EQ.4)THEN
         IF(NCON.NE.0)THEN

      CALL CONSTRNT(NEQ,NHBW,NDF,NCON,ICON,VCON,GLK,GLM,GLF,TRM,MXNEQ)
         ENDIF
      ENDIF

C
C     Impose multi-point constraints using the penalty method
C
      IF(NMPC.NE.0)THEN
         IF(NPRNT.EQ.2)THEN
            WRITE(IT,570)
            DO 201 I=1,NEQ
  201       WRITE(IT,540) (GLK(I,J),J=1,NHBW)
         ENDIF
         VMAX=0.0
         DO 204 I=1,NEQ
         DO 204 J=I,NHBW
         VALUE=DABS(GLK(I,J))
         IF(VALUE.GT.VMAX)THEN
            VMAX=VALUE
         ENDIF
  204 CONTINUE
      PNLTY=VMAX*1.0E4
      DO 205 NC=1,NMPC
         NDOF1=(IMC1(NC,1)-1)*NDF+IMC1(NC,2)
         NDOF2=(IMC2(NC,1)-1)*NDF+IMC2(NC,2)
         GLK(NDOF1,1)=GLK(NDOF1,1)+PNLTY*VMPC(NC,1)*VMPC(NC,1)
         GLK(NDOF2,1)=GLK(NDOF2,1)+PNLTY*VMPC(NC,2)*VMPC(NC,2)
         GLF(NDOF1)=GLF(NDOF1)+PNLTY*VMPC(NC,1)*VMPC(NC,3)
         GLF(NDOF2)=GLF(NDOF2)+PNLTY*VMPC(NC,2)*VMPC(NC,3)
         IF(NDOF1.GT.NDOF2)THEN
            NW=NDOF1-NDOF2+1
            GLK(NDOF2,NW)=GLK(NDOF2,NW)+PNLTY*VMPC(NC,1)*VMPC(NC,2)
            GLF(NDOF1)=VMPC(NC,4)
         ELSE
            NW=NDOF2-NDOF1+1
            GLK(NDOF1,NW)=GLK(NDOF1,NW)+PNLTY*VMPC(NC,1)*VMPC(NC,2)
            GLF(NDOF2)=VMPC(NC,4)
         ENDIF
  205    CONTINUE
      ENDIF
C
      IF(NPRNT.EQ.2)THEN
C
C     Print assembled coefficient matrices if required
C
         WRITE(IT,570)
         DO 210 I=1,NEQ
  210    WRITE(IT,540) (GLK(I,J),J=1,NHBW)
         IF(ITEM.GE.3)THEN
            WRITE(IT,575)
            DO 215 I=1,NEQ
  215       WRITE(IT,540) (GLM(I,J),J=1,NHBW)
         ELSE
            WRITE(IT,580)
            WRITE(IT,540) (GLF(I),I=1,NEQ)
         ENDIF
      ENDIF
C
C     Call subroutine BOUNDARY to impose essential, natural and Newton's
C     type boundary conditions on the primary and secondary variables.
C
      CALL BOUNDARY(NEQ,NEQR,NHBW,NSPV,NSSV,NNBC,NDF,DT,ITEM,ALFA,IBDY,
     *              ISPV,ISSV,INBC,UREF,VSPV,VSSV,VNBC,GLK,GLM,GLF,GU0,
     *              MXEBC,MXNBC,MXMBC,MXNEQ)

      IF(NPRNT.EQ.2)THEN
C
C     Print assembled coefficient matrices if required
C
```

```
         WRITE(IT,570)
         DO 211 I=1,NEQ
  211    WRITE(IT,540) (GLK(I,J),J=1,NHBW)
      ENDIF

C
      IF(ITEM.GE.3)THEN
C
C     Call subroutine EIGNSLVR to solve for the eigenvalues and eigenvectors
C
         CALL EGNSOLVR(NEQR,GLK,GLM,EGNVAL,EGNVEC,JVEC,NROT,MXNEQ)
C
         WRITE(IT,690) NROT
         DO 230 NVEC=1,NEQR
         FRQNCY=DSQRT(EGNVAL(NVEC))
         WRITE(IT,700)NVEC,EGNVAL(NVEC),FRQNCY
  230    WRITE(IT,540)(EGNVEC(I,NVEC),I=1,NEQR)
         STOP
      ENDIF
C
      IRES = 0
C
C     Call subroutine EQNSOLVR to solve the finite-element equations
C
      CALL EQNSOLVR(MXNEQ,MXNEQ,NEQ,NHBW,GLK,GLF,IRES)
C
      IF(ITEM.EQ.0)THEN
         WRITE(IT,590)
         WRITE(IT,540) (GLF(NI),NI=1,NEQ)
      ELSE
         IF(NT.EQ.0)THEN
            DO 240 I=1,NEQ
  240       GU2(I)=GLF(I)
            NT=NT+1
            TIME=TIME+DT
            GOTO 150
         ENDIF
C
C     Compute and print current values of GU0, GU1, and GU2
C
         DO 250 I=1,NEQ
         IF(ITEM.EQ.2)THEN
            ACCLRN=A3*(GLF(I)-GU0(I))-A4*GU1(I)-A5*GU2(I)
            GU1(I)=GU1(I)+A2*GU2(I)+A1*ACCLRN
            GU2(I)=ACCLRN
            GPU(I)=GU0(I)
         ELSE
            GPU(I)=GU0(I)
         ENDIF
  250    GU0(I)=GLF(I)
C
         DIFF=0.0
         SOLN=0.0
         DO 260 I=1,NEQ
         SOLN=SOLN+GU0(I)*GU0(I)
  260    DIFF=DIFF+(GLF(I)-GPU(I))**2
         PRCNT=DSQRT(DIFF/SOLN)
         IF(PRCNT.LE.TOLRNS)THEN
            WRITE(IT,640)
            WRITE(IT,540) (GPU(I),I=1,NEQ)
            WRITE(IT,540) (GU0(I),I=1,NEQ)
            STOP
         ELSE
            IF(INTVL.LE.0)INTVL=1
               NTEN=(NT/INTVL)*INTVL
               IF(NTEN.EQ.NT)THEN
                  WRITE(IT,600) TIME, NT
                  WRITE(IT,590)
                  WRITE(IT,540) (GU0(I),I=1,NEQ)
                  IF(ITEM.NE.1) THEN
                     IF(NPRNT.LT.4)THEN
                        WRITE(IT,645)
                        WRITE(IT,540) (GU1(I),I=1,NEQ)
                        WRITE(IT,646)
                        WRITE(IT,540) (GU2(I),I=1,NEQ)
                     ENDIF
```

```
                 ENDIF
                 NT=NT+1
                 TIME=TIME+DT
             ELSE
                 NT=NT+1
                 TIME=TIME+DT
                 GOTO 150
             ENDIF
         ENDIF
      ENDIF
C
C          |                                                |
C          |     P O S T - P R O C E S S O R   U N I T      |
C          |________________________________________________|
C
      IF(NMPC.EQ.0)THEN
         IF(NPRNT.LE.1)THEN
            IF(MODEL.EQ.1)THEN
               WRITE(IT,530)
            ELSE
               IF(MODEL.EQ.4)THEN
                  WRITE(IT,630)
               ENDIF
               WRITE(IT,520)
            ENDIF
C
            IF(MODEL.EQ.1)THEN
               WRITE(IT,647)
               IF(NTYPE.EQ.0)THEN
                  WRITE(IT,610)
               ELSE
                  WRITE(IT,620)
               ENDIF
            ENDIF
C
            IF(MODEL.EQ.2 .OR. MODEL.EQ.3)THEN
               WRITE(IT,647)

               IF(NTYPE.EQ.0)THEN
                  WRITE(IT,650)
               ELSE
                  WRITE(IT,660)
               ENDIF
            ENDIF
C
            IF(MODEL.EQ.4)THEN
               IF(NTYPE.EQ.0)THEN
                  WRITE(IT,680)
               ELSE
                  WRITE(IT,670)
               ENDIF
            ENDIF
C
            IF(MODEL.EQ.1)THEN
               WRITE(IT,530)
            ELSE
               WRITE(IT,520)
            ENDIF
C
            IF(MODEL.LE.3)THEN

            CALL POSTPROC(DCAX,DCBX,DCCX,F3,GLF,GLX,NOD,ICONT,IELEM,NPE,
     *                    MODEL,NTYPE,ITEM,MXELM,MXNEQ,MXNOD,NEM,NDF)
            ELSE
            CALL REACTION(MXELM,MXNEQ,NDF,NEM,NOD,NPE,NTYPE,PR,GLF,
     *                    SE,SL,SA,SI,CS,SN,CNT,SNT,HF,VF,PF,XB)
            ENDIF
C
            IF(MODEL.EQ.1)THEN
               WRITE(IT,530)
            ELSE
               WRITE(IT,520)
            ENDIF
        ENDIF
      ELSE
C
```

```
C     Calculate the reactions at the points where constraints are imposed
C
         DO 280 NC=1,NMPC
         NDOF1=(IMC1(NC,1)-1)*NDF+IMC1(NC,2)
         NDOF2=(IMC2(NC,1)-1)*NDF+IMC2(NC,2)
         GU0(NC)=-PNLTY*VMPC(NC,1)*(VMPC(NC,1)*GLF(NDOF1)
     *            +VMPC(NC,2)*GLF(NDOF2)-VMPC(NC,3))
         GU1(NC)=-PNLTY*VMPC(NC,2)*(VMPC(NC,1)*GLF(NDOF1)
     *            +VMPC(NC,2)*GLF(NDOF2)-VMPC(NC,3))
  280    CONTINUE
         WRITE(IT,545)
         WRITE(IT,540)(GU0(I),I=1,NMPC)
         WRITE(IT,540)(GU1(I),I=1,NMPC)
      ENDIF
C
      IF(ITEM.EQ.0)STOP
      IF(NT.LT.NTIME)THEN
         IF(PRCNT.GT.TOLRNS)THEN
            GOTO 150
         ENDIF
      ELSE
         WRITE(IT,710)
      ENDIF
C
C     ----------------------------------------------------------------
C                       F   O   R   M   A   T   S
C     ----------------------------------------------------------------
C
  300 FORMAT(20A4)
  310 FORMAT(8X,'OUTPUT from program   FEM1D   by J N REDDY')
  320 FORMAT(/,4X,'*** ANALYSIS OF MODEL',I2,', AND TYPE',I2,
     *       ' PROBLEM ***',/,15X,'(see the code below)',/,
     *       /,4X,'MODEL=1,NTYPE=0: A problem described by MODEL EQ. 1',
     *       /,4X,'MODEL=1,NTYPE=1: A circular DISK (PLANE STRESS) ',
     *       /,4X,'MODEL=1,NTYPE>1: A circular DISK (PLANE STRAIN) ',
     *       /,4X,'MODEL=2,NTYPE=0: A Timoshenko BEAM (RIE) problem',
     *       /,4X,'MODEL=2,NTYPE=1: A Timoshenko PLATE (RIE) problem',
     *       /,4X,'MODEL=2,NTYPE=2: A Timoshenko BEAM (CIE) problem',
     *       /,4X,'MODEL=2,NTYPE>2: A Timoshenko PLATE (CIE) problem',
     *       /,4X,'MODEL=3,NTYPE=0: A Euler-Bernoulli BEAM problem',
     *       /,4X,'MODEL=3,NTYPE>0: A Euler-Bernoulli Circular plate',
     *       /,4X,'MODEL=4,NTYPE=0: A plane TRUSS problem',
     *       /,4X,'MODEL=4,NTYPE=1: A Euler-Bernoulli FRAME problem',
     *       /,4X,'MODEL=4,NTYPE=2: A Timoshenko (CIE) FRAME problem',/)
  330 FORMAT(/,4X,'TIME-DEPENDENT (TRANSIENT) ANALYSIS ',/)
  340 FORMAT(/,4X,'E I G E N V A L U E   A N A L Y S I S',/)
  350 FORMAT(/,8X, 'Element type (0, Hermite,>0, Lagrange)..=',I4,/,
     *         8X, 'No. of deg. of freedom per node, NDF....=',I4,/,
     *         8X, 'No. of elements in the mesh, NEM........=',I4,/,
     *         8X, 'No. of total DOF in the model, NEQ......=',I4,/,
     *         8X, 'Half bandwidth of matrix [GLK], NHBW ...=',I4,/,
     *         8X, 'No. of specified primary DOF, NSPV......=',I4,/,
     *         8X, 'No. of specified secondary DOF, NSSV....=',I4,/,
     *         8X, 'No. of specified Newton B. C.: NNBC.....=',I4,/,
     *         8X, 'No. of speci. multi-pt. cond.: NMPC.....=',I4)
  360 FORMAT(/,3X,'Element coefficient matrix, [ELM]:',/)
  370 FORMAT(/,3X, 'Initial conditions on the primary variables:',/)
  380 FORMAT(/,3X, 'Initial cond. on time der. of primary variables:',/)
  390 FORMAT(/,8X,'Coefficient, CT0.......................=',E12.4,/,
     *         8X,'Coefficient, CT1.......................=',E12.4,/,
     *         8X,'Parameter, ALFA........................=',E12.4,/,
     *         8X,'Parameter, GAMA........................=',E12.4,/,
     *         8X,'Time increment, DT.....................=',E12.4,/,
     *         8X,'No. of time steps, NTIME...............=',I4,/,
     *         8X,'Time-step interval to print soln., INTVL=',I4,/)
  400 FORMAT(/,8X,'Coefficient, CT0.......................=',E12.4,/,
     *         8X,'Coefficient, CT1.......................=',E12.4,/)
  410 FORMAT(/,3X,'Global coordinates of the nodes, {GLX}:',/)
  420 FORMAT(/,3X,'Coefficients of the differential equation:',/)
  430 FORMAT(/,5X,'Properties of Element =',I3,//,
     *         8X,'Element length, H ....... =',E12.4)
  440 FORMAT(  8X,'AX0 =',E12.4,5X,'AX1 =',E12.4,/,
     *         8X,'BX0 =',E12.4,5X,'BX1 =',E12.4,/,
     *         8X,'CX0 =',E12.4,5X,'CX1 =',E12.4,/,
     *         8X,'FX0 =',E12.4,5X,'FX1 =',E12.4,5X,'FX2 =',E12.4,/)
  445 FORMAT(  8X,'AX0 =',E12.4,5X,'AX1 =',E12.4,/,
```

```
     *         8X,'BX0 =',E12.4,5X,'BX1 =',E12.4,/,
     *         8X,'CX0 =',E12.4,5X,'CX1 =',E12.4,/)
  450 FORMAT(8X,'The poisson ratio,           PR........ =',E12.4,/,
     *       8X,'Modulus of elasticity,       SE........ =',E12.4,/,
     *       8X,'Length of the element,       SL........ =',E12.4,/,
     *       8X,'Area of cross section,       SA........ =',E12.4,/,
     *       8X,'Moment of inertia,           SI........ =',E12.4,/,
     *       8X,'Cosine of orientation,       CN........ =',E12.4,/,
     *       8X,'Sine of orientation,         SN........ =',E12.4,/,
     *       8X,'Axial body force (constant), HF........ =',E12.4,/,
     *       8X,'Transverse body force (cnst),VF........ =',E12.4,/,
     *       8X,'Internal point force,        PF........ =',E12.4,/,
     *       8X,'Location of PF from node 1,  XB........ =',E12.4,/,
     *       8X,'Orientation of PF: cosine,   CST....... =',E12.4,/,
     *       8X,'Orientation of PF: sine,     SNT....... =',E12.4,/,
     *       8X,'Nodal connectivity:          NOD(I,J).. =',2I6,/)
  460 FORMAT(//,3X,'Element No. =', I3,/)
  470 FORMAT(8X,'Modulus of elasticity,       SE........ =',E12.4,/,
     *       8X,'Length of the element,       SL........ =',E12.4,/,
     *       8X,'Area of cross section,       SA........ =',E12.4,/,
     *       8X,'Cosine of orientation,       CN........ =',E12.4,/,
     *       8X,'Sine of orientation,         SN........ =',E12.4,/,
     *       8X,'Axial body force (constant), HF........ =',E12.4,/,
     *       8X,'Nodal connectivity:          NOD(I,J).. =',2I6,/)
  480 FORMAT(/,3X, 'Boundary information on primary variables:',/)
  490 FORMAT(5X,2I5,2E15.5)
  495 FORMAT(5X,2I5,2X,2I5,/,5X,4E15.5)

  500 FORMAT(/,3X, 'Boundary information on secondary variables:',/)
  510 FORMAT(/,3X, 'Boundary information on mixed boundary cond.:',/)
  515 FORMAT(/,3X, 'Multi-point constraint information:',/)
  520 FORMAT(2X,78('_'),/)
  530 FORMAT(2X,55('_'),/)
  540 FORMAT(2X,5E13.5)
  545 FORMAT(/,3X,'Forces at the constrained points:',/)

  550 FORMAT(/,3X,'Element coefficient matrix, [ELK]:',/)
  560 FORMAT(/,3X,'Element source vector, {ELF}:',/)
  570 FORMAT(/,3X,'Global coefficient matrix, [GLK]:',/)
  575 FORMAT(/,3X,'Global coefficient matrix, [GLM]:',/)
  580 FORMAT(/,3X,'Global source vector, {GLF}:',/)
  590 FORMAT(/,1X,'SOLUTION (values of PVs) at the NODES: ',/)
  600 FORMAT(/,1X,'TIME =',E12.4,5X,'Time step number =',I3,/)
  610 FORMAT(7X,'  x  ',5X, 'P. Variable',2X,'S. Variable')
  620 FORMAT(7X,'  x  ',3X, 'Displacement',2X,'Radial Stress',2X,
     *      'Hoop Stress')
  630 FORMAT(/,15X,'Generalized internal forces in the element',/,
     * 5X,'(second line gives the results in the global coordinates)')
  640 FORMAT(/,3X,'*** THE SOLUTION HAS REACHED A STEADY STATE ***',
     *       /,3X,'SOLUTION AT THE TWO CONSECUTIVE TIME STEPS FOLLOWS:')
  645 FORMAT(/,2X,'FIRST TIME DERIVATIVE of the primary variables:',/)
  646 FORMAT(/,2X,'SECOND TIME DERIVATIVE of the primary variables:',/)
  647 FORMAT(3X,'x is the global coord. if ICONT=1 and it is the local',
     *       ' coord. if ICONT=0')
  650 FORMAT(7X,'  x  ',6X, 'Deflect.',5X,'Rotation',5X,'B. Moment',
     *      3X,'Shear Force')
  660 FORMAT(7X,'  x  ',6X, 'Deflect.',5X,'Rotation',4X,'Moment, Mr',
     *      3X,'Moment, Mt',3X,'Shear Force')
  670 FORMAT(3X,  'Ele  Force, H1   Force, V1  Moment, M1  Force, H2
     *Force, V2  Moment, M2')
  680 FORMAT(3X,  'Ele  Force, H1   Force, V1   Force, H2  Force, V2')
  690 FORMAT(/,5X,'Number of rotations taken in JACOBI =',I4,/)
  700 FORMAT(/,5X,'EIGENVALUE(',I2,') = ',E14.6,2X,'SQRT(EGNVAL) = ',
     *       E13.5,/,5X,'EIGENVECTOR:')
  710 FORMAT(/,5X,'***** Number of time steps exceeded NTIME *****',/)
      close(in)
      close(it)
      STOP
      END


      SUBROUTINE ASSEMBLE(NOD,MXELM,MXNEQ,NDF,NPE,NE,ITEM,GLK,GLM,GLF)
C
C
C       The subroutine is called in MAIN to assemble element coefficient
C       matrices (in a upper banded matrix form) and right-hand vectors
```

```
C
C        {ELF}.... Element source vector, {f}
C        {ELK}.... Element coefficient matrix, [K]
C        {ELM}.... Element coefficient matrix, [M]
C        [NOD].... Connectivity matrix, [B]
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  GLK(MXNEQ,MXNEQ),GLM(MXNEQ,MXNEQ),GLF(MXNEQ),
     *           NOD(MXELM,4)
      COMMON/STF1/ELK(9,9),ELM(9,9),ELF(9),ELX(4),ELU(9),ELV(9),ELA(9)
      IF(ITEM.LE.2)THEN
C
C     Assemble element coefficient matrix ELK and source vector ELF
C
         DO 50 I = 1, NPE
         NR = (NOD(NE,I) - 1)*NDF
         DO 40 II = 1, NDF
         NR = NR + 1
         L = (I-1)*NDF + II
         GLF(NR) = GLF(NR) + ELF(L)
         DO 30 J = 1, NPE
         NCL = (NOD(NE,J)-1)*NDF
         DO 20 JJ = 1, NDF
         M = (J-1)*NDF + JJ
         NC = NCL-NR+JJ+1
         IF(NC)20,20,10
   10    GLK(NR,NC) = GLK(NR,NC) + ELK(L,M)
   20    CONTINUE
   30    CONTINUE
   40    CONTINUE
   50    CONTINUE
      ELSE
C
C     ASSEMBLE ELEMENT MATRICES INTO FULL GLOBAL MATRICES
C
         DO 100 I=1,NPE
         NR=(NOD(NE,I)-1)*NDF
         DO 90 II=1,NDF
         NR=NR+1
         L=(I-1)*NDF+II
         DO 80 J=1,NPE
         NC=(NOD(NE,J)-1)*NDF
         DO 70 JJ=1,NDF
         M=(J-1)*NDF+JJ
         NC=NC+1
         GLK(NR,NC)=GLK(NR,NC)+ELK(L,M)
   60    GLM(NR,NC)=GLM(NR,NC)+ELM(L,M)
   70    CONTINUE
   80    CONTINUE
   90    CONTINUE
  100    CONTINUE
C
      ENDIF
      RETURN
      END


      SUBROUTINE BOUNDARY(NEQ,NEQR,NHBW,NSPV,NSSV,NNBC,NDF,DT,ITEM,ALFA,
     *                    IBDY,ISPV,ISSV,INBC,UREF,VSPV,VSSV,VNBC,
     *                    GLK,GLM,GLF,GU0,MXEBC,MXNBC,MXMBC,MXNEQ)
C
C
C     The subroutine is called in MAIN to implement specified boundary
C      conditions on the assembled system of finite element equations
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  ISPV(MXEBC,2),ISSV(MXNBC,2),INBC(MXMBC,2),IBDY(MXEBC)
      DIMENSION  UREF(MXMBC),VSPV(MXEBC),VSSV(MXNBC),VNBC(MXMBC)
      DIMENSION  GLK(MXNEQ,MXNEQ),GLM(MXNEQ,MXNEQ),GLF(MXNEQ),GU0(MXNEQ)
C
C     Impose boundary conditions for STATIC and TIME-DEPENDENT problems
C
      IF(ITEM.LE.2)THEN
C
```

```
C     Include specified PRIMARY degrees of freedom
C
         IF(NSPV.NE.0)THEN
            DO 30 NB = 1,NSPV
            IE=(ISPV(NB,1)-1)*NDF+ISPV(NB,2)
            IT=NHBW-1
            I=IE-NHBW
            DO 10 II=1,IT
            I=I+1
            IF(I .GE. 1)THEN
               J=IE-I+1
               GLF(I)=GLF(I)-GLK(I,J)*VSPV(NB)
               GLK(I,J)=0.0
            ENDIF
   10       CONTINUE
            GLK(IE,1)=1.0
            GLF(IE)=VSPV(NB)
            I=IE
            DO 20 II=2,NHBW
            I=I+1
            IF(I .LE. NEQ)THEN
               GLF(I)=GLF(I)-GLK(IE,II)*VSPV(NB)
               GLK(IE,II)=0.0
            ENDIF
   20       CONTINUE
   30       CONTINUE
        ENDIF
C
        IF(NSSV.NE.0)THEN
C
C     Include specified SECONDARY degrees of freedom
C
           DO 40 NF = 1,NSSV
           NB=(ISSV(NF,1)-1)*NDF+ISSV(NF,2)
           IF(ITEM.EQ.1)GLF(NB)=GLF(NB)+VSSV(NF)*DT
   40      IF(ITEM.NE.1)GLF(NB)=GLF(NB)+VSSV(NF)
        ENDIF
C
        IF(NNBC.NE.0)THEN
C
C     Include specified MIXED boundary conditions
C
           DO 50 IC=1,NNBC
           NC=(INBC(IC,1)-1)*NDF+INBC(IC,2)
           IF(ITEM.EQ.1)THEN
              GLK(NC,1)=GLK(NC,1)+ALFA*DT*VNBC(IC)
              GLF(NC)=GLF(NC)+DT*VNBC(IC)*(UREF(IC)
     *                       -(1.0-ALFA)*GU0(NC))
           ELSE
              GLK(NC,1)=GLK(NC,1)+VNBC(IC)
              GLF(NC)=GLF(NC)+VNBC(IC)*UREF(IC)
           ENDIF
   50      CONTINUE
        ENDIF
      ELSE
C
C     Impose boundary conditions for EIGENVALUE problems
C
        IF(NNBC.NE.0)THEN
C
C     Include specified MIXED boundary conditions
C
           DO 70 IC=1,NNBC
           NC=(INBC(IC,1)-1)*NDF+INBC(IC,2)
           GLK(NC,NC)=GLK(NC,NC)+VNBC(IC)
   70      CONTINUE
        ENDIF
C
C     Include specified PRIMARY degrees of freedom
C
        IF(NSPV.NE.0)THEN
           DO 80 IB=1,NSPV
   80      IBDY(IB)=(ISPV(IB,1)-1)*NDF+ISPV(IB,2)
           DO 120 I=1,NSPV
           IMAX=IBDY(I)
           DO 110 J=I,NSPV
```

```
           IF(IBDY(J).GE.IMAX)THEN
              IMAX=IBDY(J)
              IKEPT=J
           ENDIF
  110      CONTINUE
           IBDY(IKEPT)=IBDY(I)
           IBDY(I)=IMAX
  120      CONTINUE
           NEQR = NEQ
           DO 180 I=1,NSPV
           IB=IBDY(I)
           IF(IB .LT. NEQR)THEN
              NEQR1=NEQR-1
              DO 160 II=IB,NEQR1
              DO 140 JJ=1,NEQR
              GLM(II,JJ)=GLM(II+1,JJ)
  140         GLK(II,JJ)=GLK(II+1,JJ)
              DO 150 JJ=1,NEQR
              GLM(JJ,II)=GLM(JJ,II+1)
  150         GLK(JJ,II)=GLK(JJ,II+1)
  160         CONTINUE
           ENDIF
           NEQR=NEQR-1
  180      CONTINUE
        ENDIF
      ENDIF
      RETURN
      END


      SUBROUTINE COEFFCNT(IELEM,ITEM,MODEL,NDF,NPE,TIME,NTYPE,
     *                    NE,F3,MXELM)
C
C
C     The subroutine is called in MAIN to compute coefficient matrices
C      and source vector for the model problem in Eq. (1) (see MAIN)
C
C        X.........  Global (i.e., problem) coordinate
C        XI .......  Local (i.e., element) coordinate
C        H.........  Element length
C        {SF}......  Element interpolation (or shape) functions
C        {GDSF}....  First derivative of SF w.r.t. X
C        {GDDSF}...  Second derivative of SF w.r.t. X
C        GJ........  Determinant of the Jacobian matrix [J]
C        [GAUSPT]..  4x4 matrix of Gauss points: N-th column corresponds
C                    to the N-point Gauss rule
C        [GAUSWT]..  4x4 matrix of Gauss weights (see the comment above)
C        [A],[B],..  Element matrices needed to compute ELK
C        [ELK].....  Element coefficient matrix [K]
C        [ELM].....  Element 'mass' matrix [M]
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/STF1/ELK(9,9),ELM(9,9),ELF(9),ELX(4),ELU(9),ELV(9),ELA(9)
      COMMON/STF2/A1,A2,A3,A4,A5,AX0,AX1,BX0,BX1,CX0,CX1,CT0,CT1,FX0,
     *            FX1,FX2
      COMMON/SHP/SF(4),GDSF(4),GDDSF(4),GJ
      DIMENSION GAUSPT(5,5),GAUSWT(5,5),F3(MXELM)
C
      DATA GAUSPT/5*0.0D0,-.57735027D0,.57735027D0,3*0.0D0,-.77459667D0,
     * 0.0D0,.77459667D0,2*0.0D0,-.86113631D0,-.33998104D0,.33998104D0,
     *.86113631D0,0.0D0,-.906180D0,-.538469D0,0.0D0,.538469D0,.906180D0/
C
      DATA GAUSWT/2.0D0,4*0.0D0,2*1.0D0,3*0.0D0,.55555555D0,.88888888D0,
     * 0.55555555D0,2*0.0D0,.34785485D0,2*.65214515D0,.34785485D0,0.0D0,
     * 0.236927D0,.478629D0,.568889D0,.478629D0,.236927D0/
C
      NN=NDF*NPE
      H = ELX(NPE) - ELX(1)
      IF(IELEM .EQ. 0)THEN
         NGP=4
      ELSE
         NGP = IELEM+1
      ENDIF
C
      DO 10 J=1,NN
```

```
      IF(ITEM.LE.2)THEN
         ELF(J) = 0.0
      ENDIF
      DO 10 I=1,NN
      IF(ITEM.GT.0)THEN
         ELM(I,J)=0.0
      ENDIF
   10 ELK(I,J)=0.0
C
      IF(MODEL.NE.2)THEN
C
C     DO-LOOP on number of Gauss points begins here
C
         DO 100 NI=1,NGP
         XI = GAUSPT(NI,NGP)
C
C     Call subroutine SHAPE1D to evaluate the interpolation functions
C          and their global derivatives at the Gauss point XI
C
         CALL SHAPE1D(H,IELEM,NPE,XI)
         CONST = GJ*GAUSWT(NI,NGP)
         IF(IELEM.EQ.0)THEN
            X = ELX(1) + 0.5*H*(1.0+XI)
         ELSE
            X = 0.0
            DO 30 J=1,NPE
   30       X = X + SF(J)*ELX(J)
         ENDIF
C
C     Compute coefficient matrices and vectors for vaious model problems
C         governed by single second-order and fourth-order equations
C                      (MODEL = 1 or 3; NTYPE = 0 or 1)
C
         CX=CX0+CX1*X
         IF(ITEM.NE.3) THEN
            FX=FX0+FX1*X+FX2*X*X
         ENDIF
         IF(ITEM.GT.0)THEN
            CT=CT0+CT1*X
         ENDIF
         IF(MODEL.EQ.1)THEN
C
C     Coefficients for ALL SINGLE-VARIABLE PROBLEMS (MODEL=1)
C
            IF(NTYPE.EQ.0)THEN
C
C     All problems governed by MODEL EQUATION (3.1) (NTYPE=0)
C
               AX=AX0+AX1*X
               DO 50 J = 1,NN
               IF(ITEM.LE.2)THEN
                  ELF(J) = ELF(J) + CONST*SF(J)*FX
               ENDIF
               DO 50 I = 1,NN
               IF(ITEM.NE.0)THEN
                  ELM(I,J) = ELM(I,J) + CONST*SF(I)*SF(J)*CT
               ENDIF
               AIJ = CONST*GDSF(I)*GDSF(J)
               CIJ = CONST*SF(I)*SF(J)
   50          ELK(I,J)=ELK(I,J) + AX*AIJ + CX*CIJ
            ELSE
C
C     RADIALLY SYMMETRIC ELASTICITY problems (MODEL=1, NTYPE>0)
C            AX0=E1, AX1=E2, BX0=NU12, BX1=H, thickness
C
               ANU21=BX0*AX0/AX1
               IF(NTYPE.EQ.1)THEN
                  C11=BX1*AX0/(1.0-BX0*ANU21)
                  C22=C11*(AX1/AX0)
                  C12=BX0*C22
               ELSE
                  DENOM=1.0-BX0-ANU21
                  C11=BX1*AX0*(1.0-BX0)/(1.0+BX0)/DENOM
                  C22=BX1*AX1*(1.0-ANU21)/(1.0+ANU21)/DENOM
                  C12=BX0*C22
               ENDIF
```

```
            DO 60 J=1,NN
            IF(ITEM.LE.2)THEN
               ELF(J) = ELF(J) + CONST*SF(J)*FX*X
            ENDIF
            DO 60 I=1,NN
            IF(ITEM.NE.0)THEN
               ELM(I,J) = ELM(I,J) + CONST*SF(I)*SF(J)*CT*X
            ENDIF
            AIJ = CONST*GDSF(I)*GDSF(J)*C11*X
            CIJ = CONST*SF(I)*SF(J)*CX*X
            DIJ = CONST*(GDSF(I)*SF(J)+SF(I)*GDSF(J))*C12
            EIJ = CONST*SF(I)*SF(J)*C22/X
   60       ELK(I,J)=ELK(I,J) + AIJ + CIJ + DIJ + EIJ
         ENDIF
       ELSE
C
C     Coefficients for the EULER-BERNOULLI theory (MODEL=2)
C
         IF(NTYPE.EQ.0)THEN
C
C     The Euler-Bernoulli BEAM element (MODEL=1 and NTYPE=0)
C
            BX=BX0+BX1*X
            CX=CX0+CX1*X
            DO 70 J = 1,NN
            IF(ITEM.LE.2)THEN
               ELF(J) = ELF(J) + CONST*SF(J)*FX
            ENDIF
            DO 70 I = 1,NN
            IF(ITEM.GT.0)THEN
               IF(ITEM.LE.3)THEN
                  ELM(I,J) = ELM(I,J) + CONST*SF(I)*SF(J)*CT
               ELSE
                  ELM(I,J) = ELM(I,J) + CONST*GDSF(I)*GDSF(J)
               ENDIF
            ENDIF
            BIJ = CONST*GDDSF(I)*GDDSF(J)
            CIJ = CONST*SF(I)*SF(J)
   70       ELK(I,J)=ELK(I,J) + BX*BIJ + CX*CIJ
         ELSE
C
C     The E-B CIRCULAR PLATE element (MODEL=1 and NTYPE>0)
C
            ANU21=BX0*AX0/AX1
            DI=(BX1**3)/12.0
            D11=DI*AX0/(1.0-BX0*ANU21)
            D22=D11*(AX1/AX0)
            D12=BX0*D22
            DO 80 J=1,NN
            IF(ITEM.LE.2)THEN
               ELF(J) = ELF(J) + CONST*SF(J)*FX*X
            ENDIF
            DO 80 I=1,NN
            BIJ = CONST*GDDSF(I)*GDDSF(J)*D11*X
            CIJ = CONST*SF(I)*SF(J)*CX*X
            DIJ = CONST*(GDDSF(I)*GDSF(J)+GDSF(I)*GDDSF(J))*D12
            EIJ = CONST*GDSF(I)*GDSF(J)*D22/X
   80       ELK(I,J)=ELK(I,J) + BIJ + CIJ + DIJ + EIJ
         ENDIF
       ENDIF
  100  CONTINUE
      ELSE
C
C     Coefficients for the TIMOSHENKO beam and circular plate (MODEL=2)
C         Full integration for bending coefficients
C
       DO 160 NI=1,NGP
       XI=GAUSPT(NI,NGP)
       CALL SHAPE1D(H,IELEM,NPE,XI)
       CONST=GJ*GAUSWT(NI,NGP)
       X = 0.0
       DO 110 J=1,NPE
  110  X = X + SF(J)*ELX(J)
       IF(NTYPE.EQ.0 .OR. NTYPE.EQ.2)THEN
C
C         The Timoshenko BEAM element (MODEL=2 and NTYPE=0 OR 2)
```

```
C
           BX=BX0+BX1*X
           CX=CX0+CX1*X
           FX=FX0+FX1*X+FX2*X*X
           JJ=1
           DO 130 J=1,NPE
           IF(ITEM.LE.2)THEN
              ELF(JJ)=ELF(JJ)+FX*SF(J)*CONST
           ENDIF
           II=1
           DO 120 I=1,NPE
           CIJ=SF(I)*SF(J)*CONST
           BIJ=GDSF(I)*GDSF(J)*CONST
           ELK(II,JJ)    =ELK(II,JJ)    +CX*CIJ
           ELK(II+1,JJ+1)=ELK(II+1,JJ+1)+BX*BIJ
           IF(ITEM.NE.0)THEN
              ELM(II,JJ)    =ELM(II,JJ)    +CT0*CIJ
              ELM(II+1,JJ+1)=ELM(II+1,JJ+1)+CT1*CIJ
           ENDIF
  120      II=NDF*I+1
  130      JJ=NDF*J+1
        ELSE
C
C     Timoshenko CIRCULAR PLATE element (MODEL=2 and NTYPE=1 or 3)
C                  AX0=E1, AX1=E2, BX0=ANU12, BX1=H
C
           ANU21=BX0*AX0/AX1
           CX=CX0+CX1*X
           FX=FX0+FX1*X
           DI=(BX1**3)/12.0
           D11=DI*AX0/(1.0-BX0*ANU21)
           D22=D11*(AX1/AX0)
           D12=BX0*D22
           JJ=1
           DO 150 J=1,NPE
           IF(ITEM.LE.2)THEN
              ELF(JJ)=ELF(JJ)+FX*SF(J)*CONST*X
           ENDIF
           II=1
           DO 140 I=1,NPE
           BIJ = CONST*GDSF(I)*GDSF(J)*D11*X
           CIJ = CONST*SF(I)*SF(J)*X
           DIJ = CONST*(GDSF(I)*SF(J)+SF(I)*GDSF(J))*D12
           EIJ = CONST*SF(I)*SF(J)*D22/X
           ELK(II,JJ)    =ELK(II,JJ)     + CX*CIJ
           ELK(II+1,JJ+1)=ELK(II+1,JJ+1) + BIJ + DIJ + EIJ
           IF(ITEM.NE.0)THEN
              ELM(II,JJ)    =ELM(II,JJ)    +CT0*CIJ
              ELM(II+1,JJ+1)=ELM(II+1,JJ+1)+CT1*CIJ
           ENDIF
  140      II=NDF*I+1
  150      JJ=NDF*J+1
        ENDIF
  160   CONTINUE
C
C     Reduced integration is used to evaluate the transverse shear terms
C
        LGP=NGP-1
        DO 230 NI=1,LGP
        XI=GAUSPT(NI,LGP)
C
        CALL SHAPE1D(H,IELEM,NPE,XI)
        CONST=GJ*GAUSWT(NI,LGP)
C
        X = 0.0
        DO 170 J=1,NPE
  170   X = X + SF(J)*ELX(J)
        IF(NTYPE.EQ.0 .OR. NTYPE.EQ.2)THEN
C
C         The Timoshenko BEAM element (MODEL=2 and NTYPE=0 or 2)
C         AX = GAK = AX0 + AX1*X  (reduced integration)
C
           AX=AX0+AX1*X
           JJ=1
           DO 190 J=1,NPE
           II=1
```

```
          DO 180 I=1,NPE
          B11=GDSF(I)*GDSF(J)*CONST
          B01=SF(I)*GDSF(J)*CONST
          B10=GDSF(I)*SF(J)*CONST
          B00=SF(I)*SF(J)*CONST
          ELK(II,JJ)    =ELK(II,JJ)    +AX*B11
          ELK(II,JJ+1)  =ELK(II,JJ+1)  +AX*B10
          ELK(II+1,JJ)  =ELK(II+1,JJ)  +AX*B01
          ELK(II+1,JJ+1)=ELK(II+1,JJ+1)+AX*B00
  180     II=I*NDF+1
  190     JJ=J*NDF+1
       ELSE
C
C     Timoshenko CIRCULAR PLATE element (MODEL=2 and NTYPE=1 or 3)
C                BX1=H, FX2=G13*K (reduced integration)
C
          A33=BX1*FX2
          JJ=1
          DO 210 J=1,NPE
          II=1
          DO 200 I=1,NPE
          BIJ = CONST*GDSF(I)*GDSF(J)*X
          CIJ = CONST*SF(I)*SF(J)*X
          DIJ = CONST*GDSF(I)*SF(J)*X
          DJI = CONST*SF(I)*GDSF(J)*X
          ELK(II,JJ)    =ELK(II,JJ)     + A33*BIJ
          ELK(II,JJ+1)  =ELK(II,JJ+1)   + A33*DIJ
          ELK(II+1,JJ)  =ELK(II+1,JJ)   + A33*DJI
          ELK(II+1,JJ+1)=ELK(II+1,JJ+1) + A33*CIJ
  200     II=NDF*I+1
  210     JJ=NDF*J+1
       ENDIF
  230  CONTINUE
       IF(ITEM.EQ.0 .AND. NTYPE.GT.1)THEN
          CALL TIMFORCE(ELF,ELX,FX0,FX1,FX2,H,NTYPE,NE,F3,MXELM)
       ENDIF
      ENDIF
C
      IF(ITEM.GT.2)RETURN
        IF(ITEM.EQ.1 .OR. ITEM.EQ.2)THEN
C
C     Equivalent coefficient matrices for TIME-DEPENDENT problems
C
           IF(ITEM .EQ. 1)THEN
C
C     Alfa-family of time approximation for PARABOLIC equations
C
              DO 250 J=1,NN
              SUM=0.0
              DO 240 I=1,NN
              SUM=SUM+(ELM(I,J)-A2*ELK(I,J))*ELU(I)
  240         ELK(I,J)=ELM(I,J)+A1*ELK(I,J)
  250         ELF(J)=(A1+A2)*ELF(J)+SUM
           ELSE
C
C     Newmark-family of approximation for HYPERBOLIC equations
C
           IF(TIME.EQ.0.0)THEN
              DO 260 J=1,NN
              DO 260 I=1,NN
              ELF(J)=ELF(J)-ELK(I,J)*ELU(I)
  260         ELK(I,J)=ELM(I,J)
           ELSE
              DO 270 J=1,NN
              DO 270 I=1,NN
              ELF(J)=ELF(J)+ELM(I,J)*(A3*ELU(I)+A4*ELV(I)+A5*ELA(I))
  270         ELK(I,J)=ELK(I,J)+A3*ELM(I,J)
           ENDIF
        ENDIF
      ENDIF
      RETURN
      END


      SUBROUTINE CONSTRNT(NEQ,NHBW,NDF,NCON,ICON,VCON,GLK,GLM,GLF,
     *                    TRM,MXNEQ)
```

```
C
C
C     The subroutine is called in MAIN to implement specified constraint
C      conditions (e.g., inclined supports) on the condensed system of
C      equations. Array GLM is used here as a temporary storage array.
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  ICON(9),VCON(9),GLK(MXNEQ,MXNEQ),GLF(MXNEQ),
     *           GLM(MXNEQ,MXNEQ),TRM(MXNEQ,MXNEQ)
C
      PI=3.14159265D0
C
C     Include specified constraint conditions
C
      DO 20 IC=1,NEQ
         DO 10 JC=1,NEQ
            GLM(IC,JC)=0.0
   10       TRM(IC,JC)=0.0
   20    TRM(IC,IC)=1.0D0
      DO 30 IC=1,NCON
         BETA=VCON(IC)*PI/180.0D0
         IDOF=NDF*ICON(IC)-1
         TRM(IDOF,IDOF)     = DCOS(BETA)
         TRM(IDOF,IDOF+1)   = DSIN(BETA)
         TRM(IDOF+1,IDOF)   =-DSIN(BETA)
   30    TRM(IDOF+1,IDOF+1)= DCOS(BETA)
C
      L=0
      DO 50 I=1,NEQ
      DO 40 J=1,NHBW
   40 GLM(I,L+J)=GLK(I,J)
   50 L=L+1
      DO 60 I=1,NEQ
      DO 60 J=I,NEQ
   60 GLM(J,I)=GLM(I,J)
      DO 70 I=1,NEQ
      DO 70 J=1,NEQ
   70 GLK(I,J)=GLM(I,J)
C
      DO 80 I=1,NEQ
      DO 80 J=1,NEQ
      GLM(I,J)=0.0
      DO 80 K=1,NEQ
   80 GLM(I,J)=GLM(I,J)+TRM(I,K)*GLK(K,J)
C
      DO 90 I=1,NEQ
      DO 90 J=1,NEQ
      GLK(I,J)=0.0
      DO 90 K=1,NEQ
   90 GLK(I,J)=GLK(I,J)+GLM(I,K)*TRM(J,K)
C
      DO 100 I=1,NEQ
      DO 100 J=1,NEQ
  100 TRM(I,J)=GLK(I,J)
      L=0
      DO 120 I=1,NEQ
      DO 110 J=1,NHBW
  110 GLK(I,J)=TRM(I,L+J)
  120 L=L+1
C
      DO 150 I=1,NEQ
      GLM(I,1)=0.0
      DO 140 K=1,NEQ
  140 GLM(I,1)=GLM(I,1)+TRM(I,K)*GLF(K)
  150 GLF(I)=GLM(I,1)
C
      RETURN
      END


      SUBROUTINE ECHODATA(IN,IT)
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION AA(20)
      WRITE(IT,40)
```

```
   10 CONTINUE
      READ(IN,30,END=20) AA
      WRITE(IT,60) AA
      GO TO 10
   20 CONTINUE
      REWIND(IN)
      WRITE(IT,50)
      RETURN
   30 FORMAT(20A4)
   40 FORMAT(5X,'*** ECHO OF THE INPUT DATA STARTS ***',/)
   50 FORMAT(5X,'**** ECHO OF THE INPUT DATA ENDS ****',/)
   60 FORMAT(1X,20A4)
      END


      SUBROUTINE EGNSOLVR(N,A,B,XX,X,NEGN,NR,MXNEQ)
C
C
C      The subroutine is called in MAIN to solve the EIGENVALUE PROBLEM
C
C                           [A]{X} = Lambda.[B]{X}
C
C       The program can be used only for positive-definite [B] matrix.
C       The dimensions of V, VT, W, and IH should be equal to MXNEQ.
C
C     ________________________________________________________________
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  A(MXNEQ,MXNEQ),B(MXNEQ,MXNEQ),XX(MXNEQ),X(MXNEQ,MXNEQ)
      DIMENSION  V(500,500),VT(500,500),W(500,500),IH(500)
C
C     Call subroutine JACOBI to diagonalize [B]
C
      CALL JACOBI (N,B,NEGN,NR,V,XX,IH,MXNEQ)
C
C     Make diagonalized [B] symmetric
C
      DO 10 I=1,N
      DO 10 J=1,N
   10 B(J,I)=B(I,J)
C
C     Check (to make sure) that [B] is positive-definite
C
      DO 30 I=1,N
      IF (B(I,I))20,30,30
   20 WRITE(6,80)
      STOP
   30 CONTINUE
C
C     The eigenvectors of [B] are stored in array V(I,J)
C     Form the transpose of [V] as [VT]
C
      DO 40 I=1,N
      DO 40 J=1,N
   40 VT(I,J)=V(J,I)
C
C     Find the product [F]=[VT][A][V] and store in [A] to save storage
C
      CALL MATRXMLT (MXNEQ,N,VT,A,W)
      CALL MATRXMLT (MXNEQ,N,W,V,A)
C
C     Get [GI] from diagonalized [B], but store it in [B]
C
      DO 50 I=1,N
   50 B(I,I)=1.0/DSQRT(B(I,I))
C
C     Find the product [Q]=[GI][F][GI]=[B][A][B] and store in [A]
C
      CALL MATRXMLT (MXNEQ,N,B,A,W)
      CALL MATRXMLT (MXNEQ,N,W,B,A)
C
C     We now have the form [Q]{Z}=Lamda{Z}. Diagonalize [Q] to obtain
C     the eigenvalues by calling JACOBI.
C
      CALL JACOBI (N,A,NEGN,NR,VT,XX,IH,MXNEQ)
C
C     The eigenvalues are returned as diag [A].
```

```
C
      DO 60 J=1,N
   60 XX(J)=A(J,J)
C
C     The eigenvectors are computed from the relation,
C                      {X}=[V][GI]{Z}=[V][B][VT]
C     since {Z} is stored in [VT].
C
      CALL MATRXMLT (MXNEQ,N,V,B,W)
      CALL MATRXMLT (MXNEQ,N,W,VT,X)
C
   80 FORMAT(/'*** Matrix [GLM] is NOT positive-definite ***')
      RETURN
      END



      SUBROUTINE EQNSOLVR(NRM,NCM,NEQNS,NBW,BAND,RHS,IRES)
C
C
C     The subroutine is called in MAIN to solve symmetric and banded set
C     of equations using the Gauss elimination method:[BAND]{U} = {RHS}.
C     The coefficient matrix is input as BAND(NEQNS,NBW) and the column
C     vector is input as  RHS(NEQNS),  where NEQNS is the actual number
C     of equations and NBW is the half band width.  The true dimensions
C     of the matrix [BAND] in the calling program, are NRM by NCM. When
C     IRES is greater than zero, the right hand elimination is skipped.
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION BAND(NRM,NCM),RHS(NRM)
C
      MEQNS=NEQNS-1
      IF(IRES.LE.0) THEN
         DO 30 NPIV=1,MEQNS
         NPIVOT=NPIV+1
         LSTSUB=NPIV+NBW-1
         IF(LSTSUB.GT.NEQNS) THEN
            LSTSUB=NEQNS
         ENDIF
C
         DO 20 NROW=NPIVOT,LSTSUB
         NCOL=NROW-NPIV+1
         FACTOR=BAND(NPIV,NCOL)/BAND(NPIV,1)
         DO 10 NCOL=NROW,LSTSUB
         ICOL=NCOL-NROW+1
         JCOL=NCOL-NPIV+1
   10    BAND(NROW,ICOL)=BAND(NROW,ICOL)-FACTOR*BAND(NPIV,JCOL)
   20    RHS(NROW)=RHS(NROW)-FACTOR*RHS(NPIV)
   30    CONTINUE
      ELSE
   40    DO 60 NPIV=1,MEQNS
         NPIVOT=NPIV+1
         LSTSUB=NPIV+NBW-1
         IF(LSTSUB.GT.NEQNS) THEN
            LSTSUB=NEQNS
         ENDIF
         DO 50 NROW=NPIVOT,LSTSUB
         NCOL=NROW-NPIV+1
         FACTOR=BAND(NPIV,NCOL)/BAND(NPIV,1)
   50    RHS(NROW)=RHS(NROW)-FACTOR*RHS(NPIV)
   60    CONTINUE
      ENDIF
C
C     Back substitution
C
      DO 90 IJK=2,NEQNS
      NPIV=NEQNS-IJK+2
      RHS(NPIV)=RHS(NPIV)/BAND(NPIV,1)
      LSTSUB=NPIV-NBW+1
      IF(LSTSUB.LT.1) THEN
         LSTSUB=1
      ENDIF
      NPIVOT=NPIV-1
      DO 80 JKI=LSTSUB,NPIVOT
      NROW=NPIVOT-JKI+LSTSUB
```

```
      NCOL=NPIV-NROW+1
      FACTOR=BAND(NROW,NCOL)
   80 RHS(NROW)=RHS(NROW)-FACTOR*RHS(NPIV)
   90 CONTINUE
      RHS(1)=RHS(1)/BAND(1,1)
      RETURN
      END



      SUBROUTINE JACOBI (N,Q,JVEC,M,V,X,IH,MXNEQ)
C
C
C       Called in EGNSOLVR to diagonalize [Q] by successive rotations
C
C       DESCRIPTION OF THE VARIABLES:
C
C         N   .... Order of the real, symmetric matrix [Q] (N > 2)
C        [Q]  .... The matrix to be diagonalized (destroyed)
C        JVEC .... 0, when only eigenvalues alone have to be found
C        [V]  .... Matrix of eigenvectors
C         M   .... Number of rotations performed
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Q(MXNEQ,MXNEQ),V(MXNEQ,MXNEQ),X(MXNEQ),IH(MXNEQ)
      EPSI=1.0D-08
C
      IF(JVEC)10,50,10
   10 DO 40 I=1,N
      DO 40 J=1,N
      IF(I-J)30,20,30
   20 V(I,J)=1.0
      GO TO 40
   30 V(I,J)=0.0
   40 CONTINUE
   50 M=0
      MI=N-1
      DO 70 I=1,MI
      X(I)=0.0
      MJ=I+1
      DO 70 J=MJ,N
      IF(X(I)-DABS(Q(I,J)))60,60,70
   60 X(I)=DABS(Q(I,J))
      IH(I)=J
   70 CONTINUE
   75 DO 100 I=1,MI
      IF(I-1)90,90,80
   80 IF(XMAX-X(I))90,100,100
   90 XMAX=X(I)
      IP=I
      JP=IH(I)
  100 CONTINUE
      IF(XMAX-EPSI)500,500,110
  110 M=M+1
      IF(Q(IP,IP)-Q(JP,JP))120,130,130
  120 TANG=-2.0*Q(IP,JP)/(DABS(Q(IP,IP)-Q(JP,JP))+DSQRT((Q(IP,IP)
     1     -Q(JP,JP))**2+4.0*Q(IP,JP)**2))
      GO TO 140
  130 TANG= 2.0*Q(IP,JP)/(DABS(Q(IP,IP)-Q(JP,JP))+DSQRT((Q(IP,IP)
     1     -Q(JP,JP))**2+4.0*Q(IP,JP)**2))
  140 COSN=1.0/DSQRT(1.0+TANG**2)
      SINE=TANG*COSN
      QII=Q(IP,IP)
      Q(IP,IP)=COSN**2*(QII+TANG*(2.*Q(IP,JP)+TANG*Q(JP,JP)))
      Q(JP,JP)=COSN**2*(Q(JP,JP)-TANG*(2.*Q(IP,JP)-TANG*QII))
      Q(IP,JP)=0.0
      IF (Q(IP,IP)-Q(JP,JP)) 150,190,190
  150 TEMP=Q(IP,IP)
      Q(IP,IP)=Q(JP,JP)
      Q(JP,JP)=TEMP
      IF(SINE) 160,170,170
  160 TEMP=COSN
      GOTO 180
  170 TEMP=-COSN
  180 COSN=DABS(SINE)
```

```
      SINE=TEMP
  190 DO 260 I=1,MI
      IF (I-IP) 210,260,200
  200 IF (I-JP) 210,260,210
  210 IF (IH(I)-IP) 220,230,220
  220 IF (IH(I)-JP) 260,230,260
  230 K=IH(I)
      TEMP=Q(I,K)
      Q(I,K)=0.0
      MJ=I+1
      X(I)=0.0
      DO 250 J=MJ,N
      IF (X(I)-DABS(Q(I,J))) 240,240,250
  240 X(I)=DABS(Q(I,J))
      IH(I)=J
  250 CONTINUE
      Q(I,K)=TEMP
  260 CONTINUE
      X(IP)=0.0
      X(JP)=0.0
      DO 430 I=1,N
      IF(I-IP) 270,430,320
  270 TEMP=Q(I,IP)
      Q(I,IP)=COSN*TEMP+SINE*Q(I,JP)
      IF (X(I)-DABS(Q(I,IP))) 280,290,290
  280 X(I)=DABS(Q(I,IP))
      IH(I)=IP
  290 Q(I,JP)=-SINE*TEMP+COSN*Q(I,JP)
      IF (X(I)-DABS(Q(I,JP))) 300,430,430
  300 X(I)=DABS(Q(I,JP))
      IH(I)=JP
      GO TO 430
  320 IF(I-JP) 330,430,380
  330 TEMP=Q(IP,I)
      Q(IP,I)=COSN*TEMP+SINE*Q(I,JP)
      IF(X(IP)-DABS(Q(IP,I)))340,350,350
  340 X(IP)=DABS(Q(IP,I))
      IH(IP)=I
  350 Q(I,JP)=-SINE*TEMP+COSN*Q(I,JP)
      IF (X(I)-DABS(Q(I,JP))) 300,430,430
  380 TEMP=Q(IP,I)
      Q(IP,I)=COSN*TEMP+SINE*Q(JP,I)
      IF(X(IP)-DABS(Q(IP,I)))390,400,400
  390 X(IP)=DABS(Q(IP,I))
      IH(IP)=I
  400 Q(JP,I)=-SINE*TEMP+COSN*Q(JP,I)
      IF(X(JP)-DABS(Q(JP,I)))410,430,430
  410 X(JP)=DABS(Q(JP,I))
      IH(JP)=I
  430 CONTINUE
      IF(JVEC)440,75,440
  440 DO 450 I=1,N
      TEMP=V(I,IP)
      V(I,IP)=COSN*TEMP+SINE*V(I,JP)
  450 V(I,JP)=-SINE*TEMP+COSN*V(I,JP)
      GOTO 75
  500 RETURN
      END


      SUBROUTINE MATRXMLT(MXNEQ,N,A,B,C)
C
C
C     Called in EGNSOLVR to compute the product of matrices [A] & [B]:
C                                [C]=[A][B]
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(MXNEQ,MXNEQ),B(MXNEQ,MXNEQ),C(MXNEQ,MXNEQ)
      DO 10 I=1,N
      DO 10 J=1,N
      C(I,J)=0.0
      DO 10 K=1,N
   10 C(I,J)=C(I,J)+A(I,K)*B(K,J)
      RETURN
      END
```

```
      SUBROUTINE MESH1D(NEM,NPE,NOD,MXELM,MXNOD,DX,GLX)
C
C
C     The subroutine is called in MAIN to compute arrays {GLX} and [NOD]
C
C        {GLX}.... Vector of global coordinates
C        {DX}..... Vector of element lengths [DX(1) = node 1 coord.]
C        [NOD].... Connectivity matrix
C
C     ________________________________________________________________
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION GLX(MXNOD),DX(MXNOD),NOD(MXELM,4)
C
C     Generate the elements of the connectivity matrix
C
      DO 10 I=1,NPE
   10 NOD(1,I)=I
      DO 20 N=2,NEM
      DO 20 I=1,NPE
   20 NOD(N,I) = NOD(N-1,I)+NPE-1
C
C     Generate global coordinates of the global nodes
C
      GLX(1)=DX(1)
      IF(NPE.EQ.2)THEN
         DO 30 I=1,NEM
   30    GLX(I+1) = GLX(I) + DX(I+1)
      ELSE
         DO 40 I=1,NEM
         II=2*I
         GLX(II) = GLX(II-1) + 0.5*DX(I+1)
   40    GLX(II+1)=GLX(II-1) + DX(I+1)
      ENDIF
      RETURN
      END


      SUBROUTINE POSTPROC(DCAX,DCBX,DCCX,F3,GLF,GLX,NOD,ICONT,IELEM,NPE,
     *                    MODEL,NTYPE,ITEM,MXELM,MXNEQ,MXNOD,NEM,NDF)
C
C
C     The subroutine is called in MAIN to compute the solution and its
C     derivatives at  five points, including the nodes of the element.
C     The bending moment (BM) and shear force (VF) are computed as per
C     the definitions given in Fig. 5.2.1 and Eq. (5.3.1) of the book.

C
C        X........ Global (i.e., problem) coordinate
C        XI ...... Local (i.e., element) coordinate
C        SF....... Element interpolation (or shape) functions
C        GDSF..... First derivative of SF w.r.t. global coordinate
C        GDDSF.... Second derivative of SF w.r.t. global coordinate
C        ELU...... Element solution vector
C        U........ Interpolated solution
C        DU....... Interpolated derivative of the solution
C        W........ Interpolated transverse deflection
C        S........ Interpolated rotation function
C        DS....... Interpolated derivative of the rotation
C        DW....... Interpolated derivative of the transverse deflection
C        DDW...... Interpolated second derivative of trans. deflection
C        DDDW..... Interpolated third derivative of trans. deflection
C
C     ________________________________________________________________
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION DCAX(MXELM,2),DCBX(MXELM,2),DCCX(MXELM,2)
      DIMENSION F3(MXELM),GLF(MXNEQ),GLX(MXNOD),NOD(MXELM,4)
      DIMENSION XP(9),ELX(4),ELU(9)
      COMMON/IO/IN,IT
      COMMON/SHP/SF(4),GDSF(4),GDDSF(4),GJ
      COMMON/STF2/A1,A2,A3,A4,A5,AX0,AX1,BX0,BX1,CX0,CX1,CT0,CT1,FX0,
     *            FX1,FX2
      DATA XP/-1.0D0, -0.750D0, -0.50D0, -0.250D0, 0.0D0, 0.250D0,
     *                         0.50D0,  0.750D0, 1.0D0/
C
```

```
      NPTS=9
      DO 80 NE = 1, NEM
         IF(ICONT.NE.1) THEN
            AX0=DCAX(NE,1)
            AX1=DCAX(NE,2)
            BX0=DCBX(NE,1)
            BX1=DCBX(NE,2)
            CX0=DCCX(NE,1)
            CX1=DCCX(NE,2)
         ENDIF
      L=0
      DO 10 I=1,NPE
      NI=NOD(NE,I)
      IF(ICONT.NE.1)THEN
         ELX(1)=0.0
         ELX(2)=0.5*GLX(NE)
         ELX(NPE)=GLX(NE)
      ELSE
         ELX(I)=GLX(NI)
      ENDIF
      LI=(NI-1)*NDF
      DO 10 J=1,NDF
      LI=LI+1
      L=L+1
   10 ELU(L)=GLF(LI)
      H = ELX(NPE) - ELX(1)
C
      DO 70 NI=1,NPTS
      XI = XP(NI)
      CALL SHAPE1D(H,IELEM,NPE,XI)
      IF(MODEL.EQ.3)THEN
         W=0.0
         DW=0.0
         DDW=0.0
         XC=ELX(1)+0.5*H*(1.0+XI)
         DO 20 I=1,4
         W  =W  + SF(I)*ELU(I)
         DW =DW + GDSF(I)*ELU(I)
   20    DDW=DDW+ GDDSF(I)*ELU(I)
         DDDW=((ELU(1)-ELU(3))*2.0/H-(ELU(4)+ELU(2)))*6.0/(H*H)
         THETA=-DW
         IF(NTYPE.EQ.0)THEN
            BM=-(BX0+XC*BX1)*DDW
            VF=-(BX0+XC*BX1)*DDDW - BX1*DDW
            WRITE(IT,90)XC,W,THETA,BM,VF
         ELSE
            ANU21=BX0*AX0/AX1
            DI=(BX1**3)/12.0
            D11=DI*AX0/(1.0-BX0*ANU21)
            D22=D11*(AX1/AX0)
            D12=BX0*D22
            BMR=-(D11*DDW*XC+D12*DW)
            BMT=-(D12*DDW*XC+D22*DW)
            IF(XC.NE.0.0)THEN
               SFV=-D11*(XC*DDDW+DDW)+D22*DW/XC
               WRITE(IT,90)XC,W,THETA,BMR,BMT,SFV
            ELSE
               WRITE(IT,90)XC,W,THETA,BMR,BMT
            ENDIF
         ENDIF
      ELSE
         XC=0.0
         DO 30 I=1,NPE
   30    XC=XC+SF(I)*ELX(I)
         IF(MODEL.EQ.1)THEN
            U=0.0
            DU=0.0
            DO 40 I=1,NPE
            U=U+SF(I)*ELU(I)
   40       DU=DU+GDSF(I)*ELU(I)
            IF(NTYPE.EQ.0)THEN
               SV=(AX0+AX1*XC)*DU
               WRITE(IT,90)XC,U,SV
            ELSE
               ANU21=BX0*AX0/AX1
               IF(NTYPE.EQ.1)THEN
```

```
                 C11=BX1*AX0/(1.0-BX0*ANU21)
                 C22=C11*(AX1/AX0)
                 C12=BX0*C22
              ELSE
                 DENOM=1.0-BX0-ANU21
                 C11=BX1*AX0*(1.0-BX0)/(1.0+BX0)/DENOM
                 C22=BX1*AX1*(1.0-ANU21)/(1.0+ANU21)/DENOM
                 C12=BX0*C22
              ENDIF
              IF(XC.NE.0.0)THEN
                 SR=C11*DU+C12*U/XC
                 ST=C12*DU+C22*U/XC
                 WRITE(IT,90)XC,U,SR,ST
              ELSE
                 WRITE(IT,90)XC,U,DU
              ENDIF
           ENDIF
        ELSE
C
C     MODEL.EQ.2  Calculations
C
           IF(ITEM.EQ.0 .AND. NTYPE.GT.1)THEN
              H=ELX(NPE)-ELX(1)
              CALL TIMSTRES(AX0,ELU,XI,W,DW,PSI,DPSI,NE,F3,H,MXELM)
           ELSE
              W=0.0
              DW=0.0
              PSI = 0.0
              DPSI=0.0
              DO 50 I=1,NPE
              L=2*I-1
              W = W + SF(I)*ELU(L)
              DW=DW+GDSF(I)*ELU(L)
              PSI = PSI + SF(I)*ELU(L+1)
   50         DPSI=DPSI+GDSF(I)*ELU(L+1)
           ENDIF
           IF(NTYPE.EQ.0 .OR. NTYPE.EQ.2)THEN
              BM=(BX0+BX1*XC)*DPSI
              VF=(AX0+AX1*XC)*(DW+PSI)
              WRITE(IT,90)XC,W,PSI,BM,VF
           ELSE
              ANU21=BX0*AX0/AX1
              DI=(BX1**3)/12.0
              D11=DI*AX0/(1.0-BX0*ANU21)
              D22=D11*(AX1/AX0)
              D12=BX0*D22
              BMR=(D11*DPSI*XC+D12*PSI)
              BMT=(D12*DPSI*XC+D22*PSI)
              SFV=FX2*(DW+PSI)*XC
              WRITE(IT,90)XC,W,PSI,BMR,BMT,SFV
           ENDIF
        ENDIF
      ENDIF
   70 CONTINUE
   80 CONTINUE
      RETURN
   90 FORMAT(2X,6E13.5)
      END


      SUBROUTINE REACTION(MXELM,MXNEQ,NDF,NEM,NOD,NPE,NTYPE,PR,GLF,
     *                    SE,SL,SA,SI,CS,SN,CNT,SNT,HF,VF,PF,XB)
C
C
C      The subroutine is called in MAIN to compute generalized reaction
C     forces in each element of truss (NDF=2) or frame (NDF=3) structure
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION  PR(MXELM),SE(MXELM),SL(MXELM),SA(MXELM),SI(MXELM)
      DIMENSION  CS(MXELM),SN(MXELM),CNT(MXELM),SNT(MXELM)
      DIMENSION  HF(MXELM),VF(MXELM),PF(MXELM),XB(MXELM)
      DIMENSION  NOD(MXELM,4),GLF(MXNEQ),ELR(6)
      COMMON/STF1/ELK(9,9),ELM(9,9),ELF(9),ELX(4),ELU(9),ELV(9),ELA(9)
C
      NN=NPE*NDF
```

```
      DO 140 N=1,NEM
      CN1=CS(N)
      SN1=SN(N)
C
C     Call TRANSFRM to compute element stiffness matrix and force vector
C
      L=0
      DO 100 I=1,NPE
      NI=NOD(N,I)
      LI=(NI-1)*NDF
      DO 100 J=1,NDF
      LI=LI+1
      L=L+1
  100 ELU(L)=GLF(LI)
      CALL TRANSFRM(MXELM,N,NTYPE,PR,SE,SL,SA,SI,CS,SN,
     *              CNT,SNT,HF,VF,PF,XB)
C
C     Compute the FORCE and MOMENT RESULTANTS
C
      DO 120 I=1,NN
      ELR(I) = 0.0
      DO 110 J=1,NN
  110 ELR(I) =  ELR(I) + ELK(I,J)*ELU(J)
  120 ELR(I) =  ELR(I) - ELF(I)
      ELF(1) =  ELR(1)*CN1+ELR(2)*SN1
      ELF(2) = -ELR(1)*SN1+ELR(2)*CN1
      IF(NTYPE.NE.0) THEN
         ELF(3) =  ELR(3)
         ELF(4) =  ELR(4)*CN1+ELR(5)*SN1
         ELF(5) = -ELR(4)*SN1+ELR(5)*CN1
         ELF(6) =  ELR(6)
      ELSE
         ELF(3) =  ELR(3)*CN1+ELR(4)*SN1
         ELF(4) = -ELR(3)*SN1+ELR(4)*CN1
      ENDIF
      WRITE(6,150)N, (ELF(I),I=1,NN)
      WRITE(6,160)   (ELR(I),I=1,NN)
  140 CONTINUE
      RETURN
  150 FORMAT (3X,I2,6E12.4)
  160 FORMAT (5X,6E12.4,/)
      END


      SUBROUTINE SHAPE1D(H,IELEM,NPE,XI)
C     ______________________________________________________________________
C
C       Called in MAIN to compute shape functions and their derivatives
C     for Hermite cubic and Lagrange linear, quadratic and cubic elements
C
C        X.........  Global (i.e., problem) coordinate
C        XI .......  Local (i.e., element) coordinate
C        H.........  Element length
C        {SF}......  Interpolation (or shape) functions
C        {DSF}.....  First derivative of SF w.r.t. XI
C        {DDSF}....  Second derivative of SFH w.r.t. XI
C        {GDSF}....  First derivative of SF w.r.t. X
C        {GDDSF}...  Second derivative of SFH w.r.t. X
C        GJ........  Jacobian of the transformation
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/SHP/SF(4),GDSF(4),GDDSF(4),GJ
      DIMENSION DSF(4),DDSF(4)
      IF(IELEM.EQ.0)THEN
C
C     HERMITE interpolation functions (for the Euler-Bernoulli theory)
C
         NET=4
         SF(1)  = 0.25*(2.0-3.0*XI+XI**3)
         SF(2)  =  -H*(1.0-XI)*(1.0-XI*XI)/8.0
         SF(3)  = 0.25*(2.0+3.0*XI-XI**3)
         SF(4)  =   H*(1.0+XI)*(1.0-XI*XI)/8.0
         DSF(1) = -0.75*(1.0-XI*XI)
         DSF(2) =  H*(1.0+2.0*XI-3.0*XI*XI)/8.0
         DSF(3) =  0.75*(1.0-XI*XI)
```

```
         DSF(4) =  H*(1.0-2.0*XI-3.0*XI*XI)/8.0
         DDSF(1)= 1.5*XI
         DDSF(2)= 0.25*H*(1.0-3.0*XI)
         DDSF(3)= -1.5*XI
         DDSF(4)=-0.25*(1.0+3.0*XI)*H
      ELSE
         NET=NPE
         IF(IELEM.EQ.1)THEN
C
C     LAGRANGE interpolation functions used for linear, quadratic and
C              cubic approximation of second-order equations
C
C     LINEAR interpolation functions
C
            SF(1)  = 0.5*(1.0-XI)
            SF(2)  = 0.5*(1.0+XI)
            DSF(1) = -0.5
            DSF(2) = 0.5
            DDSF(1)= 0.0
            DDSF(2)= 0.0
         ELSE
            IF(IELEM.EQ.2)THEN
C
C     QUADRATIC interpolation functions
C
               SF(1)  = -0.5*XI*(1.0-XI)
               SF(2)  = 1.0-XI*XI
               SF(3)  = 0.5*XI*(1.0+XI)
               DSF(1) = -0.5*(1.0-2.0*XI)
               DSF(2) = -2.0*XI
               DSF(3) = 0.5*(1.0+2.0*XI)
               DDSF(1)= 1.0
               DDSF(2)= -2.0
               DDSF(3)= 1.0
            ELSE
C
C     CUBIC interpolation functions
C
               SF(1)  = 0.0625*(1.0-XI)*(9.0*XI*XI-1.)
               SF(2)  = 0.5625*(1.0-XI*XI)*(1.0-3.0*XI)
               SF(3)  = 0.5625*(1.0-XI*XI)*(1.0+3.0*XI)
               SF(4)  = 0.0625*(9.0*XI*XI-1.0)*(1.0+XI)
               DSF(1) = 0.0625*(1.0+18.0*XI-27.0*XI*XI)
               DSF(2) = 0.5625*(-3.0-2.0*XI+9.0*XI*XI)
               DSF(3) = 0.5625*(3.0-2.0*XI-9.0*XI*XI)
               DSF(4) = 0.0625*(18.0*XI+27.0*XI*XI-1.0)
               DDSF(1)= 0.0625*(18.0-54.0*XI)
               DDSF(2)= 0.5625*(-2.0+18.0*XI)
               DDSF(3)= 0.5625*(-2.0-18.0*XI)
               DDSF(4)= 0.0625*(18.0+54.0*XI)
            ENDIF
         ENDIF
      ENDIF
C
C     Compute derivatives of the interpolation functions w.r.t. X
C
   80 GJ =  H*0.5
      DO 90 I  = 1,NET
      GDSF(I)  = DSF(I)/GJ
   90 GDDSF(I) = DDSF(I)/GJ/GJ
      RETURN
      END


      SUBROUTINE TIMFORCE(ELF,ELX,FX0,FX1,FX2,H,NTYPE,NE,F3,MXELM)
C
C
C        Called in COEFFCNT to compute element force vector for the
C            consistent interpolation Timoshenko element (CIE)
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/SHP/SF(4),GDSF(4),GDDSF(4),GJ
      DIMENSION GAUSPT(5,5),GAUSWT(5,5),ELF(9),ELX(4),EX(3),F3(MXELM)
C
      DATA GAUSPT/5*0.0D0,-.57735027D0,.57735027D0,3*0.0D0,-.77459667D0,
```

```
     * 0.0D0,.77459667D0,2*0.0D0,-.86113631D0,-.33998104D0,.33998104D0,
     *.86113631D0,0.0D0,-.906180D0,-.538469D0,0.0D0,.538469D0,.906180D0/
C
      DATA GAUSWT/2.0D0,4*0.0D0,2*1.0D0,3*0.0D0,.55555555D0,.88888888D0,
     * 0.55555555D0,2*0.0D0,.34785485D0,2*.65214515D0,.34785485D0,0.0D0,
     * 0.236927D0,.478629D0,.568889D0,.478629D0,.236927D0/
C
      NPE=3
      IEL=2
      NDF=2
      NGP=IEL+1
      DO 10 I=1,6
10    ELF(I)=0.0
C
      EX(1)=ELX(1)
      EX(2)=ELX(1)+0.5*H
      EX(3)=ELX(2)
C
      DO 50 NI=1,NGP
      XI=GAUSPT(NI,NGP)
      CALL SHAPE1D(H,IEL,NPE,XI)
      CONST=GJ*GAUSWT(NI,NGP)
      X = 0.0
      DO 20 J=1,NPE
20    X = X + SF(J)*EX(J)
C
C     Compute the polynomial variation of FX
C
      IF(NTYPE.EQ.2)THEN
         FX=FX0+(FX1+FX2*X)*X
      ELSE
         FX=(FX0+FX1*X)*X
      ENDIF
C
C     Element force vector for the consistent interpolation beam element
C
25    II=1
      DO 40 I=1,NPE
      ELF(II)=ELF(II)+FX*SF(I)*CONST
40    II=NDF*I+1
50    CONTINUE
C
C     Rearrange the element coefficients
C
      F3(NE)=ELF(3)
      ELF(1)=ELF(1)+0.5*F3(NE)
      ELF(2)=-0.125*F3(NE)*H
      ELF(3)=ELF(5)+0.5*F3(NE)
      ELF(4)= 0.125*F3(NE)*H
      RETURN
      END


      SUBROUTINE TIMSTRES(GA,ELU,XI,W,DW,S,DS,NE,F3,H,MXELM)
C
C
C     Called in POSTPROC to compute solution and its global derivatives
C       at nine points (including the nodes) of the Timoshenko element
C
C        XC........ Global (i.e., problem) coordinate
C        XI ....... Local (i.e., element) coordinate
C        SFL, SFQ.. Lagrange linear and quadratic shape functions
C        DSFL,DSFQ: First derivative of SF w.r.t. global coordinate
C        ELU....... Column vector of generalized displacements
C        W, DW..... Transverse deflection and its derivative
C        S, DS..... Rotation and its derivative
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/IO/IN,IT
      DIMENSION ELU(9),SFL(2),SFQ(3),DSFL(2),DSFQ(3),F3(MXELM)
C
      GJ =  H*0.5
C
C     Interpolation functions for the Lagrange LINEAR element
C
```

```
      SFL(1) = 0.5*(1.0-XI)
      SFL(2) = 0.5*(1.0+XI)
      DSFL(1) = -0.5/GJ
      DSFL(2) = 0.5/GJ
C
C     Interpolation functions for the Lagrange QUADRATIC element
C
      SFQ(1) = -0.5*XI*(1.0-XI)
      SFQ(2) = 1.0-XI*XI
      SFQ(3) = 0.5*XI*(1.0+XI)
      DSFQ(1) = -0.5*(1.0-2.0*XI)/GJ
      DSFQ(2) = -2.0*XI/GJ
      DSFQ(3) = 0.5*(1.0+2.0*XI)/GJ
C
      W3=(3.0*H*F3(NE)/GA + 8.0*(ELU(1)+ELU(3))
     *                    + 2.0*(ELU(4)-ELU(2))*H)/16.0
      W =  SFQ(1)*ELU(1) + SFQ(2)*W3 + SFQ(3)*ELU(3)
      DW= DSFQ(1)*ELU(1) +DSFQ(2)*W3 +DSFQ(3)*ELU(3)
      S =  SFL(1)*ELU(2) + SFL(2)*ELU(4)
      DS= DSFL(1)*ELU(2) +DSFL(2)*ELU(4)
C
      RETURN
      END


      SUBROUTINE TRANSFRM(MXELM,N,NTYPE,PR,SE,SL,SA,SI,CS,SN,CNT,SNT,
     *                    HF,VF,PF,XB)
C
C
C     Called in both MAIN and REACTION to compute stiffness matrix and
C      force vector for the truss (NDF=2) and frame (NDF=3) elements
C
C     SE......Young's modulus
C     SL......Element length
C     SA......Cross-sectional area
C     SI......Moment of inertia
C     CS......Cosine of the angle of orientation
C     SN......Sine of the angle of orientation
C     HF......Distributed force along the length of the element
C     VF......Distributed force transverse to the element
C     PF......Point force at point other than nodes
C     XB......Distance along the length from node 1 of the element
C             of the location of the point force, PF
C     CNT,SNT:Direction cosines of the point force's line of application
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION  PR(MXELM),SE(MXELM),SL(MXELM),SA(MXELM),SI(MXELM)
      DIMENSION  CS(MXELM),SN(MXELM),CNT(MXELM),SNT(MXELM)
      DIMENSION  HF(MXELM),VF(MXELM),PF(MXELM),XB(MXELM)
      DIMENSION  TRM(6,6),TMPK(6,6)
      COMMON/STF1/ELK(9,9),ELM(9,9),ELF(9),ELX(4),ELU(9),ELV(9),ELA(9)
C
      CN1=CS(N)
      SN1=SN(N)
      CN2=CN1*CN1
      SN2=SN1*SN1
      CSN=CN1*SN1
C
C     Element coefficients
C
      IF(NTYPE.EQ.0) THEN
C
C     The plane TRUSS element
C
         NN=4
         C1=SA(N)*SE(N)/SL(N)
         ELK(1,1) = C1*CN2
         ELK(2,1) = C1*CSN
         ELK(2,2) = C1*SN2
         ELK(3,1) = -ELK(1,1)
         ELK(3,2) = -ELK(2,1)
         ELK(3,3) =  ELK(1,1)
         ELK(4,1) = -ELK(2,1)
         ELK(4,2) = -ELK(2,2)
         ELK(4,3) = -ELK(3,2)
```

```
         ELK(4,4) =  ELK(2,2)
C
         DO 10 I=1,NN
         DO 10 J=I,NN
   10    ELK(I,J) = ELK(J,I)
C
C     Contribution of the point force to nodal forces
C
         XI=XB(N)/SL(N)
         SFL1 = 1.0-XI
         SFL2 = XI
C
         F1=0.5*HF(N)*SL(N)
         F3=0.5*HF(N)*SL(N)
         ELF(1) = F1*CN1
         ELF(2) = F1*SN1
         ELF(3) = F3*CN1
         ELF(4) = F3*SN1
      ELSE
         NN=6
         IF(NTYPE.EQ.1)THEN
C
C     The EULER-BERNOULLI FRAME element
C
            AMU=0.5*SA(N)*SL(N)*SL(N)/SI(N)
            C1=2.0*SE(N)*SI(N)/(SL(N)**3)
            C2=6.0*SE(N)*SI(N)/(SL(N)*SL(N))
            C3=C1*(AMU*CN2+6.0*SN2)
            C4=C1*(AMU-6.0)*CSN
            C5=C1*(AMU*SN2+6.0*CN2)
            C6=4.0*SE(N)*SI(N)/SL(N)
C
            ELK(1,1) = C3
            ELK(2,1) = C4
            ELK(2,2) = C5
            ELK(3,1) = C2*SN1
            ELK(3,2) =-C2*CN1
            ELK(3,3) = C6
            ELK(4,1) =-C3
            ELK(4,2) =-C4
            ELK(4,3) =-C2*SN1
            ELK(4,4) = C3
            ELK(5,1) =-C4
            ELK(5,2) =-C5
            ELK(5,3) = C2*CN1
            ELK(5,4) = C4
            ELK(5,5) = C5
            ELK(6,1) = C2*SN1
            ELK(6,2) =-C2*CN1
            ELK(6,3) = 0.5*C6
            ELK(6,4) =-C2*SN1
            ELK(6,5) = C2*CN1
            ELK(6,6) = C6
C
            DO 20 I=1,NN
            DO 20 J=I,NN
   20       ELK(I,J) = ELK(J,I)
C
C     Contribution of the point force to nodal generalized forces
C
            XI=XB(N)/SL(N)
            TF=PF(N)*SNT(N)
            AF=PF(N)*CNT(N)
            SFL1 = 1.0-XI
            SFL2 = XI
            SFH1 = 1.0 - 3.0*XI*XI + 2.0*(XI**3)
            SFH2 = -XI*(1.0+XI*XI-2.0*XI)*SL(N)
            SFH3 = 3.0*XI*XI - 2.0*(XI**3)
            SFH4 = -XI*(XI*XI - XI)*SL(N)
C
            F1=0.5*HF(N)*SL(N)          + SFL1*AF
            F2=0.5*VF(N)*SL(N)          + SFH1*TF
            F3=-VF(N)*SL(N)*SL(N)/12.0  + SFH2*TF
            F4=0.5*HF(N)*SL(N)          + SFL2*AF
            F5=0.5*VF(N)*SL(N)          + SFH3*TF
            F6=VF(N)*SL(N)*SL(N)/12.0   + SFH4*TF
```

```
            ELF(1) = F1*CN1-F2*SN1
            ELF(2) = F1*SN1+F2*CN1
            ELF(3) = F3
            ELF(4) = F4*CN1-F5*SN1
            ELF(5) = F4*SN1+F5*CN1
            ELF(6) = F6
         ELSE
C
C     The TIMOSHENKO FRAME element (shear coefficient=5/6)
C
            SG=5.0*SE(N)/(1.0+PR(N))/12.0
            C1=SA(N)*SE(N)/SL(N)
            C2=SG*SA(N)/SL(N)
            C3=0.5*SG*SA(N)
            C4=0.25*SG*SA(N)*SL(N)
            C5=SE(N)*SI(N)/SL(N)
            ELK(1,1)=C1
            ELK(2,1)=0.0
            ELK(2,2)=C2
            ELK(3,1)=0.0
            ELK(3,2)=-C3
            ELK(3,3)=C4+C5
            ELK(4,1)=-C1
            ELK(4,2)=0.0
            ELK(4,3)=0.0
            ELK(4,4)=C1
            ELK(5,1)=0.0
            ELK(5,2)=-C2
            ELK(5,3)=C3
            ELK(5,4)=0.0
            ELK(5,5)=C2
            ELK(6,1)=0.0
            ELK(6,2)=-C3
            ELK(6,3)=C4-C5
            ELK(6,4)=0.0
            ELK(6,5)=C3
            ELK(6,6)=C4+C5
C
            DO 25 I=1,NN
            DO 25 J=1,NN
   25       TRM(J,I)=0.0
C
            TRM(1,1)=CN1
            TRM(1,2)=SN1
            TRM(2,1)=-SN1
            TRM(2,2)=CN1
            TRM(3,3)=1.0
            TRM(4,4)=CN1
            TRM(4,5)=SN1
            TRM(5,4)=-SN1
            TRM(5,5)=CN1
            TRM(6,6)=1.0
C
            DO 30 I=1,NN
            DO 30 J=I,NN
   30       ELK(I,J) = ELK(J,I)
C
            DO 40 I=1,NN
            DO 40 J=1,NN
            TMPK(I,J)=0.0
            DO 40 K=1,NN
   40       TMPK(I,J)=TMPK(I,J)+TRM(K,I)*ELK(K,J)
C
            DO 50 I=1,NN
            DO 50 J=1,NN
            ELK(I,J)=0.0
            DO 50 K=1,NN
   50       ELK(I,J)=ELK(I,J)+TMPK(I,K)*TRM(K,J)
C
C     Contribution of the point force to nodal generalized forces
C
            XI=XB(N)/SL(N)
            TF=PF(N)*SNT(N)
            AF=PF(N)*CNT(N)
            SFL1 = 1.0-XI
            SFL2 = XI
```

```
         SFQ1 = (1.0-XI)*(1.0-2.0*XI)
         SFQ2 = -XI*(1.0-2.0*XI)
         SFQ3 = 4.0*XI*(1.0-XI)
C
         F1=0.5*HF(N)*SL(N)          + SFL1*AF
         F2=0.5*VF(N)*SL(N)          + (SFQ1+0.5*SFQ3)*TF
         F3=-VF(N)*SL(N)*SL(N)/12.0 - 0.125*SFQ3*SL(N)*TF
         F4=0.5*HF(N)*SL(N)          + SFL2*AF
         F5=0.5*VF(N)*SL(N)          + (SFQ2+0.5*SFQ3)*TF
         F6=VF(N)*SL(N)*SL(N)/12.0  + 0.125*SFQ3*SL(N)*TF
         ELF(1) = F1*CN1-F2*SN1
         ELF(2) = F1*SN1+F2*CN1
         ELF(3) = F3
         ELF(4) = F4*CN1-F5*SN1
         ELF(5) = F4*SN1+F5*CN1
         ELF(6) = F6
      ENDIF
   ENDIF
   RETURN
   END
```