```
C     Program name: FEM2D         Length (including blanks):3000 lines
C
C    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C    *                        Program  FEM2D                          *
C    *          (A FINITE ELEMENT ANALYSIS COMPUTER PROGRAM)          *
C    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C    . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C    .                                                                .
C    .     This  is  a  finite  element  computer  program  for  the .
C    . analysis of two-dimensional problems governed by second-order .
C    . partial  differential  equations  arising in:  heat transfer, .
C    . electrical engineering, fluid dynamics, and solid mechanics.  .
C    .                                                                .
C    .     The  program  uses  linear and quadratic,  triangular and .
C    . rectangular, elements with isoparametric formulations. Meshes .
C    . of only one type of element are allowed for a problem  (i.e., .
C    . two different types of elements cannot be used in a problem). .
C    .                                                                .
C    .     Many  field  problems  of engineering and applied science .
C    . can be analyzed using this program.  In  particular,  FEM2DV2 .
C    . can be used in the finite element analysis of problems in the .
C    . following fields:                                              .
C    .                                                                .
C    .     1.  Heat conduction and convection                         .
C    .     2.  Flows of viscous incompressible fluids (by penalty     .
C    .         function formulation)                                  .
C    .     3.  Plane elasticity problems                              .
C    .     4.  Plate bending problems using rectangular elements      .
C    .         based on the classical and first-order (or Mindlin)    .
C    .         plate theory.                                          .
C    .                                                                .
C    .     The main objective of this  program  is to illustrate how .
C    . finite element formulations developed in  Chapters  8 thru 12 .
C    . can be implemented on a computer  and used in the analysis of .
C    . engineering problems.  Modeling of large and complex problems .
C    . was not an objective of the program.  The program or parts of .
C    . it can be modified to analyze field problems not discussed in .
C    . the book.                                                      .
C    . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C          DESCRIPTION OF SOME KEY VARIABLES USED IN THE PROGRAM
C    (See Table 13.1 of the BOOK for a description of other variables)
C
C      [CMAT]    Matrix of stiffnesses in elasticity and plate bending
C                problems (computed  in  the  program from engineering
C                constants, E1, E2, G12, ANU12, etc. and thickness)
C
C      {ELA}     Vector of elemental nodal accelerations
C      {ELF}     Vector of element nodal source (or force) vector
C      [ELK]     Element coefficient (or stiffness) matrix
C      {ELU}     Vector of element nodal values of primary variables
C      {ELV}     Vector of elemental nodal velocities
C      {ELXY}    Vector of elemental global coordinates:
C                ELXY(I,1)=x-coordinate;  ELXY(I,2)=y-coordinate
C      {GLA}     Vector of global nodal accelerations
C      {GLF}     Vector of global nodal source (or force) vector
C      [GLK]     Global coefficient (or stiffness) matrix
C      {GLU}     Vector of global nodal values of primary variables
C      {GLV}     Vector of global nodal velocities
C
C      NDF       Number of degrees of freedom per node:
C                NDF=1, For SINGLE VARIABLE problems
C                NDF=2, For ELASTICITY and VISCOUS FLUID FLOW
C                NDF=3, For PLATE BENDING when FSDT or CST(N)
C                       elements are used
C                NDF=4, For PLATE BENDING when CST(C) element
C                       is used
C
C      NEQ       Total number of equations in the problem (=NNM*NDF)
C      NHBW      Half band width of the global coefficient matrix, GLK
C      NN        Total number of degrees of freedom per element
C      -----------------------------------------------------------------
C
C          DESCRIPTION OF PARAMETERS USED TO DIMENSION THE ARRAYS
C
C      MAXCNV... Maximum number of elements with convection B.C.
```

```
C      MAXELM... Maximum number of elements allowed in the program
C      MAXNOD... Maximum number of nodes allowed in the program
C      MAXNX.... Maximum number of allowed subdivisions DX(I) along x
C      MAXNY.... Maximum number of allowed subdivisions DY(I) along y
C      MAXSPV... Maximum number of specified primary variables
C      MAXSSV... Maximum number of specified secondary variables
C      NCMAX.... Actual column dimension of: [GLK],[GLM],{GLU},{GLV},
C                {GLA}, and {GLF}
C
C                The actual row dimension of the assembled coefficient
C                matrix  should be greater than or equal to  the total
C                number of algebraic equations in the FE model.
C
C      NRMAX.... Actual row dimension of: [GLK] and [GLM]
C
C                The actual column-dimension of  assembled coefficient
C                matrix  should be greater than or equal to  the  half
C                bandwidth for static analysis or the total  number of
C                equations for the dynamic analysis.
C
C      NOTE:     The  values  of NRMAX, NCMAX, MAXELM, MAXNOD, MAXCNV,
C                MAXSSV and MAXSPV in the 'PARAMETER' statement should
C                modified as required by the size of the problem.
C                When  an  eigenvalue problem is solved, the following
C                dimension  statement should be in  'AXLBX'  should be
C                modified (i.e., replace 500 with the value of NRMAX):
C
C                DIMENSION  V(7500,750),VT(750,750),W(750,750),IH(750)
C
C      ---------------------------------------------------------------
C                     SUBROUTINES USED IN THE PROGRAM
C
C       BOUNDARY,CONCTVTY,DATAECHO,EGNBNDRY,EGNSOLVR,ELKMFRCT,ELKMFTRI
C         EQNSOLVR,INVERSE,JACOBI,MESH2DG,MESH2DR,MATRXMLT,POSTPROC
C                    QUADRTRI,SHAPERCT,SHAPETRI,TEMPORAL
C      ---------------------------------------------------------------
C
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (NRMAX =750,NCMAX =750,MAXELM=500,MAXNOD=500,
     1           MAXSPV=500,MAXSSV=100,MAXCNV=200,MAXNX =25,MAXNY=25)
C
      DIMENSION ISPV(MAXSPV,2),VSPV(MAXSPV),ISSV(MAXSSV,2),VSSV(MAXSSV)
      DIMENSION IBN(MAXCNV),INOD(MAXCNV,3),BETA(MAXSPV),TINF(MAXSSV)
      DIMENSION GLF(NRMAX),TITLE(20),IBS(3),IBP(3),GLM(NRMAX,NRMAX)
      DIMENSION GLK(NRMAX,NCMAX),GLU(NRMAX),GLV(NRMAX),GLA(NRMAX)
      DIMENSION NOD(MAXELM,9),GLXY(MAXNOD,2),DX(MAXNX),DY(MAXNY)
      DIMENSION EGNVAL(NRMAX),EGNVEC(NRMAX,NRMAX),IBDY(MAXSPV)
C
      COMMON/STF/ELF(27),ELK(27,27),ELM(27,27),ELXY(9,2),ELU(27),
     1           ELV(27),ELA(27),A1,A2,A3,A4,A5
      COMMON/PST/A10,A1X,A1Y,A20,A2X,A2Y,A00,C0,CX,CY,F0,FX,FY,
     1           C44,C55,VISCSITY,PENALTY,CMAT(3,3)
      COMMON/PNT/IPDF,IPDR,NIPF,NIPR
      COMMON/IO/IN,ITT
      COMMON/WORKSP/RWKSP
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                           *
C     *            P R E P R O C E S S O R   U N I T              *
C     *                                                           *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      IN=5
      ITT=6
C     **************************************************************
      open (in,file = ' ')
      open (itt,file = ' ')
C     **************************************************************
      CALL DATAECHO(IN,ITT)
      ICONV=0
      INTIAL=0
      JVEC=1
      NSSV=0
      NFLAG=1
C
C     R E A D I N   T H E   I N P U T   D A T A   H E R E
```

```
C
      READ(IN,400) TITLE
C
C     Read problem and analysis type
C
      READ(IN,*)ITYPE,IGRAD,ITEM,NEIGN
      IF(ITEM.EQ.0) NEIGN=0
      IF(NEIGN.NE.0) THEN
         IF(ITYPE.LE.3 .AND. NEIGN.GT.1) THEN
            WRITE(ITT,991)
            STOP
         ELSE
            READ(IN,*) NVALU,NVCTR
         ENDIF
      ENDIF
C
C     Read finite element mesh information
C
      READ(IN,*) IELTYP,NPE,MESH,NPRNT
      IF(ITYPE.GE.3 .AND. IELTYP.EQ.0) THEN
         WRITE(ITT,990)
         STOP
      ENDIF
      IF(NPE.LE.4) THEN
         IEL=1
      ELSE
         IEL=2
      ENDIF
      IF(MESH.NE.1) THEN
         READ(IN,*) NEM,NNM
         IF(MESH.EQ.0)THEN
C
C     If mesh CANNOT be generated by the program, read the mesh data in
C     the next three statements
C
            DO 10 N=1,NEM
   10       READ(IN,*) (NOD(N,I),I=1,NPE)
            READ(IN,*) ((GLXY(I,J),J=1,2),I=1,NNM)
         ELSE
C
C     When mesh is to be generated by the program for more complicated
C     geometries, call MESH2DGeneral (which reads pertinent data there)
C
            CALL MESH2DG(NEM,NNM,NOD,MAXELM,MAXNOD,GLXY)
         ENDIF
      ELSE
C
C     When mesh is to be generated for rectangular domains, call program
C     MESH2DRectangular, which requires the following data:
C
        READ(IN,*) NX,NY
        READ(IN,*) X0,(DX(I),I=1,NX)
        READ(IN,*) Y0,(DY(I),I=1,NY)
        CALL MESH2DR (IEL,IELTYP,NX,NY,NPE,NNM,NEM,NOD,DX,DY,X0,Y0,
     *                 GLXY,MAXELM,MAXNOD,MAXNX,MAXNY)
      ENDIF
C
      IF(ITYPE.EQ.0) THEN
         NDF = 1
      ELSE
         IF(ITYPE.GE.3) THEN
            NDF = 3
         ELSE
            NDF = 2
         ENDIF
      ENDIF
      IF(ITYPE.EQ.5) NDF=4
C
      NEQ=NNM*NDF
      NN=NPE*NDF
      IF(NEIGN.EQ.0) THEN
C
C     Compute the half bandwidth of the global coefficient matrix
C
         NHBW=0
         DO 20 N=1,NEM
```

```
         DO 20 I=1,NPE
         DO 20 J=1,NPE
         NW=(IABS(NOD(N,I)-NOD(N,J))+1)*NDF
   20    IF (NHBW.LT.NW) NHBW=NW
      ELSE
         NHBW=NEQ
      ENDIF
C
C     Read specified primary and secondary degrees of freedom: node
C     number, local degree of freedom number, and specified value.
C
      READ(IN,*) NSPV
      IF(NSPV.NE.0) THEN
         READ(IN,*) ((ISPV(I,J),J=1,2),I=1,NSPV)
         IF(NEIGN.EQ.0) THEN
            READ(IN,*) (VSPV(I),I=1,NSPV)
         ENDIF
      ENDIF
      IF(NEIGN.EQ.0) THEN
         READ(IN,*) NSSV
         IF(NSSV.NE.0) THEN
            READ(IN,*) ((ISSV(I,J),J=1,2),I=1,NSSV)
            READ(IN,*) (VSSV(I),I=1,NSSV)
         ENDIF
      ENDIF
      WRITE(ITT,400) TITLE
      WRITE(ITT,910)
      WRITE(ITT,890)
      WRITE(ITT,910)
      IF(ITYPE.EQ.0) THEN
C
C     Heat transfer and like problems:___________________________________
C     Read the coefficients of the differential equation modeled
C     A11 = A10 + A1X*X + A1Y*Y; A22 = A20 + A2X*X + A2Y*Y; A00=CONST.
C
         WRITE(ITT,410)
         READ(IN,*)A10,A1X,A1Y
         READ(IN,*)A20,A2X,A2Y
         READ(IN,*)A00
         WRITE(ITT,420) A10,A1X,A1Y,A20,A2X,A2Y,A00
         READ(IN,*)ICONV
         IF(ICONV.NE.0)THEN
            READ(IN,*)NBE
            READ(IN,*)(IBN(I),(INOD(I,J),J=1,2),BETA(I),TINF(I),I=1,NBE)
            WRITE(ITT,440) NBE
            DO 30 I=1,NBE
   30       WRITE(ITT,860) IBN(I),(INOD(I,J),J=1,2),BETA(I),TINF(I)
         ENDIF
      ELSE
         IF(ITYPE.EQ.1) THEN
C
C     Viscous incompressible flows:_____________________________________
C
            WRITE(ITT,450)
            READ(IN,*)VISCSITY,PENALTY
            WRITE(ITT,460) VISCSITY,PENALTY
         ELSE
            IF(ITYPE.EQ.2) THEN
C
C     Plane elasticity problems:________________________________________
C
               READ(IN,*) LNSTRS
               WRITE(ITT,470)
               READ(IN,*) E1,E2,ANU12,G12,THKNS
               WRITE(ITT,520) THKNS,E1,E2,ANU12,G12
C
C     Compute the material coefficient matrix, CMAT(I,J), I,J=1,2,3.
C
               ANU21=ANU12*E2/E1
               DENOM=1.0-ANU12*ANU21
               CMAT(3,3)=G12*THKNS
               IF(LNSTRS.EQ.0) THEN
C
C     Plane strain  (ANU23 = ANU12)
C
                  WRITE(ITT,490)
```

```
                 S0=(1.0-ANU12-2.0*ANU12*ANU21)
                 CMAT(1,1)=THKNS*E1*(1.0-ANU12)/S0
                 CMAT(1,2)=THKNS*E1*ANU21/S0
                 CMAT(2,2)=THKNS*E2*DENOM/S0/(1.0+ANU12)
              ELSE
C
C     Plane stress
C
                 WRITE(ITT,510)
                 CMAT(1,1)=THKNS*E1/DENOM
                 CMAT(1,2)=ANU21*CMAT(1,1)
                 CMAT(2,2)=E2*CMAT(1,1)/E1
              ENDIF
           ELSE
C
C     Plate bending problems:___________________________________________
C
              WRITE(ITT,500)
              IF(ITYPE.EQ.3) THEN
                 WRITE(ITT,505)
              ELSE
                 WRITE(ITT,506)
              ENDIF
              READ(IN,*) E1,E2,ANU12,G12,G13,G23,THKNS
              WRITE(ITT,520) THKNS,E1,E2,ANU12,G12
              WRITE(ITT,530) G13,G23
              ANU21=ANU12*E2/E1
              DENOM=1.0-ANU12*ANU21
              CMAT(1,1)=(THKNS**3)*E1/DENOM/12.0D0
              CMAT(1,2)=ANU21*CMAT(1,1)
              CMAT(2,2)=E2*CMAT(1,1)/E1
              CMAT(3,3)=G12*(THKNS**3)/12.0D0
              SCF=5.0D0/6.0D0
              C44=SCF*G23*THKNS
              C55=SCF*G13*THKNS
           ENDIF
           CMAT(1,3)=0.0
           CMAT(2,3)=0.0
           CMAT(2,1)=CMAT(1,2)
           CMAT(3,1)=CMAT(1,3)
           CMAT(3,2)=CMAT(2,3)
        ENDIF
      ENDIF
C
      IF(NEIGN.EQ.0) THEN
         READ(IN,*)F0,FX,FY
         WRITE(ITT,430) F0,FX,FY
      ENDIF
C
      IF(ITEM.NE.0) THEN
         READ(IN,*) C0,CX,CY
         IF(ITYPE.GT.1) THEN
            IF(ITYPE.EQ.2)THEN
               C0=THKNS*C0
               CX=THKNS*CX
               CY=THKNS*CY
            ELSE
               IF(NEIGN.LE.1) THEN
                  C0=THKNS*C0
                  CX=(THKNS**2)*C0/12.0D0
                  CY=CX
               ENDIF
            ENDIF
         ENDIF
C
         IF(NEIGN.NE.0) THEN
            WRITE(ITT,810)
            WRITE(ITT,540) C0,CX,CY
         ELSE
            WRITE(ITT,820)
            WRITE(ITT,540) C0,CX,CY
C
C     Read the necessary data for time-dependent problems
C
            READ(IN,*) NTIME,NSTP,INTVL,INTIAL
            IF(INTVL.LE.0)INTVL=1
```

```
            READ(IN,*) DT,ALFA,GAMA,EPSLN
            A1=ALFA*DT
            A2=(1.0-ALFA)*DT
            WRITE(ITT,550) DT,ALFA,GAMA,NTIME,NSTP,INTVL
            IF(ITEM.EQ.1) THEN
               IF(NSSV.NE.0) THEN
                  DO 40 I=1,NSSV
   40             VSSV(I)=VSSV(I)*DT
               ENDIF
               IF(INTIAL.NE.0) THEN
                  READ(IN,*) (GLU(I),I=1,NEQ)
               ELSE
                  DO 50 I=1,NEQ
   50             GLU(I)=0.0
               ENDIF
            ELSE
               DT2=DT*DT
               A3=2.0/GAMA/DT2
               A4=A3*DT
               A5=1.0/GAMA-1.0
               IF(INTIAL.NE.0) THEN
                  READ(IN,*) (GLU(I),I=1,NEQ)
                  READ(IN,*) (GLV(I),I=1,NEQ)
                  DO 60 I=1,NEQ
   60             GLA(I)=0.0
               ELSE
                  DO 70 I=1,NEQ
                  GLU(I)=0.0
                  GLV(I)=0.0
   70             GLA(I)=0.0
               ENDIF
            ENDIF
         ENDIF
      ELSE
         WRITE(ITT,830)
      ENDIF
C
C     *****     E N D   O F   T H E   D A T A   I N P U T      *****
C
      IF(IELTYP.EQ.0) THEN
         WRITE(ITT,790)
      ELSE
         WRITE(ITT,800)
      ENDIF
C
      WRITE(ITT,560) IELTYP,NPE,NDF,NEM,NNM,NEQ,NHBW
      IF(MESH.EQ.1) WRITE(ITT,570) NX,NY
      WRITE(ITT,710)NSPV
      IF(NSSV.NE.0) THEN
         WRITE(ITT,715)NSSV
         WRITE(ITT,720)
         DO 80 IB=1,NSSV
   80    WRITE(ITT,960)(ISSV(IB,JB),JB=1,2),VSSV(IB)
      ENDIF
C
      IF(NPRNT.EQ.1) THEN
         WRITE(ITT,700)
         DO 100 I=1,NEM
  100    WRITE(ITT,900) I,(NOD(I,J),J=1,NPE)
      ENDIF
C
      WRITE(ITT,910)
      WRITE(ITT,580)
      WRITE(ITT,910)
      DO 150 IM=1,NNM
      DO 110 K=1,NDF
      IBP(K)=0
  110 IBS(K)=0
      IF(NSPV.NE.0) THEN
         DO 120 JP=1,NSPV
         NODE=ISPV(JP,1)
         NDOF=ISPV(JP,2)
         IF(NODE.EQ.IM) THEN
            IBP(NDOF)=NDOF
         ENDIF
  120    CONTINUE
```

```
      ENDIF
C
      IF(NSSV.NE.0) THEN
         DO 140 JS=1,NSSV
         NODE=ISSV(JS,1)
         NDOF=ISSV(JS,2)
         IF(NODE.EQ.IM) THEN
            IBS(NDOF)=NDOF
         ENDIF
  140    CONTINUE
      ENDIF
C
      IF(NDF.EQ.1) THEN
         WRITE(ITT,870)IM,(GLXY(IM,J),J=1,2),(IBP(K),K=1,NDF),
     *                    (IBS(K),K=1,NDF)
      ELSE
         IF(NDF.EQ.2) THEN
            WRITE(ITT,920)IM,(GLXY(IM,J),J=1,2),(IBP(K),K=1,NDF),
     *                       (IBS(K),K=1,NDF)
         ELSE
            IF(NDF.EQ.3) THEN
               WRITE(ITT,880)IM,(GLXY(IM,J),J=1,2),(IBP(K),K=1,NDF),
     *                          (IBS(K),K=1,NDF)
            ELSE
               WRITE(ITT,885)IM,(GLXY(IM,J),J=1,2),(IBP(K),K=1,NDF),
     *                          (IBS(K),K=1,NDF)
            ENDIF
         ENDIF
      ENDIF
  150 CONTINUE
      WRITE(ITT,910)
C
C     Define the polynomial degree and number of integration points
C     (based on the assumed variation of the coefficients AX, BX, etc.)
C
      IPDR = IEL
      NIPR = IPDR+IEL-1
      IF(IELTYP.EQ.0) THEN
         IF(ITYPE.EQ.0) THEN
            IPDF = 2*IEL+1
            NIPF = IPDF+IEL
         ELSE
            IF(ITEM.NE.0) THEN
               IPDF = 2*IEL+1
               NIPF = IPDF+IEL
            ELSE
               IPDF = IEL+1
               NIPF = IPDF+1
            ENDIF
         ENDIF
         ISTR = 1
         NSTR = 1
         WRITE(ITT,480) IPDF,NIPF,IPDR,NIPR,ISTR,NSTR
      ELSE
         IF(ITYPE.GE.4) THEN
            IPDF = 4
            ISTR = 2
         ELSE
            IPDF = IEL+1
            ISTR = IEL
         ENDIF
         WRITE(ITT,485) IPDF,IPDR,ISTR
      ENDIF
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                             *
C     *                P R O C E S S O R   U N I T                  *
C     *                                                             *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      IF(ITEM.NE.0) THEN
         TIME=0.0
      ENDIF
C
C     Counter on number of TIME steps begins here
C
```

```
      NT = 0
      NCOUNT=0
  170 NCOUNT=NCOUNT+1
      IF(ITEM.NE.0 .AND. NEIGN.EQ.0) THEN
         IF(NCOUNT.GE.NSTP) THEN
            F0=0.0
            FX=0.0
            FY=0.0
         ENDIF
      ENDIF
C
C     Initialize the global coefficient matrices and vectors
C
      DO 180 I=1,NEQ
      GLF(I)=0.0
      DO 180 J=1,NHBW
      IF(NEIGN.NE.0) GLM(I,J)=0.0
  180 GLK(I,J)=0.0
C
C     Do-loop on the number of ELEMENTS to compute element matrices
C     and their assembly begins here
C
      DO 250 N=1,NEM
      DO 200 I=1,NPE
      NI=NOD(N,I)
      ELXY(I,1)=GLXY(NI,1)
      ELXY(I,2)=GLXY(NI,2)
      IF(NEIGN.EQ.0) THEN
         IF(ITEM.NE.0) THEN
            LI=(NI-1)*NDF
            L = (I-1)*NDF
            DO 190 J=1,NDF
            LI=LI+1
            L=L+1
            ELU(L)=GLU(LI)
            IF(ITEM.EQ.2) THEN
               ELV(L)=GLV(LI)
               ELA(L)=GLA(LI)
            ENDIF
  190       CONTINUE
         ENDIF
      ENDIF
  200 CONTINUE
C
C     Call subroutine ELKMFTRI (for Triangular elements) or ELKMFRCT (for
C     Rectangular elements) to compute the ELement [K], [M] and {F}.
C
      IF(IELTYP.EQ.0) THEN
         CALL ELKMFTRI (NEIGN,NPE,NN,ITYPE,ITEM)
      ELSE
         CALL ELKMFRCT (NEIGN,NPE,NN,ITYPE,ITEM)
      ENDIF
C
      IF(ICONV.NE.0) THEN
C
C     Add the convective terms for CONVECTION type boundary conditions
C     (exact for straight sided elements; otherwise approximate values)
C
         DO 210 M = 1,NBE
         IF(IBN(M).EQ.N) THEN
            M1  = INOD(M,1)
            M2  = INOD(M,2)
            NM1 = NOD(N,M1)
            NM2 = NOD(N,M2)
            DL  = DSQRT((GLXY(NM2,1)-GLXY(NM1,1))**2
     *                +(GLXY(NM2,2)-GLXY(NM1,2))**2)
            BL  = BETA(M)*DL
            TF  = TINF(M)*BL
            IF(IEL.EQ.1)THEN
               ELK(M1,M1)=ELK(M1,M1)+BL/3.0
               ELK(M1,M2)=ELK(M1,M2)+BL/6.0
               ELK(M2,M1)=ELK(M2,M1)+BL/6.0
               ELK(M2,M2)=ELK(M2,M2)+BL/3.0
               ELF(M1)=ELF(M1)+0.5*TF
               ELF(M2)=ELF(M2)+0.5*TF
            ELSE
```

```
            IF(NPE.GE.8) THEN
               NPEL=4
            ELSE
               NPEL=3
            ENDIF
            M3=M1+NPEL
            ELK(M1,M1)=ELK(M1,M1)+4.0*BL/30.0
            ELK(M1,M3)=ELK(M1,M3)+2.0*BL/30.0
            ELK(M1,M2)=ELK(M1,M2)-BL/30.0
            ELK(M3,M1)=ELK(M3,M1)+2.0*BL/30.0
            ELK(M3,M3)=ELK(M3,M3)+16.0*BL/30.0
            ELK(M2,M3)=ELK(M2,M3)+2.0*BL/30.0
            ELK(M2,M1)=ELK(M2,M1)-BL/30.0
            ELK(M3,M2)=ELK(M3,M2)+2.0*BL/30.0
            ELK(M2,M2)=ELK(M2,M2)+4.0*BL/30.0
            ELF(M1)=ELF(M1)+TF/6.0
            ELF(M3)=ELF(M3)+4.0*TF/6.0
            ELF(M2)=ELF(M2)+TF/6.0
         ENDIF
      ENDIF
  210 CONTINUE
    ENDIF
C
    IF(NCOUNT.EQ.1) THEN
       IF(NPRNT.EQ.1 .OR. NPRNT.EQ.3) THEN
          IF(N.EQ.1) THEN
C
C     Print element matrices and vectors (only when NPRNT=1 or NPRNT=3)
C
            WRITE(ITT,610)
            DO 220 I=1,NN
  220       WRITE(ITT,930) (ELK(I,J),J=1,NN)
            IF(NEIGN.EQ.0) THEN
               WRITE(ITT,630)
               WRITE(ITT,930) (ELF(I),I=1,NN)
            ELSE
               WRITE(ITT,620)
               DO 230 I=1,NN
  230          WRITE(ITT,930) (ELM(I,J),J=1,NN)
            ENDIF
         ENDIF
      ENDIF
    ENDIF
C
    IF(NEIGN.EQ.0) THEN
       IF(ITEM.NE.0) THEN
C
C     Compute the element coefficient matrices [K-hat] and {F-hat}
C     (i.e., after time approximation) in the transient analysis:_______
C
         CALL TEMPORAL(NCOUNT,INTIAL,ITEM,NN)
       ENDIF
    ENDIF
C
C     ASSEMBLE element matrices to obtain global matrices:______________
C
    DO 240 I=1,NPE
       NR=(NOD(N,I)-1)*NDF
       DO 240 II=1,NDF
          NR=NR+1
          L=(I-1)*NDF+II
          IF(NEIGN.EQ.0) THEN
             GLF(NR)=GLF(NR)+ELF(L)
          ENDIF
          DO 240 J=1,NPE
             IF(NEIGN.EQ.0) THEN
                NCL=(NOD(N,J)-1)*NDF
             ELSE
                NC=(NOD(N,J)-1)*NDF
             ENDIF
             DO 240 JJ=1,NDF
                M=(J-1)*NDF+JJ
                IF(NEIGN.EQ.0) THEN
                   NC=NCL+JJ+1-NR
                   IF(NC.GT.0) THEN
                      GLK(NR,NC)=GLK(NR,NC)+ELK(L,M)
```

```
                  ENDIF
               ELSE
                  NC=NC+1
                  GLK(NR,NC)=GLK(NR,NC)+ELK(L,M)
                  GLM(NR,NC)=GLM(NR,NC)+ELM(L,M)
               ENDIF
  240    CONTINUE
  250 CONTINUE
C
C     Print global matrices when NPRNT > 2
C
      IF(NCOUNT.LE.1) THEN
         IF(NPRNT.GE.2) THEN
            WRITE(ITT,640)
            DO 260 I=1,NEQ
  260       WRITE(ITT,930) (GLK(I,J),J=1,NHBW)
            IF(NEIGN.EQ.0) THEN
               WRITE(ITT,650)
               WRITE(ITT,930) (GLF(I),I=1,NEQ)
            ELSE
               WRITE(ITT,655)
               DO 265 I=1,NEQ
  265          WRITE(ITT,930) (GLM(I,J),J=1,NEQ)
            ENDIF
         ENDIF
      ENDIF
C
C     Impose BOUNDARY CONDITIONS on primary and secondary variables
C
      IF(NEIGN.NE.0) THEN
         CALL EGNBNDRY(GLK,GLM,IBDY,ISPV,MAXSPV,NDF,NEQ,NEQR,NSPV,NRMAX)
C
C     Call subroutine EGNSOLVR to solve for eigenvalues and eigenvectors
C     and print them as specified
C
         CALL EGNSOLVR(NEQR,GLK,GLM,EGNVAL,EGNVEC,JVEC,NROT,NRMAX)
         WRITE(ITT,660)
         WRITE(ITT,665) NROT
         IF(NVALU.GT.NEQR)NVALU=NEQR
            DO 270 I=1,NVALU
            IF(ITEM.GE.2 .AND. NEIGN.EQ.1) THEN
               VALUE = DSQRT(EGNVAL(I))
               WRITE(ITT,840)I,EGNVAL(I),VALUE
            ELSE
               WRITE(ITT,845)I,EGNVAL(I)
            ENDIF
            IF(NVCTR.NE.0) THEN
               WRITE(ITT,850)
               WRITE(ITT,930)(EGNVEC(J,I),J=1,NEQR)
            ENDIF
  270       CONTINUE
            STOP
         ELSE
            CALL BOUNDARY(ISPV,ISSV,MAXSPV,MAXSSV,NDF,NCMAX,NRMAX,NEQ,
     *                 NHBW,NSPV,NSSV,GLK,GLF,VSPV,VSSV,NCOUNT,INTIAL)
            IF(NCOUNT.LE.1) THEN
               IF(NPRNT.GE.2) THEN
                  WRITE(ITT,650)
                  WRITE(ITT,930) (GLF(I),I=1,NEQ)
               ENDIF
            ENDIF
C
C     Call subroutine  EQNSOLVR  to solve the system of algebraic equations
C     The solution is returned in the array GLF
C
            IRES=0
            CALL EQNSOLVR(NRMAX,NCMAX,NEQ,NHBW,GLK,GLF,IRES)
C
            IF(ITEM.NE.0) THEN
C
C     For nonzero initial conditions, GLF in the very first solution
C     is the acceleration, {A}=[MINV]({F}-[K]{U})
C
               IF(NCOUNT.EQ.1 .AND. INTIAL.NE.0) THEN
                  IF(ITEM.EQ.2) THEN
                     DO 280 I=1,NEQ
```

```
  280                 GLA(I)=GLF(I)
                      WRITE(ITT,600) TIME
                      WRITE(ITT,930) (GLA(I),I=1,NEQ)
                      GOTO 170
                   ELSE
                      NT = NT + 1
                      TIME=TIME+DT
                   ENDIF
                ELSE
                   NT = NT + 1
                   TIME=TIME+DT
                ENDIF
             ENDIF
C
C     Compute the difference between solutions at two consecutive times,
C     and calculate new velocities and accelerations
C
             DIFF=0.0
             SOLN=0.0
             DO 290 I=1,NEQ
             IF(ITEM.NE.0) THEN
                SOLN=SOLN+GLF(I)*GLF(I)
                DIFF=DIFF+(GLF(I)-GLU(I))*(GLF(I)-GLU(I))
             ENDIF
             IF(ITEM.EQ.2) THEN
                GLU(I)=A3*(GLF(I)-GLU(I))-A4*GLV(I)-A5*GLA(I)
                GLV(I)=GLV(I)+A1*GLU(I)+A2*GLA(I)
                GLA(I)=GLU(I)
             ENDIF
  290        GLU(I)=GLF(I)
             IF(ITEM.NE.0 .AND. NT.GT.1) THEN
                NFLAG=0
                PERCNT=DSQRT(DIFF/SOLN)
                IF(PERCNT.LE.EPSLN) THEN
                   WRITE(ITT,980)
                   STOP
                ELSE
                   INTGR=(NT/INTVL)*INTVL
                   IF(INTGR.EQ.NT) NFLAG=1
                ENDIF
             ENDIF
             IF(NFLAG.NE.0) THEN
C
C     Print the solution (i.e., nodal values of the primary variables)
C
                IF(ITEM.NE.0) THEN
                   WRITE(ITT,590) TIME,NT
                ENDIF
                WRITE(ITT,660)
                IF(NDF.LE.3) THEN
                   MDF=NDF
                ELSE
                   MDF=3
                   WRITE(ITT,666)
                   WRITE(ITT,930)(GLU(J),J=NDF,NEQ,NDF)
                ENDIF
                IF(ITYPE.EQ.0) THEN
                   WRITE(ITT,940)
                ELSE
                   WRITE(ITT,970)
                ENDIF
                   IF(NDF.EQ.1)WRITE(ITT,670)
                   IF(NDF.EQ.2)WRITE(ITT,680)
                   IF(NDF.GE.3)WRITE(ITT,690)
                   IF(ITYPE.EQ.0) THEN
                      WRITE(ITT,940)
                   ELSE
                      WRITE(ITT,970)
                   ENDIF
                   DO 300 I=1,NNM
                   II=NDF*(I-1)+1
                   JJ=II+MDF-1
  300              WRITE(ITT,950)I,(GLXY(I,J),J=1,2),(GLU(J),J=II,JJ)
                WRITE(ITT,970)
             ENDIF
             IF(IGRAD.NE.0) THEN
```

```
               IF(NFLAG.EQ.1) THEN
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     *                                                               *
C     *             P O S T P R O C E S S O R   U N I T               *
C     *                                                               *
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
               IF(ITYPE.LE.1) THEN
                  WRITE(ITT,970)
               ELSE
                  WRITE(ITT,940)
               ENDIF
               IF(ITYPE.LE.0) THEN
                  WRITE(ITT,730)
                  IF(IGRAD.EQ.1) THEN
                     WRITE(6,740)
                  ELSE
                     WRITE(6,750)
                  ENDIF
               ELSE
                  IF(ITYPE.EQ.1)WRITE(ITT,760)
                  IF(ITYPE.GE.2)WRITE(ITT,770)
                  IF(ITYPE.EQ.3)WRITE(ITT,780)
               ENDIF
               IF(ITYPE.LE.1) THEN
                  WRITE(ITT,970)
               ELSE
                  WRITE(ITT,940)
               ENDIF
C
C     Compute the GRADIENT of the solution for single-variable problems
C     or STRESSES for viscous flows, plane elasticity and plate bending
C
               DO 320 N=1,NEM
               DO 310 I=1,NPE
               NI=NOD(N,I)
               ELXY(I,1)=GLXY(NI,1)
               ELXY(I,2)=GLXY(NI,2)
               LI=(NI-1)*NDF
               L=(I-1)*NDF
               DO 310 J=1,NDF
               LI=LI+1
               L=L+1
               ELU(L)=GLU(LI)
  310          CONTINUE
  320          CALL POSTPROC(ELXY,ITYPE,IELTYP,IGRAD,NDF,NPE,THKNS,
     *                       ELU,ISTR,NSTR)
               IF(ITYPE.LE.1) THEN
                  WRITE(ITT,970)
               ELSE
                  WRITE(ITT,940)
               ENDIF
            ENDIF
         ENDIF
C
         IF(ITEM.NE.0) THEN
            IF(NT.GE.NTIME) THEN
               STOP
            ELSE
               GOTO 170
            ENDIF
         ENDIF
      ENDIF
      STOP
C
C                       F  O  R  M  A  T  S
C
  400 FORMAT(20A4)
  410 FORMAT (/,16X,'ANALYSIS  OF   A  POISSON/LAPLACE  EQUATION')
  420 FORMAT (/,5X,'COEFFICIENTS OF THE DIFFERENTIAL EQUATION:',//,
     *        8X,'Coefficient, A10 .......................=',E12.4,/,
     *        8X,'Coefficient, A1X .......................=',E12.4,/,
     *        8X,'Coefficient, A1Y .......................=',E12.4,/,
     *        8X,'Coefficient, A20 .......................=',E12.4,/,
     *        8X,'Coefficient, A2X .......................=',E12.4,/,
```

```
     *        8X,'Coefficient, A2Y .......................=',E12.4,/,
     *        8X,'Coefficient, A00 .......................=',E12.4,/)
  430 FORMAT (/,5X,'CONTINUOUS SOURCE COEFFICIENTS:',//,
     *        8X,'Coefficient, F0  .......................=',E12.4,/,
     *        8X,'Coefficient, FX  .......................=',E12.4,/,
     *        8X,'Coefficient, FY  .......................=',E12.4,/)
  440 FORMAT (/,5X,'CONVECTIVE HEAT TRANSFER DATA:',//,
     *        8X,'Number of elements with convection, NBE .=',I4,/,
     *        8X,'Elements, their LOCAL nodes and convective',/,
     *        8X,'heat transfer data:',/,
     *        8X,'Ele. No.',4X,'End Nodes',8X,'Film Coeff.',6X,
     *           'T-Infinity',/)
  450 FORMAT (/,16X,'A  VISCOUS INCOMPRESSIBLE FLOW IS ANALYZED')
  460 FORMAT (/,5X,'PARAMETERS OF THE FLUID FLOW PROBLEM:',//,
     *        8X,'Viscosity of the fluid, VISCSITY ........=',E12.4,/,
     *        8X,'Penalty parameter, PENALTY .............=',E12.4,/)
  470 FORMAT (/,16X,'A  2-D  ELASTICITY   PROBLEM  IS  ANALYZED')
  480 FORMAT (/,5X,'NUMERICAL INTEGRATION DATA:',//,
     *        8X,'Full Integration polynomial degree, IPDF =',I4,/,
     *        8X,'Number of full integration points,  NIPF =',I4,/,
     *        8X,'Reduced Integration polynomial deg.,IPDR =',I4,/,
     *        8X,'No.  of reduced integration points, NIPR =',I4,/,
     *        8X,'Integ. poly. deg. for stress comp., ISTR =',I4,/,
     *        8X,'No. of integ. pts. for stress comp.,NSTR =',I4,/)
  485 FORMAT (/,5X,'NUMERICAL INTEGRATION DATA:',//,
     *        8X,'Full quadrature (IPDF x IPDF) rule, IPDF =',I4,/,
     *        8X,'Reduced quadrature (IPDR x IPDR),   IPDR =',I4,/,
     *        8X,'Quadrature rule used in postproc.,  ISTR =',I4,/)
  490 FORMAT (9X,'**PLANE STRAIN assumption is selected by user**',/)
  500 FORMAT (/,16X,'A  PLATE   BENDING   PROBLEM  IS  ANALYZED')
  505 FORMAT (16X,  '*** using the shear deformation theory ***')
  506 FORMAT (16X,  '**** using the classical plate theory ****')
  510 FORMAT (/,8X,'***PLANE STRESS assumption is selected by user**',/)
  520 FORMAT (/,5X,'MATERIAL PROPERTIES OF THE SOLID ANALYZED:',//,
     *        8X,'Thickness of the body, THKNS ............=',E12.4,/,
     *        8X,'Modulus of elasticity, E1 ..............=',E12.4,/,
     *        8X,'Modulus of elasticity, E2 ..............=',E12.4,/,
     *        8X,'Poisson s ratio, ANU12 .................=',E12.4,/,
     *        8X,'Shear modulus, G12 .....................=',E12.4)
  530 FORMAT (8X,'Shear modulus, G13 .....................=',E12.4,/,
     *        8X,'Shear modulus, G23 .....................=',E12.4,/)
  540 FORMAT (/,5X,'PARAMETERS OF THE DYNAMIC ANALYSIS:',//,
     *        8X,'Coefficient, C0 ........................=',E12.4,/,
     *        8X,'Coefficient, CX ........................=',E12.4,/,
     *        8X,'Coefficient, CY ........................=',E12.4)
  550 FORMAT (8X,'Time increment used, DT ................=',E12.4,/,
     *        8X,'Parameter, ALFA ........................=',E12.4,/,
     *        8X,'Parameter, GAMA ........................=',E12.4,/,
     *        8X,'Number of time steps used, NTIME ........=',I4,/,
     *        8X,'Time step at which load is removed, NSTP.=',I4,/,
     *        8X,'Time interval at which soln. is printed..=',I4,/)
  560 FORMAT (/,5X,'FINITE ELEMENT MESH INFORMATION:',//,
     *        8X,'Element type: 0 = Triangle; >0 = Quad.)..=',I4,/,
     *        8X,'Number of nodes per element, NPE .......=',I4,/,
     *        8X,'No. of primary deg. of freedom/node, NDF =',I4,/,
     *        8X,'Number of elements in the mesh, NEM .....=',I4,/,
     *        8X,'Number of nodes in the mesh, NNM ........=',I4,/,
     *        8X,'Number of equations to be solved, NEQ ...=',I4,/,
     *        8X,'Half bandwidth of the matrix GLK, NHBW ..=',I4)
  570 FORMAT (8X,'Mesh subdivisions, NX and NY ...........=',2I4,/)
  580 FORMAT (5X,'Node   x-coord.   y-coord.    Speci. primary & seconda
     *ry variables',/,38X,'(0, unspecified; >0, specified)',
     *               /,41X,'Primary DOF  Secondary DOF')
  590 FORMAT (/,5X,'*TIME* =',E12.5,5X,'Time Step Number =',I3)
  600 FORMAT (/,5X,'*TIME* =',E12.5,' (Initial acceleration vector:)',/)
  610 FORMAT (/,5X,'Element coefficient matrix: ',/)
  620 FORMAT (/,5X,'Element mass matrix: ',/)
  630 FORMAT (/,5X,'Element source vector:',/)
  640 FORMAT (/,5X,'Global coefficient matrix (upper band):',/)
  650 FORMAT (/,5X,'Global source vector:',/)
  655 FORMAT (/,5X,'Global mass matrix (full form):',/)
  660 FORMAT (/,5X,'S O L U T I O N :',/)
  665 FORMAT (/,8X,'Number of Jacobi iterations ..... NROT =',I6,//)
  666 FORMAT (5X,'Nodal values of W,xy for conforming plate element:',/)
  670 FORMAT (5X,'Node    x-coord.      y-coord.    Primary DOF')
  680 FORMAT (5X,'Node    x-coord.      y-coord.     Value of u',
```

```
     *              '   Value of v')
  690 FORMAT (5X,'Node    x-coord.      y-coord.     deflec. w',
     *              '    x-rotation    y-rotation')
  700 FORMAT (/,5X,'Connectivity Matrix, [NOD]',/)
  710 FORMAT (8X,'No. of specified PRIMARY variables, NSPV =',I4)
  715 FORMAT (8X,'No. of speci.  SECONDARY variables, NSSV =',I4,/)
  720 FORMAT (6X,'Node  DOF     Value',/)
  730 FORMAT (4X,'The orientation of  gradient vector is measured from
     1the positive x-axis',/)
  740 FORMAT (4X,'x-coord.     y-coord.   -a11(du/dx)  -a22(du/dy)',
     1        3X,'Flux Mgntd  Orientation')
  750 FORMAT (4X,'x-coord.     y-coord.    a22(du/dy)  -a11(du/dx)',
     1        3X,'Flux Mgntd  Orientation')
  760 FORMAT (5X,'x-coord.     y-coord.     sigma-x      sigma-y',
     *'     sigma-xy    pressure')
  770 FORMAT (5X,'x-coord.     y-coord.     sigma-x      sigma-y',
     *'     sigma-xy')
  780 FORMAT (5X,'                          sigma-xz     sigma-yz')
  790 FORMAT (/,8X,'*** A mesh of   TRIANGLES    is chosen by user ***')
  800 FORMAT (/,8X,'*** A mesh of QUADRILATERALS is chosen by user ***')
  810 FORMAT (/,8X,'******* An EIGENVALUE PROBLEM  is analyzed *******')
  820 FORMAT (/,8X,'******** A TRANSIENT PROBLEM  is analyzed ********')
  830 FORMAT (/,8X,'******* A STEADY-STATE PROBLEM is analyzed *******')
  840 FORMAT(/,3X,'Eigenvalue(',I3,') =',E15.6,3X,'Frequency =',E13.5)
  845 FORMAT(8X,'E I G E N V A L U E (',I3,') =',E15.6)
  850 FORMAT(/,8X,'E I G E N V E C T O R :',/)
  860 FORMAT (8X,I5,5X,2I5,6X,E13.5,5X,E13.5)
  870 FORMAT (5X,I3,2E12.4,8X,I9,9X,I5)
  880 FORMAT (5X,I3,2E12.4,7X,3I4,2X,3I4)
  885 FORMAT (5X,I3,2E12.4,5X,4I4,2X,4I4)
  890 FORMAT (12X,'OUTPUT  from  program *** FEM2D *** by J. N. REDDY')
  900 FORMAT (10X,10I5)
  910 FORMAT (2X,70(' '),/)
  920 FORMAT (5X,I3,2E12.4,8X,2I5,4X,2I5)
  930 FORMAT (8X,5E14.5)
  940 FORMAT (2X,65('_'),/)
  950 FORMAT (5X,I3,5E14.5)
  960 FORMAT (5X,I5,I4,E14.5)
  970 FORMAT (2X,77('_'),/)
  980 FORMAT (/,3X,'*** THE SOLUTION HAS REACHED  A  STEADY STATE ***')
  990 FORMAT (/,3X,'**TRIANGULAR ELEMENTS ARE NOT ALLOWED FOR PLATES**')
  991 FORMAT (/,3X,'*STABILITY ANALYSIS IS ONLY FOR BENDING OF PLATES*',
     *        /,3X,'**** according to the  classical plate theory ****')
      END


      SUBROUTINE EGNSOLVR(N,A,B,XX,X,NEGN,NR,MXNEQ)
C     __________________________________________________________________
C
C       Subroutine 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(750,750),VT(750,750),W(750,750),IH(750)
C
C     Call 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 BOUNDARY(ISPV,ISSV,MAXSPV,MAXSSV,NDF,NCMAX,NRMAX,NEQ,
     *                    NHBW,NSPV,NSSV,S,SL,VSPV,VSSV,NCOUNT,INTIAL)
C     _________________________________________________________________
C
C     Called in MAIN to implement specified values of the primary and
C     secondary variables by modifying the coefficient matrix [S] and
C     (banded and symmetric) and the right-hand side vector {SL}.
C     _________________________________________________________________
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION S(NRMAX,NCMAX),SL(NRMAX),ISPV(MAXSPV,2),VSPV(MAXSPV),
     *          ISSV(MAXSSV,2),VSSV(MAXSSV)
      COMMON/IO/IN,ITT
C
      IF(NSSV.NE.0) THEN
        IF(INTIAL.EQ.0 .OR. NCOUNT.NE. 1) THEN
C
C     Implement specified values of the SECONDARY VARIABLES:____________
C
            DO 10 I=1,NSSV
            II=(ISSV(I,1)-1)*NDF+ISSV(I,2)
   10         SL(II)=SL(II)+VSSV(I)
        ENDIF
      ENDIF
C
C     Implement specified values of the PRIMARY VARIABLES:______________
C
      IF(NSPV.NE.0) THEN
        DO 50 NB=1,NSPV
```

```
         IE=(ISPV(NB,1)-1)*NDF+ISPV(NB,2)
         VALUE=VSPV(NB)
         IT=NHBW-1
         I=IE-NHBW
         DO 30 II=1,IT
         I=I+1
         IF(I.GE.1) THEN
             J=IE-I+1
             SL(I)=SL(I)-S(I,J)*VALUE
             S(I,J)=0.0
         ENDIF
   30      CONTINUE
         S(IE,1)=1.0
         SL(IE)=VALUE
         I=IE
         DO 40 II=2,NHBW
         I=I+1
         IF(I.LE.NEQ) THEN
            SL(I)=SL(I)-S(IE,II)*VALUE
            S(IE,II)=0.0
         ENDIF
   40      CONTINUE
   50      CONTINUE
       ENDIF
       RETURN
       END


       SUBROUTINE CONCTVTY(NELEM,NODES,MAXELM,MAXNOD,GLXY)
C      _______________________________________________________________
C
C      Generates nodal connectivity array for a specified type of mesh
C
C      NEL1    = First element in the row of elements
C      NELL    = Last element in the row
C      IELINC  = Increment from element to the next in the row
C      NODINC  = Node increment from one element to the next
C      NPE     = Number of nodes per element
C      NODE(I) = Global node numbers corresponding to the local nodes
C                of the first element in the row
C      _______________________________________________________________
C
       IMPLICIT REAL*8(A-H,O-Z)
       DIMENSION  NODES(MAXELM,9),GLXY(MAXNOD,2),NODE(9)
C
C      Read element data
C
       READ(5,*) NRECEL
       DO 30 IREC=1,NRECEL
       READ(5,*) NEL1,NELL,IELINC,NODINC,NPE,(NODE(I),I=1,NPE)
       IF(IELINC.LE.0) IELINC=1
       IF(NODINC.LE.0) NODINC=1
       IF(NELL.LE.NEL1) NELL=NEL1
       IF(NELL.GT.NELEM) THEN
         WRITE(6,60)
         STOP
       ELSE
         NINC=-1
         DO 20 N=NEL1,NELL,IELINC
         NINC=NINC+1
         DO 10 M=1,NPE
   10      NODES(N,M)=NODE(M)+NINC*NODINC
   20      CONTINUE
       ENDIF
   30 CONTINUE
C
       DO 50 N=1,NELEM
       SUMX=0.0
       SUMY=0.0
       NEN=NPE
       IF(NEN.NE.4) THEN
          DO 40 M=5,NEN
          MM=NODES(N,M)
          IF(M.NE.9 .OR. M.NE.6) THEN
             M4=NODES(N,M-4)
             M3=NODES(N,M-3)
```

```
            IF(M.EQ.8) M3=NODES(N,1)
            IF(GLXY(MM,1).EQ.1.E20)
     *           GLXY(MM,1)=0.5*(GLXY(M4,1)+GLXY(M3,1))
            IF(GLXY(MM,2).EQ.1.E20)
     *           GLXY(MM,2)=0.5*(GLXY(M4,2)+GLXY(M3,2))
            IF(NEN.NE.8) THEN
               SUMX=SUMX+GLXY(M4,1)
               SUMY=SUMY+GLXY(M4,2)
            ENDIF
         ELSE
            IF(GLXY(MM,1).EQ.1.E20) GLXY(MM,1)=0.25*SUMX
            IF(GLXY(MM,2).EQ.1.E20) GLXY(MM,2)=0.25*SUMY
         ENDIF
   40    CONTINUE
      ENDIF
   50 CONTINUE
   60 FORMAT(/,'MSG from CNCTVT: Element number exceeds maximum value')
      RETURN
      END


      SUBROUTINE EGNBNDRY(A,D,IBDY,ISPV,MXPV,NDF,NEQ,NEQR,NSPV,NRM)
C     ________________________________________________________________
C
C     Imposes specified homogeneous boundary conditions on the primary
C     variables  by  eliminating rows and columns corresponding to the
C     specified degrees of freedom
C     ________________________________________________________________
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NRM,NRM),D(NRM,NRM),ISPV(MXPV,2),IBDY(MXPV)
C
      DO 10 I=1,NSPV
   10 IBDY(I)=(ISPV(I,1)-1)*NDF+ISPV(I,2)
      DO 30 I=1,NSPV
      IMAX=IBDY(I)
      DO 20 J=I,NSPV
      IF(IBDY(J).GE.IMAX) THEN
        IMAX=IBDY(J)
        IKEPT=J
      ENDIF
   20 CONTINUE
      IBDY(IKEPT)=IBDY(I)
      IBDY(I)=IMAX
   30 CONTINUE
      NEQR = NEQ
      DO 80 I=1,NSPV
      IB=IBDY(I)
      IF(IB .LT. NEQR) THEN
        NEQR1=NEQR-1
        DO 60 II=IB,NEQR1
        DO 40 JJ=1,NEQR
        D(II,JJ)=D(II+1,JJ)
   40     A(II,JJ)=A(II+1,JJ)
        DO 50 JJ=1,NEQR
        D(JJ,II)=D(JJ,II+1)
   50     A(JJ,II)=A(JJ,II+1)
   60     CONTINUE
      ENDIF
      NEQR=NEQR-1
   80 CONTINUE
      RETURN
      END


      SUBROUTINE INVERSE(A,B)
      IMPLICIT REAL*8 (A-H,O-Z)
C     ________________________________________________________________
C
C     Called in SHAPERCT to compute the inverse of a 3x3 matrix,[A].
C     The inverse is stored in matrix [B]
C     ________________________________________________________________
C
      DIMENSION  A(3,3), B(3,3)
C
      G(Z1,Z2,Z3,Z4) = Z1*Z2 - Z3*Z4
```

```
      F(Z1,Z2,Z3,Z4) = G(Z1,Z2,Z3,Z4) / DET
      C1  = G(A(2,2),A(3,3),A(2,3),A(3,2))
      C2  = G(A(2,3),A(3,1),A(2,1),A(3,3))
      C3  = G(A(2,1),A(3,2),A(2,2),A(3,1))
      DET = A(1,1)*C1 + A(1,2)*C2 + A(1,3)*C3
      B(1,1) =  F(A(2,2),A(3,3),A(3,2),A(2,3))
      B(1,2) = -F(A(1,2),A(3,3),A(1,3),A(3,2))
      B(1,3) =  F(A(1,2),A(2,3),A(1,3),A(2,2))
      B(2,1) = -F(A(2,1),A(3,3),A(2,3),A(3,1))
      B(2,2) =  F(A(1,1),A(3,3),A(3,1),A(1,3))
      B(2,3) = -F(A(1,1),A(2,3),A(1,3),A(2,1))
      B(3,1) =  F(A(2,1),A(3,2),A(3,1),A(2,2))
      B(3,2) = -F(A(1,1),A(3,2),A(1,2),A(3,1))
      B(3,3) =  F(A(1,1),A(2,2),A(2,1),A(1,2))
      RETURN
      END


      SUBROUTINE DATAECHO(IN,IT)
C
      DIMENSION AA(20)
      WRITE(IT,40)
   10 CONTINUE
      READ(IN,30,END=20) AA
      WRITE(IT,30) 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 ****',/)
      END


      SUBROUTINE ELKMFRCT(NEIGN,NPE,NN,ITYPE,ITEM)
C     ________________________________________________________________
C
C     Called in MAIN to compute element  matrices based on  linear and
C     quadratic ReCTangular elements and isoparametric formulation for
C     for all classes of problems of the book. Reduced  integration is
C     used on certain terms of viscous flow and plate bending problems.
C     ________________________________________________________________
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/STF/ELF(27),ELK(27,27),ELM(27,27),ELXY(9,2),ELU(27),
     1           ELV(27),ELA(27),A1,A2,A3,A4,A5
      COMMON/PST/A10,A1X,A1Y,A20,A2X,A2Y,A00,C0,CX,CY,F0,FX,FY,
     1           C44,C55,VISCSITY,PENALTY,CMAT(3,3)
      COMMON/SHP/SF(9),GDSF(2,9),SFH(16),GDSFH(2,16),GDDSFH(3,16)
      COMMON/PNT/IPDF,IPDR,NIPF,NIPR
      DIMENSION  GAUSPT(5,5),GAUSWT(5,5)
      COMMON/IO/IN,ITT
C
      DATA GAUSPT/5*0.0D0, -0.57735027D0, 0.57735027D0, 3*0.0D0,
     2  -0.77459667D0, 0.0D0, 0.77459667D0, 2*0.0D0, -0.86113631D0,
     3  -0.33998104D0, 0.33998104D0, 0.86113631D0, 0.0D0, -0.90617984D0,
     4  -0.53846931D0,0.0D0,0.53846931D0,0.90617984D0/
C
      DATA GAUSWT/2.0D0, 4*0.0D0, 2*1.0D0, 3*0.0D0, 0.55555555D0,
     2   0.88888888D0, 0.55555555D0, 2*0.0D0, 0.34785485D0,
     3 2*0.65214515D0, 0.34785485D0, 0.0D0, 0.23692688D0,
     4   0.47862867D0, 0.56888888D0, 0.47862867D0, 0.23692688D0/
C
      NDF = NN/NPE
      IF(ITYPE.LE.3) THEN
         NET=NPE
      ELSE
         NET=NN
      ENDIF
C
C     Initialize the arrays
C
      DO 120 I = 1,NN
      IF(NEIGN.EQ.0) THEN
```

```
            ELF(I)  = 0.0
         ENDIF
         DO 120 J = 1,NN
         IF(ITEM.NE.0) THEN
            ELM(I,J)= 0.0
         ENDIF
  120    ELK(I,J)= 0.0
C
C        Do-loops on numerical (Gauss) integration begin here. Subroutine
C        SHAPERCT (SHAPE functions for ReCTangular elements) is called here
C
         DO 200 NI = 1,IPDF
         DO 200 NJ = 1,IPDF
         XI  = GAUSPT(NI,IPDF)
         ETA = GAUSPT(NJ,IPDF)
         CALL SHAPERCT (NPE,XI,ETA,DET,ELXY,NDF,ITYPE)
         CNST = DET*GAUSWT(NI,IPDF)*GAUSWT(NJ,IPDF)
         X=0.0
         Y=0.0
         DO 140 I=1,NPE
         X=X+ELXY(I,1)*SF(I)
  140    Y=Y+ELXY(I,2)*SF(I)
C
         IF(NEIGN.EQ.0) THEN
            SOURCE=F0+FX*X+FY*Y
         ENDIF
         IF(ITEM.NE.0) THEN
            IF(ITYPE.LE.2)THEN
               CT=C0+CX*X+CY*Y
            ENDIF
         ENDIF
         IF(ITYPE.LE.0) THEN
            A11=A10+A1X*X+A1Y*Y
            A22=A20+A2X*X+A2Y*Y
         ENDIF
C
         II=1
         DO 180 I=1,NET
         JJ=1
         DO 160 J=1,NET
         IF(ITYPE.LE.3) THEN
            S00=SF(I)*SF(J)*CNST
            S11=GDSF(1,I)*GDSF(1,J)*CNST
            S22=GDSF(2,I)*GDSF(2,J)*CNST
            S12=GDSF(1,I)*GDSF(2,J)*CNST
            S21=GDSF(2,I)*GDSF(1,J)*CNST
         ENDIF
         IF(ITYPE.EQ.0) THEN
C
C        Heat transfer and like problems (i.e. single DOF problems):_______
C
            ELK(I,J) = ELK(I,J) + A11*S11 + A22*S22 + A00*S00
            IF(ITEM.NE.0) THEN
               ELM(I,J) = ELM(I,J) + CT*S00
            ENDIF
         ELSE
            IF(ITYPE.EQ.1) THEN
C
C        Viscous incompressible fluids:                                  __
C        Compute coefficients associated with viscous terms (full integ.)
C
               ELK(II,JJ)     = ELK(II,JJ)      + VISCSITY*(2.0*S11 + S22)
               ELK(II+1,JJ)   = ELK(II+1,JJ)    + VISCSITY*S12
               ELK(II,JJ+1)   = ELK(II,JJ+1)    + VISCSITY*S21
               ELK(II+1,JJ+1)= ELK(II+1,JJ+1)  + VISCSITY*(S11 + 2.0*S22)
               IF(ITEM.NE.0) THEN
                  ELM(II,JJ)     = ELM(II,JJ)      + CT*S00
                  ELM(II+1,JJ+1)= ELM(II+1,JJ+1)  + CT*S00
               ENDIF
            ELSE
               IF(ITYPE.EQ.2) THEN
C
C        Plane elasticity problems:_________________________________________
C
                  ELK(II,JJ)    =ELK(II,JJ)    +CMAT(1,1)*S11+CMAT(3,3)*S22
                  ELK(II,JJ+1)  =ELK(II,JJ+1)  +CMAT(1,2)*S12+CMAT(3,3)*S21
```

```
               ELK(II+1,JJ)  =ELK(II+1,JJ)  +CMAT(1,2)*S21+CMAT(3,3)*S12
               ELK(II+1,JJ+1)=ELK(II+1,JJ+1)+CMAT(3,3)*S11+CMAT(2,2)*S22
               IF(ITEM.NE.0) THEN
                  ELM(II,JJ)     = ELM(II,JJ)     + CT*S00
                  ELM(II+1,JJ+1)= ELM(II+1,JJ+1) + CT*S00
               ENDIF
            ELSE
               IF(ITYPE.GE.4) THEN
C
C     Classical plate theory:____________________________________________
C
                  BM1=CMAT(1,1)*GDDSFH(1,J)+CMAT(1,2)*GDDSFH(2,J)
                  BM2=CMAT(1,2)*GDDSFH(1,J)+CMAT(2,2)*GDDSFH(2,J)
                  BM6=2.0*CMAT(3,3)*GDDSFH(3,J)
                  ELK(I,J)=ELK(I,J)+CNST*(GDDSFH(1,I)*BM1
     *                     +GDDSFH(2,I)*BM2+2.0*GDDSFH(3,I)*BM6)
                  IF(ITEM.NE.0) THEN
                     S00=SFH(I)*SFH(J)*CNST
                     SXX=GDSFH(1,I)*GDSFH(1,J)*CNST
                     SYY=GDSFH(2,I)*GDSFH(2,J)*CNST
                     IF(NEIGN.LE.1) THEN
                        ELM(I,J)=ELM(I,J) + C0*S00+CX*SXX+CY*SYY
                     ELSE
                        SXY=GDSFH(1,I)*GDSFH(2,J)*CNST
                        SYX=GDSFH(2,I)*GDSFH(1,J)*CNST
                        ELM(I,J)=ELM(I,J) + C0*SXX + CX*SYY
     *                                 + CY*(SXY + SYX)
                     ENDIF
                  ENDIF
               ELSE
C
C     Shear deformable plate theory:___________________________________
C
                  ELK(II+1,JJ+1)= ELK(II+1,JJ+1) +
     *                           CMAT(1,1)*S11+CMAT(3,3)*S22
                  ELK(II+1,JJ+2)= ELK(II+1,JJ+2) +
     *                           CMAT(1,2)*S12+CMAT(3,3)*S21
                  ELK(II+2,JJ+1)= ELK(II+2,JJ+1) +
     *                           CMAT(3,3)*S12+CMAT(1,2)*S21
                  ELK(II+2,JJ+2)= ELK(II+2,JJ+2) +
     *                           CMAT(3,3)*S11+CMAT(2,2)*S22
                  IF(ITEM.NE.0) THEN
                     IF(NEIGN.LE.1) THEN
                        ELM(II,JJ)     = ELM(II,JJ)     + C0*S00
                        ELM(II+1,JJ+1)= ELM(II+1,JJ+1) + CX*S00
                        ELM(II+2,JJ+2)= ELM(II+2,JJ+2) + CY*S00
                     ELSE
                        ELM(II,JJ)     = ELM(II,JJ)+C0*S11+CX*S22
     *                                  +CY*(S12+S21)
                     ENDIF
                  ENDIF
               ENDIF
            ENDIF
         ENDIF
      ENDIF
  160 JJ = NDF*J+1
      IF(NEIGN.EQ.0) THEN
C
C     Source of the form fx = F0 + FX*X + FY*Y is assumed
C
         IF(ITYPE.LE.3) THEN
            L=(I-1)*NDF+1
            ELF(L) = ELF(L)+CNST*SF(I)*SOURCE
         ELSE
            ELF(I) = ELF(I)+CNST*SFH(I)*SOURCE
         ENDIF
      ENDIF
  180 II = NDF*I+1
  200 CONTINUE
C
      IF(ITYPE.EQ.1 .OR. ITYPE.EQ.3) THEN
C
C     Use reduced integration to evaluate coefficients associated with
C     penalty terms for flows and transverse shear terms for plates.
C
         DO 280 NI=1,IPDR
```

```
         DO 280 NJ=1,IPDR
         XI  = GAUSPT(NI,IPDR)
         ETA = GAUSPT(NJ,IPDR)
         CALL SHAPERCT (NPE,XI,ETA,DET,ELXY,NDF,ITYPE)
         CNST=DET*GAUSWT(NI,IPDR)*GAUSWT(NJ,IPDR)
C
         II=1
         DO 260 I=1,NPE
         JJ = 1
         DO 240 J=1,NPE
         S11=GDSF(1,I)*GDSF(1,J)*CNST
         S22=GDSF(2,I)*GDSF(2,J)*CNST
         S12=GDSF(1,I)*GDSF(2,J)*CNST
         S21=GDSF(2,I)*GDSF(1,J)*CNST
         IF(ITYPE.EQ.1) THEN
C
C     Viscous incompressible fluids (penalty terms):____________________
C
            ELK(II,JJ)     = ELK(II,JJ)      + PENALTY*S11
            ELK(II+1,JJ)   = ELK(II+1,JJ)    + PENALTY*S21
            ELK(II,JJ+1)   = ELK(II,JJ+1)    + PENALTY*S12
            ELK(II+1,JJ+1)= ELK(II+1,JJ+1)  + PENALTY*S22
         ELSE
C
C     Shear deformable plates (transverse shear terms):_________________
C
            S00=SF(I)*SF(J)*CNST
            S10 = GDSF(1,I)*SF(J)*CNST
            S01 = SF(I)*GDSF(1,J)*CNST
            S20 = GDSF(2,I)*SF(J)*CNST
            S02 = SF(I)*GDSF(2,J)*CNST
            ELK(II,JJ)     = ELK(II,JJ)      + C55*S11+C44*S22
            ELK(II,JJ+1)   = ELK(II,JJ+1)    + C55*S10
            ELK(II+1,JJ)   = ELK(II+1,JJ)    + C55*S01
            ELK(II,JJ+2)   = ELK(II,JJ+2)    + C44*S20
            ELK(II+2,JJ)   = ELK(II+2,JJ)    + C44*S02
            ELK(II+1,JJ+1)= ELK(II+1,JJ+1)  + C55*S00
            ELK(II+2,JJ+2)= ELK(II+2,JJ+2)  + C44*S00
         ENDIF
  240 JJ=NDF*J+1
  260 II=NDF*I+1
  280 CONTINUE
      ENDIF
      RETURN
      END


      SUBROUTINE ELKMFTRI(NEIGN,NPE,NN,ITYPE,ITEM)
C     _________________________________________________________________
C
C     Called in MAIN to compute element  matrices based on  linear and
C     quadratic  TRIangular elements and isoparametric formulation for
C     for all classes of problems of the book. Reduced  integration is
C     used on certain terms of viscous flow and plate bending problems.
C     _________________________________________________________________
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/STF/ELF(27),ELK(27,27),ELM(27,27),ELXY(9,2),ELU(27),
     1           ELV(27),ELA(27),A1,A2,A3,A4,A5
      COMMON/PST/A10,A1X,A1Y,A20,A2X,A2Y,A00,C0,CX,CY,F0,FX,FY,
     1           C44,C55,VISCSITY,PENALTY,CMAT(3,3)
      COMMON/QUAD/AL1(7,5),AL2(7,5),AL3(7,5),ALWT(7,5)
      COMMON/PNT/IPDF,IPDR,NIPF,NIPR
      COMMON/SHP/SF(9),GDSF(2,9),SFH(16),GDSFH(2,16),GDDSFH(3,16)
      COMMON/IO/IN,ITT
C
      NDF = NN/NPE
C
C     Call subroutine QUADRature for TRIangle to compute arrays of
C     integration points and weights for the given NIPF and IPDF
C
      CALL QUADRTRI (NIPF,IPDF)
C
C     Initialize the arrays
C
      DO 120 I = 1,NN
```

```
      IF(NEIGN.EQ.0) THEN
         ELF(I)  = 0.0
      ENDIF
      DO 120 J = 1,NN
      IF(ITEM.NE.0) THEN
         ELM(I,J)= 0.0
      ENDIF
  120 ELK(I,J)= 0.0
C
C     Do-loop on the numerical integration begins here
C
      DO 200 NI = 1,NIPF
      AC1 = AL1(NI,IPDF)
      AC2 = AL2(NI,IPDF)
      AC3 = AL3(NI,IPDF)
      CALL SHAPETRI(NPE,AC1,AC2,AC3,DET,ELXY)
      CNST = 0.50D0*DET*ALWT(NI,IPDF)
      X=0.0
      Y=0.0
      DO 140 I=1,NPE
      X=X+ELXY(I,1)*SF(I)
  140 Y=Y+ELXY(I,2)*SF(I)
C
      IF(NEIGN.EQ.0) THEN
         SOURCE=F0+FX*X+FY*Y
      ENDIF
      IF(ITEM.NE.0) THEN
         CT =C0+CX*X+CY*Y
      ENDIF
      IF(ITYPE.LE.0) THEN
         A11=A10+A1X*X+A1Y*Y
         A22=A20+A2X*X+A2Y*Y
      ENDIF
C
      II=1
      DO 180 I=1,NPE
      JJ=1
      DO 160 J=1,NPE
      S00=SF(I)*SF(J)*CNST
      S11=GDSF(1,I)*GDSF(1,J)*CNST
      S22=GDSF(2,I)*GDSF(2,J)*CNST
      S12=GDSF(1,I)*GDSF(2,J)*CNST
      S21=GDSF(2,I)*GDSF(1,J)*CNST
      IF(ITYPE.EQ.0) THEN
C
C     Heat transfer and like problems (i.e. single DOF problems):_______
C
         ELK(I,J) = ELK(I,J) + A11*S11 + A22*S22 + A00*S00
         IF(ITEM.NE.0) THEN
            ELM(I,J) = ELM(I,J) + CT*S00
         ENDIF
      ELSE
         IF(ITYPE.EQ.1) THEN
C
C     Viscous incompressible fluids:                                  __
C     Compute coefficients associated with viscous terms (full integ.)
C
            ELK(II,JJ)    = ELK(II,JJ)     + VISCSITY*(2.0*S11 + S22)
            ELK(II+1,JJ)  = ELK(II+1,JJ)   + VISCSITY*S12
            ELK(II,JJ+1)  = ELK(II,JJ+1)   + VISCSITY*S21
            ELK(II+1,JJ+1)= ELK(II+1,JJ+1) + VISCSITY*(S11 + 2.0*S22)
            IF(ITEM.NE.0) THEN
               ELM(II,JJ)    = ELM(II,JJ)     + CT*S00
               ELM(II+1,JJ+1)= ELM(II+1,JJ+1) + CT*S00
            ENDIF
         ELSE
C
C     Plane elasticity problems:________________________________________
C
            ELK(II,JJ)    =ELK(II,JJ)    +CMAT(1,1)*S11+CMAT(3,3)*S22
            ELK(II,JJ+1)  =ELK(II,JJ+1)  +CMAT(1,2)*S12+CMAT(3,3)*S21
            ELK(II+1,JJ)  =ELK(II+1,JJ)  +CMAT(1,2)*S21+CMAT(3,3)*S12
            ELK(II+1,JJ+1)=ELK(II+1,JJ+1)+CMAT(3,3)*S11+CMAT(2,2)*S22
            IF(ITEM.NE.0) THEN
               ELM(II,JJ)    = ELM(II,JJ)     + CT*S00
               ELM(II+1,JJ+1)= ELM(II+1,JJ+1) + CT*S00
```

```
            ENDIF
         ENDIF
      ENDIF
  160 JJ = NDF*J+1
      IF(NEIGN.EQ.0) THEN
C
C     Source of the form fx = F0 + FX*X + FY*Y is assumed
C
         L=(I-1)*NDF+1
         ELF(L) = ELF(L)+CNST*SF(I)*SOURCE
      ENDIF
  180 II = NDF*I+1
  200 CONTINUE
C
      IF(ITYPE.EQ.1 .OR. ITYPE.EQ.3) THEN
C
C     Use reduced integration to evaluate coefficients associated with
C     penalty terms for flows and transverse shear terms for plates.
C
C     Call subroutine QUADRature for TRIangles to compute arrays of integration
C     points and weights for the given NIPR and IPDR
C
        CALL QUADRTRI (NIPR,IPDR)
C
        DO 280 NI=1,NIPR
        AC1 = AL1(NI,IPDR)
        AC2 = AL2(NI,IPDR)
        AC3 = AL3(NI,IPDR)
        CALL SHAPETRI(NPE,AC1,AC2,AC3,DET,ELXY)
        CNST = 0.50D0*DET*ALWT(NI,IPDR)
C
        II=1
        DO 260 I=1,NPE
        JJ = 1
        DO 240 J=1,NPE
        S11=GDSF(1,I)*GDSF(1,J)*CNST
        S22=GDSF(2,I)*GDSF(2,J)*CNST
        S12=GDSF(1,I)*GDSF(2,J)*CNST
        S21=GDSF(2,I)*GDSF(1,J)*CNST
        IF(ITYPE.EQ.1) THEN
C
C     Viscous incompressible fluids (penalty terms):____________________
C
           ELK(II,JJ)     = ELK(II,JJ)      + PENALTY*S11
           ELK(II+1,JJ)   = ELK(II+1,JJ)    + PENALTY*S21
           ELK(II,JJ+1)   = ELK(II,JJ+1)    + PENALTY*S12
           ELK(II+1,JJ+1)= ELK(II+1,JJ+1)  + PENALTY*S22
        ELSE
C
C     Shear deformable plates (transverse shear terms):_________________
C
           S00=SF(I)*SF(J)*CNST
           S10 = GDSF(1,I)*SF(J)*CNST
           S01 = SF(I)*GDSF(1,J)*CNST
           S20 = GDSF(2,I)*SF(J)*CNST
           S02 = SF(I)*GDSF(2,J)*CNST
           ELK(II,JJ)     = ELK(II,JJ)      + C55*S11+C44*S22
           ELK(II,JJ+1)   = ELK(II,JJ+1)    + C55*S10
           ELK(II+1,JJ)   = ELK(II+1,JJ)    + C55*S01
           ELK(II,JJ+2)   = ELK(II,JJ+2)    + C44*S20
           ELK(II+2,JJ)   = ELK(II+2,JJ)    + C44*S02
           ELK(II+1,JJ+1)= ELK(II+1,JJ+1)  + C55*S00
           ELK(II+2,JJ+2)= ELK(II+2,JJ+2)  + C44*S00
        ENDIF
  240   JJ=NDF*J+1
  260   II=NDF*I+1
  280   CONTINUE
      ENDIF
      RETURN
      END


      SUBROUTINE JACOBI (N,Q,JVEC,M,V,X,IH,MXNEQ)
C     __________________________________________________________________
C
C       Called in EGNSOLVR to diagonalize [Q] by successive rotations
```

```fortran
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 MESH2DG(NELEM,NNODE,NOD,MAXELM,MAXNOD,GLXY)
C     ________________________________________________________________
C
C     Called in MAIN to generate nodal point coordinates for specified
C     type meshes (see Fig. 13.4.2 for examples)
C
C     NOD1  = First node number in the line segment
C     NODL  = Last node number in the line segment
C     NODINC= Node increment from one node to the next along the line
C     X1,Y1 = Global coordinates of the first node on the line
C     XL,YL = Global coordinates of the last node on the line
C     RATIO = The ratio of the first element to the last element
C     ________________________________________________________________
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION  GLXY(MAXNOD,2),NOD(MAXELM,9)
C
      DO 10 I=1,NNODE
      GLXY(I,1)=1.E20
   10 GLXY(I,2)=1.E20
C
C     Read number of the records (line segmments) and data in each line
C
      READ(5,*)NRECL
      DO 30 IREC=1,NRECL
      READ(5,*)NOD1,NODL,NODINC,X1,Y1,XL,YL,RATIO
      IF(NODL.LT.NOD1) NODL = NOD1
      IF(NODL.NE.NOD1) THEN
         IF(NODINC.LE.0) NODINC = 1
         IF(RATIO.LE.0.0) RATIO=1.0
```

```
         NODIF = (NODL-NOD1)/NODINC
         XL1=XL-X1
         YL1=YL-Y1
         GLXY(NOD1,1)=X1
         GLXY(NOD1,2)=Y1
         ALNGTH=DSQRT(XL1*XL1+YL1*YL1)
         ALINC=(2.0*ALNGTH/NODIF)*RATIO/(RATIO+1.0)
         ALRAT=ALINC/RATIO
         IF(NODIF.NE.1) DEL=(ALINC-ALRAT)/(NODIF-1)
         IF(NODIF.EQ.1) DEL=0.0
         SUM=0.0
         I=-1
         DO 20 N=1,NODIF
         I=I+1
         SUM=SUM+ALINC-I*DEL
         NI=NOD1+N*NODINC
         GLXY(NI,1)=X1+XL1*SUM/ALNGTH
         GLXY(NI,2)=Y1+YL1*SUM/ALNGTH
   20    CONTINUE
      ENDIF
   30 CONTINUE
      CALL CONCTVTY(NELEM,NOD,MAXELM,MAXNOD,GLXY)
      RETURN
      END


      SUBROUTINE MESH2DR(IEL,IELTYP,NX,NY,NPE,NNM,NEM,NOD,DX,DY,X0,Y0,
     1                   GLXY,MAXELM,MAXNOD,MAXNX,MAXNY)
C     ________________________________________________________________
C
C     Called in  MAIN to compute arrays [NOD] & [GLXY] for rectangular
C     domains. The domain is divided  into  NX  subdivisions along the
C     x-direction and NY subdivisions in the y-direction. The subdivi-
C     sions  define  rectangular elements of the type required.  For a
C     triangular element mesh, the subdivision defines two linear ele-
C     ments per a rectangular element with their common diagonal being
C     inclined to the right (see Fig. 13.4.1 of the text).
C     ________________________________________________________________
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NOD(MAXELM,9),GLXY(MAXNOD,2),DX(MAXNX),DY(MAXNY)
      COMMON/IO/IN,ITT
C
      NEX1 = NX+1
      NEY1 = NY+1
      NXX  = IEL*NX
      NYY  = IEL*NY
      NXX1 = NXX + 1
      NYY1 = NYY + 1
      NEM  = NX*NY
      IF(IELTYP.EQ.0)NEM=2*NX*NY
      NNM=NXX1*NYY1
      IF(NPE.EQ.8)NNM = NXX1*NYY1 - NX*NY
      IF(IELTYP.EQ.0) THEN
C
C     Generate the array [NOD]:_________________________________________
C     TRIANGULAR ELEMENTS
C
         NX2=2*NX
         NY2=2*NY
         NOD(1,1) = 1
         NOD(1,2) = IEL+1
         NOD(1,3) = IEL*NXX1+IEL+1
         IF(NPE .GT. 3) THEN
            NOD(1,4) = 2
            NOD(1,5) = NXX1 + 3
            NOD(1,6) = NXX1 + 2
         ENDIF
         NOD(2,1) = 1
         NOD(2,2) = NOD(1,3)
         NOD(2,3) = IEL*NXX1+1
         IF(NPE .GT. 3) THEN
            NOD(2,4) = NOD(1,6)
            NOD(2,5) = NOD(1,3) - 1
            NOD(2,6) = NOD(2,4) - 1
         ENDIF
```

```
         K=3
         DO 60 IY=1,NY
         L=IY*NX2
         M=(IY-1)*NX2
         IF(NX.GT.1) THEN
            DO 30 N=K,L,2
            DO 20 I=1,NPE
            NOD(N,I)  = NOD(N-2,I)+IEL
   20       NOD(N+1,I)= NOD(N-1,I)+IEL
   30       CONTINUE
         ENDIF
         IF(IY.LT.NY) THEN
            DO 40 I=1,NPE
            NOD(L+1,I)=NOD(M+1,I)+IEL*NXX1
   40       NOD(L+2,I)=NOD(M+2,I)+IEL*NXX1
         ENDIF
   60    K=L+3
      ELSE
C
C     RECTANGULAR ELEMENTS
C
         K0 = 0
         IF(NPE .EQ. 9)K0=1
         NOD(1,1) = 1
         NOD(1,2) = IEL+1
         NOD(1,3) = NXX1+(IEL-1)*NEX1+IEL+1
         IF(NPE .EQ. 9) NOD(1,3)=4*NX+5
         NOD(1,4) = NOD(1,3) - IEL
         IF(NPE .GT. 4) THEN
            NOD(1,5) = 2
            NOD(1,6) = NXX1 + (NPE-6)
            NOD(1,7) = NOD(1,3) - 1
            NOD(1,8) = NXX1+1
            IF(NPE .EQ. 9) THEN
            NOD(1,9)=NXX1+2
            ENDIF
         ENDIF
         IF(NY .GT. 1) THEN
            M = 1
            DO 110 N = 2,NY
            L = (N-1)*NX + 1
            DO 100 I = 1,NPE
  100       NOD(L,I) = NOD(M,I)+NXX1+(IEL-1)*NEX1+K0*NX
  110       M=L
         ENDIF
C
         IF(NX .GT .1) THEN
            DO 140 NI = 2,NX
            DO 120  I = 1,NPE
            K1 = IEL
            IF(I .EQ. 6 .OR. I .EQ. 8)K1=1+K0
  120          NOD(NI,I) = NOD(NI-1,I)+K1
            M = NI
            DO 140 NJ = 2,NY
            L = (NJ-1)*NX+NI
            DO 130 J = 1,NPE
  130       NOD(L,J) = NOD(M,J)+NXX1+(IEL-1)*NEX1+K0*NX
  140       M = L
         ENDIF
      ENDIF
C
C     Generate the global coordinates of the nodes, [GLXY]:_____________
C
      DX(NEX1)=0.0
      DY(NEY1)=0.0
      XC=X0
      YC=Y0
      IF(NPE .EQ. 8) THEN
         DO 180 NI = 1, NEY1
         I = (NXX1+NEX1)*(NI-1)+1
         J = 2*NI-1
         GLXY(I,1) = XC
         GLXY(I,2) = YC
         DO 150 NJ = 1,NX
         DELX=0.5*DX(NJ)
         I=I+1
```

```
         GLXY(I,1) = GLXY(I-1,1)+DELX
         GLXY(I,2) = YC
         I=I+1
         GLXY(I,1) = GLXY(I-1,1)+DELX
         GLXY(I,2) = YC
  150    CONTINUE
         IF(NI.LE.NY) THEN
            I = I+1
            YC= YC+0.5*DY(NI)
            GLXY(I,1) = XC
            GLXY(I,2) = YC
            DO 160 II = 1, NX
            I = I+1
            GLXY(I,1) = GLXY(I-1,1)+DX(II)
  160       GLXY(I,2) = YC
         ENDIF
  180       YC = YC+0.5*DY(NI)
C
      ELSE
         YC=Y0
         DO 200 NI = 1, NEY1
         XC = X0
         I = NXX1*IEL*(NI-1)
         DO 190 NJ = 1, NEX1
         I=I+1
         GLXY(I,1) = XC
         GLXY(I,2) = YC
         IF(NJ.LT.NEX1) THEN
            IF(IEL.EQ.2) THEN
            I=I+1
            XC = XC + 0.5*DX(NJ)
            GLXY(I,1) = XC
            GLXY(I,2) = YC
            ENDIF
         ENDIF
  190    XC = XC + DX(NJ)/IEL
         XC = X0
         IF(IEL.EQ.2) THEN
            YC = YC + 0.5*DY(NI)
            DO 195 NJ = 1, NEX1
            I=I+1
            GLXY(I,1) = XC
            GLXY(I,2) = YC
            IF(NJ.LT.NEX1) THEN
               I=I+1
               XC = XC + 0.5*DX(NJ)
               GLXY(I,1) = XC
               GLXY(I,2) = YC
            ENDIF
  195       XC = XC + 0.5*DX(NJ)
         ENDIF
  200    YC = YC + DY(NI)/IEL
      ENDIF
      RETURN
      END


      SUBROUTINE MATRXMLT(MXNEQ,N,A,B,C)
C     ________________________________________________________________
C
C      Called in EGNSOLVR to computer 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 POSTPROC(ELXY,ITYPE,IELTYP,IGRAD,NDF,NPE,THKNS,ELU,
```

```
     *                     ISTR,NSTR)
C     ________________________________________________________________
C
C     Called in MAIN to compute the derivatives of the solution for
C     heat transfer and like problems, and stresses for fluid flow,
C     plane elasticity and plate bending problems.
C     ________________________________________________________________
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  ELXY(9,2),ELU(27),GAUSPT(4,4)
      COMMON/PST/A10,A1X,A1Y,A20,A2X,A2Y,A00,C0,CX,CY,F0,FX,FY,
     1           C44,C55,VISCSITY,PENALTY,CMAT(3,3)
      COMMON/SHP/SF(9),GDSF(2,9),SFH(16),GDSFH(2,16),GDDSFH(3,16)
      COMMON/QUAD/AL1(7,5),AL2(7,5),AL3(7,5),ALWT(7,5)
      COMMON/IO/IN,ITT
C
      DATA GAUSPT/4*0.0D0, -0.57735027D0, 0.57735027D0, 2*0.0D0,
     2     -0.77459667D0, 0.0D0, 0.77459667D0, 0.0D0, -0.86113631D0,
     3     -0.33998104D0, 0.33998104D0, 0.86113631D0/
C
      PI=4.0D0*DATAN(1.0D0)
      CONST=180.0D0/PI
      IF(IELTYP.EQ.0) THEN
C
C     Computation of the gradient/stresses at the reduced-integration
C     points of TRIANGULAR ELEMENTS:____________________________________
C
         CALL QUADRTRI (NSTR,ISTR)
C
         DO 40 NI=1,NSTR
         AC1 = AL1(NI,ISTR)
         AC2 = AL2(NI,ISTR)
         AC3 = AL3(NI,ISTR)
         CALL SHAPETRI(NPE,AC1,AC2,AC3,DET,ELXY)
         XC  = 0.0
         YC  = 0.0
         DO 10 I=1,NPE
         XC  = XC+SF(I)*ELXY(I,1)
   10    YC  = YC+SF(I)*ELXY(I,2)
         IF(ITYPE.LT.3) THEN
            UX  = 0.0
            UY  = 0.0
            VX  = 0.0
            VY  = 0.0
            DO 20 I=1,NPE
            J=NDF*I-1
            IF(ITYPE.EQ.0)J=I
            UX  = UX + ELU(J)*GDSF(1,I)
            UY  = UY + ELU(J)*GDSF(2,I)
            IF(ITYPE.GE.1) THEN
               K=J+1
               VX  = VX + ELU(K)*GDSF(1,I)
               VY  = VY + ELU(K)*GDSF(2,I)
            ENDIF
   20       CONTINUE
C
            IF(ITYPE.EQ.0) THEN
C
C     Single-degree-of-freedom problems:--------------------------------
C
               SX   = -(A10+A1X*XC+A1Y*YC)*UX
               SY   = -(A20+A2X*XC+A2Y*YC)*UY
               VALUE= DSQRT(SX**2+SY**2)
               IF(IGRAD.EQ.1) THEN
                  QX=SX
                  QY=SY
               ELSE
                  QX=-SY
                  QY= SX
               ENDIF
               IF(QX.EQ.0.0) THEN
                  IF(QY.LT.0.0) THEN
                     ANGLE =-90.0
                  ELSE
                     ANGLE = 90.0
                  ENDIF
```

```
            ELSE
               ANGLE=DATAN2(QY,QX)*CONST
            ENDIF
            WRITE(ITT,200) XC,YC,QX,QY,VALUE,ANGLE
         ELSE
C
            IF(ITYPE.EQ.1) THEN
C
C     Viscous incompressible flows (penalty model):---------------------
C
               PRESSR = -PENALTY*(UX+VY)
               STRESX = 2.0*VISCSITY*UX-PRESSR
               STRESY = 2.0*VISCSITY*VY-PRESSR
               STRSXY = VISCSITY*(UY+VX)
               WRITE(ITT,300) XC,YC,STRESX,STRESY,STRSXY,PRESSR
            ELSE
C
C     Plane elasticity problems:---------------------------------------
C
               STRESX = (CMAT(1,1)*UX+CMAT(1,2)*VY)/THKNS
               STRESY = (CMAT(1,2)*UX+CMAT(2,2)*VY)/THKNS
               STRSXY = CMAT(3,3)*(UY+VX)/THKNS
               WRITE(ITT,300) XC,YC,STRESX,STRESY,STRSXY
            ENDIF
         ENDIF
       ENDIF
   40  CONTINUE
     ELSE
C
C     Calculation of the gradient/stresses at the reduced integration
C     gauss points of RECTANGULAR ELEMENTS:_____________________________
C
       DO 100 NI=1,ISTR
       DO 100 NJ=1,ISTR
       XI  = GAUSPT(NI,ISTR)
       ETA = GAUSPT(NJ,ISTR)
       CALL SHAPERCT (NPE,XI,ETA,DET,ELXY,NDF,ITYPE)
       XC  = 0.0
       YC  = 0.0
       DO 50 I=1,NPE
       XC  = XC+SF(I)*ELXY(I,1)
   50  YC  = YC+SF(I)*ELXY(I,2)
       IF(ITYPE.LT.3) THEN
          UX  = 0.0
          UY  = 0.0
          VX  = 0.0
          VY  = 0.0
          DO 60 I=1,NPE
          J=NDF*I-1
          IF(ITYPE.EQ.0)J=I
          UX  = UX + ELU(J)*GDSF(1,I)
          UY  = UY + ELU(J)*GDSF(2,I)
          IF(ITYPE.GE.1) THEN
             K=J+1
             VX  = VX + ELU(K)*GDSF(1,I)
             VY  = VY + ELU(K)*GDSF(2,I)
          ENDIF
   60     CONTINUE
          IF(ITYPE.EQ.0) THEN
C
C     Single-degree-of-freedom problems:--------------------------------
C
             SX   = -(A10+A1X*XC+A1Y*YC)*UX
             SY   = -(A20+A2X*XC+A2Y*YC)*UY
             VALUE= DSQRT(SX**2+SY**2)
             IF(IGRAD.EQ.1) THEN
                QX=SX
                QY=SY
             ELSE
                QX=-SY
                QY= SX
             ENDIF
             IF(QX.EQ.0.0) THEN
                IF(QY.LT.0.0) THEN
                   ANGLE =-90.0
                ELSE
```

```
                  ANGLE = 90.0
               ENDIF
            ELSE
               ANGLE=DATAN2(QY,QX)*CONST
            ENDIF
            WRITE(ITT,200) XC,YC,QX,QY,VALUE,ANGLE
         ELSE
C
            IF(ITYPE.EQ.1) THEN
C
C     Viscous incompressible flows (penalty model):---------------------
C
               PRESSR = -PENALTY*(UX+VY)
               STRESX = 2.0*VISCSITY*UX-PRESSR
               STRESY = 2.0*VISCSITY*VY-PRESSR
               STRSXY = VISCSITY*(UY+VX)
               WRITE(ITT,300) XC,YC,STRESX,STRESY,STRSXY,PRESSR
            ELSE
C
C     Plane elasticity problems:---------------------------------------
C
               STRESX = (CMAT(1,1)*UX+CMAT(1,2)*VY)/THKNS
               STRESY = (CMAT(1,2)*UX+CMAT(2,2)*VY)/THKNS
               STRSXY = CMAT(3,3)*(UY+VX)/THKNS
               WRITE(ITT,300) XC,YC,STRESX,STRESY,STRSXY
            ENDIF
         ENDIF
       ELSE
C
C     Plate bending problems:------------------------------------------
C     Stresses SGMAX, SGMAY and SGMXY are computed at the top/bottom of
C     the plate (and SGMXZ and SGMYZ are constant through thickness)
C
         PLTD=(THKNS*THKNS)/6.0D0
         SIX  = 0.0
         SIY  = 0.0
         DWX  = 0.0
         DWY  = 0.0
         DSXY = 0.0
         DSYX = 0.0
         DSXX = 0.0
         DSYY = 0.0
         IF(ITYPE.EQ.3) THEN
C
C     First-order shear deformation theory of plates:-------------------
C
            DO 80 I=1,NPE
            J=NDF*(I-1)+1
            K=J+1
            L=K+1
            DWX  = DWX+GDSF(1,I)*ELU(J)
            DWY  = DWY+GDSF(2,I)*ELU(J)
            SIX  = SIX+SF(I)*ELU(K)
            SIY  = SIY+SF(I)*ELU(L)
            DSXX = DSXX+GDSF(1,I)*ELU(K)
            DSXY = DSXY+GDSF(2,I)*ELU(K)
            DSYX = DSYX+GDSF(1,I)*ELU(L)
   80       DSYY = DSYY+GDSF(2,I)*ELU(L)
            SGMAX = (CMAT(1,1)*DSXX+CMAT(1,2)*DSYY)/PLTD
            SGMAY = (CMAT(1,2)*DSXX+CMAT(2,2)*DSYY)/PLTD
            SGMXY = CMAT(3,3)*(DSXY+DSYX)/PLTD
            SGMXZ = 1.2*C55*(DWX+SIX)/THKNS
            SGMYZ = 1.2*C44*(DWY+SIY)/THKNS
            WRITE(ITT,300) XC,YC,SGMAX,SGMAY,SGMXY
            WRITE(ITT,400) SGMXZ,SGMYZ
         ELSE
C
C     Classical theory of plates:--------------------------------------
C
            NN=NPE*NDF
            DO 90 I=1,NN
            DSXX = DSXX+GDDSFH(1,I)*ELU(I)
            DSYY = DSYY+GDDSFH(2,I)*ELU(I)
   90       DSXY = DSXY+GDDSFH(3,I)*ELU(I)
C
            SGMAX =-(CMAT(1,1)*DSXX+CMAT(1,2)*DSYY)/PLTD
```

```
                SGMAY =-(CMAT(1,2)*DSXX+CMAT(2,2)*DSYY)/PLTD
                SGMXY =-4.0*CMAT(3,3)*DSXY/PLTD
                WRITE(ITT,300) XC,YC,SGMAX,SGMAY,SGMXY
             ENDIF
          ENDIF
  100     CONTINUE
      ENDIF
  200 FORMAT(5E13.4,3X,F7.2)
  300 FORMAT(6E13.4)
  400 FORMAT(26X,2E13.4)
      RETURN
      END


      SUBROUTINE QUADRTRI(NIP,IPD)
C     ________________________________________________________________
C
C     Called in ELKMFTRI to compute the quadrature points and weights
C     for triangular elements
C
C       IPD = Integrand Polynomial Degree
C       NIP = Number of Integration Points
C     ________________________________________________________________
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/QUAD/AL1(7,5),AL2(7,5),AL3(7,5),ALWT(7,5)
C
C     Initialize arrays
C
      DO 20 I = 1, NIP
      DO 10 J = 1, IPD
      AL1(I,J)   =  0.000000000000000
      AL2(I,J)   =  0.000000000000000
      AL3(I,J)   =  0.000000000000000
      ALWT(I,J)  =  0.000000000000000
   10 CONTINUE
   20 CONTINUE
C
C     One-point quadrature (for polynomials of order 1):________________
C
      AL1(1,1)   =  0.333333333333333
      AL2(1,1)   =  0.333333333333333
      AL3(1,1)   =  0.333333333333333
      ALWT(1,1)  =  1.000000000000000
C
C     Three-point quadrature (for polynomials of order 2):______________
C
      AL1(1,2)   =  0.000000000000000
      AL2(1,2)   =  0.500000000000000
      AL3(1,2)   =  0.500000000000000
      AL1(2,2)   =  0.500000000000000
      AL2(2,2)   =  0.000000000000000
      AL3(2,2)   =  0.500000000000000
      AL1(3,2)   =  0.500000000000000
      AL2(3,2)   =  0.500000000000000
      AL3(3,2)   =  0.000000000000000
      ALWT(1,2)  =  0.333333333333333
      ALWT(2,2)  =  0.333333333333333
      ALWT(3,2)  =  0.333333333333333
C
C     Four-point quadrature (for polynomials of order 3):_______________
C
      AL1(1,3)   =  0.333333333333333
      AL2(1,3)   =  0.333333333333333
      AL3(1,3)   =  0.333333333333333
      AL1(2,3)   =  0.600000000000000
      AL2(2,3)   =  0.200000000000000
      AL3(2,3)   =  0.200000000000000
      AL1(3,3)   =  0.200000000000000
      AL2(3,3)   =  0.600000000000000
      AL3(3,3)   =  0.200000000000000
      AL1(4,3)   =  0.200000000000000
      AL2(4,3)   =  0.200000000000000
      AL3(4,3)   =  0.600000000000000
      ALWT(1,3)  = -0.562500000000000
      ALWT(2,3)  =  0.520833333333333
```

```
      ALWT(3,3)  =  0.520833333333333
      ALWT(4,3)  =  0.520833333333333
C
C     Six-point quadrature (for polynomials of order 4):________________
C
      AL1(1,4)   =  0.816847572980459
      AL2(1,4)   =  0.091576213509771
      AL3(1,4)   =  0.091576213509771
      AL1(2,4)   =  0.091576213509771
      AL2(2,4)   =  0.816847572980459
      AL3(2,4)   =  0.091576213509771
      AL1(3,4)   =  0.091576213509771
      AL2(3,4)   =  0.091576213509771
      AL3(3,4)   =  0.816847572980459
      AL1(4,4)   =  0.108103018168070
      AL2(4,4)   =  0.445948490915965
      AL3(4,4)   =  0.445948490915965
      AL1(5,4)   =  0.445948490915965
      AL2(5,4)   =  0.108103018168070
      AL3(5,4)   =  0.445948490915965
      AL1(6,4)   =  0.445948490915965
      AL2(6,4)   =  0.445948490915965
      AL3(6,4)   =  0.108103018168070
      ALWT(1,4)  =  0.109951743655322
      ALWT(2,4)  =  0.109951743655322
      ALWT(3,4)  =  0.109951743655322
      ALWT(4,4)  =  0.223381589678011
      ALWT(5,4)  =  0.223381589678011
      ALWT(6,4)  =  0.223381589678011
C
C     Seven-point quadrature (for polynomials of order 5):______________
C
      AL1(1,5)   =  0.333333333333333
      AL2(1,5)   =  0.333333333333333
      AL3(1,5)   =  0.333333333333333
      AL1(2,5)   =  0.797426985353087
      AL2(2,5)   =  0.101286507323456
      AL3(2,5)   =  0.101286507323456
      AL1(3,5)   =  0.101286507323456
      AL2(3,5)   =  0.797426985353087
      AL3(3,5)   =  0.101286507323456
      AL1(4,5)   =  0.101286507323456
      AL2(4,5)   =  0.101286507323456
      AL3(4,5)   =  0.797426985353087
      AL1(5,5)   =  0.059715871789770
      AL2(5,5)   =  0.470142064105115
      AL3(5,5)   =  0.470142064105115
      AL1(6,5)   =  0.470142064105115
      AL2(6,5)   =  0.059715871789770
      AL3(6,5)   =  0.470142064105115
      AL1(7,5)   =  0.470142064105115
      AL2(7,5)   =  0.470142064105115
      AL3(7,5)   =  0.059715871789770
      ALWT(1,5)  =  0.225000000000000
      ALWT(2,5)  =  0.125939180544827
      ALWT(3,5)  =  0.125939180544827
      ALWT(4,5)  =  0.125939180544827
      ALWT(5,5)  =  0.132394152788506
      ALWT(6,5)  =  0.132394152788506
      ALWT(7,5)  =  0.132394152788506
C
      RETURN
      END

      SUBROUTINE SHAPERCT(NPE,XI,ETA,DET,ELXY,NDF,ITYPE)
C     ________________________________________________________________
C
C     Called in SHAPERCT to evaluate the interpolation functions SF(I)
C     and the derivatives with respect to global coordinates GDSF(I,J)
C     for Lagrange linear & quadratic rectangular elements, using  the
C     isoparametric formulation. The subroutine also evaluates Hermite
C     interpolation functions and their  global derivatives  using the
C     subparametric formulation.
C
C     SF(I)........Interpolation function for node I of the element
```

```
C     DSF(J,I).....Derivative of SF(I) with respect to XI if J=1 and
C                  and ETA if J=2
C     GDSF(J,I)....Derivative of SF(I) with respect to  X if J=1 and
C                  and  Y  if J=2
C     XNODE(I,J)...J-TH (J=1,2) Coordinate of node I of the element
C     NP(I)........Array of element nodes (used to define SF and DSF)
C     GJ(I,J)....…Determinant of the Jacobian matrix
C     GJINV(I,J)...Inverse of the jacobian matrix
C     ________________________________________________________________
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ELXY(9,2),XNODE(9,2),NP(9),DSF(2,9),GJ(2,2),GJINV(2,2)
      DIMENSION GGJ(3,3),GGINV(3,3),DDSJ(3,16),DDSF(3,4),DJCB(3,2),
     *          DSFH(3,16),DDSFH(3,16)
      COMMON/SHP/SF(9),GDSF(2,9),SFH(16),GDSFH(2,16),GDDSFH(3,16)
      COMMON/IO/IN,ITT
      DATA XNODE/-1.0D0, 2*1.0D0, -1.0D0, 0.0D0, 1.0D0, 0.0D0, -1.0D0,
     *        0.0D0, 2*-1.0D0, 2*1.0D0, -1.0D0, 0.0D0, 1.0D0, 2*0.0D0/
      DATA NP/1,2,3,4,5,7,6,8,9/
C
      FNC(A,B) = A*B
      IF(NPE.EQ.4) THEN
C
C     LINEAR Lagrange interpolation functions for FOUR-NODE element
C
         DO 10 I = 1, NPE
         XP  = XNODE(I,1)
         YP  = XNODE(I,2)
         XI0 = 1.0+XI*XP
         ETA0=1.0+ETA*YP
         SF(I)   = 0.25*FNC(XI0,ETA0)
         DSF(1,I)= 0.25*FNC(XP,ETA0)
   10    DSF(2,I)= 0.25*FNC(YP,XI0)
      ELSE
         IF(NPE.EQ.8) THEN
C
C     QUADRATIC Lagrange interpolation functions for EIGHT-NODE element
C
            DO 20 I = 1, NPE
            NI   = NP(I)
            XP   = XNODE(NI,1)
            YP   = XNODE(NI,2)
            XI0  = 1.0+XI*XP
            ETA0 = 1.0+ETA*YP
            XI1  = 1.0-XI*XI
            ETA1 = 1.0-ETA*ETA
            IF(I.LE.4) THEN
               SF(NI)    = 0.25*FNC(XI0,ETA0)*(XI*XP+ETA*YP-1.0)
               DSF(1,NI) = 0.25*FNC(ETA0,XP)*(2.0*XI*XP+ETA*YP)
               DSF(2,NI) = 0.25*FNC(XI0,YP)*(2.0*ETA*YP+XI*XP)
            ELSE
               IF(I.LE.6) THEN
                  SF(NI)    = 0.5*FNC(XI1,ETA0)
                  DSF(1,NI) = -FNC(XI,ETA0)
                  DSF(2,NI) = 0.5*FNC(YP,XI1)
               ELSE
                  SF(NI)    = 0.5*FNC(ETA1,XI0)
                  DSF(1,NI) = 0.5*FNC(XP,ETA1)
                  DSF(2,NI) = -FNC(ETA,XI0)
               ENDIF
            ENDIF
   20       CONTINUE
         ELSE
C
C     QUADRATIC Lagrange interpolation functions for NINE-NODE element
C
            DO 30 I=1,NPE
            NI   = NP(I)
            XP   = XNODE(NI,1)
            YP   = XNODE(NI,2)
            XI0  = 1.0+XI*XP
            ETA0 = 1.0+ETA*YP
            XI1  = 1.0-XI*XI
            ETA1 = 1.0-ETA*ETA
            XI2  = XP*XI
            ETA2 = YP*ETA
```

```
            IF(I .LE. 4) THEN
               SF(NI)   = 0.25*FNC(XI0,ETA0)*XI2*ETA2
               DSF(1,NI)= 0.25*XP*FNC(ETA2,ETA0)*(1.0+2.0*XI2)
               DSF(2,NI)= 0.25*YP*FNC(XI2,XI0)*(1.0+2.0*ETA2)
            ELSE
               IF(I .LE. 6) THEN
                  SF(NI)    = 0.5*FNC(XI1,ETA0)*ETA2
                  DSF(1,NI) = -XI*FNC(ETA2,ETA0)
                  DSF(2,NI) = 0.5*FNC(XI1,YP)*(1.0+2.0*ETA2)
               ELSE
                  IF(I .LE. 8) THEN
                     SF(NI)    = 0.5*FNC(ETA1,XI0)*XI2
                     DSF(2,NI) = -ETA*FNC(XI2,XI0)
                     DSF(1,NI) = 0.5*FNC(ETA1,XP)*(1.0+2.0*XI2)
                  ELSE
                     SF(NI)    = FNC(XI1,ETA1)
                     DSF(1,NI) = -2.0*XI*ETA1
                     DSF(2,NI) = -2.0*ETA*XI1
                  ENDIF
               ENDIF
            ENDIF
   30       CONTINUE
         ENDIF
      ENDIF
C
C     Compute the Jacobian matrix [GJ] and its inverse [GJINV]
C
      DO 40 I = 1,2
      DO 40 J = 1,2
      GJ(I,J)  = 0.0
      DO 40 K = 1,NPE
   40 GJ(I,J)  = GJ(I,J) + DSF(I,K)*ELXY(K,J)
C
      DET = GJ(1,1)*GJ(2,2)-GJ(1,2)*GJ(2,1)
      GJINV(1,1) = GJ(2,2)/DET
      GJINV(2,2) = GJ(1,1)/DET
      GJINV(1,2) = -GJ(1,2)/DET
      GJINV(2,1) = -GJ(2,1)/DET
C
      IF(ITYPE.LE.3) THEN
C
C     Compute the derivatives of the interpolation functions with
C     respect to the global coordinates (x,y): [GDSF]
C
         DO 50 I  = 1,2
         DO 50 J  = 1,NPE
         GDSF(I,J) = 0.0
         DO 50 K  = 1, 2
   50    GDSF(I,J) = GDSF(I,J) + GJINV(I,K)*DSF(K,J)
      ELSE
C
C     Conforming Hermite interpolation functions (four-node element)
C
         IF(NDF.EQ.4) THEN
            II = 1
            DO 60 I = 1, NPE
            XP   = XNODE(I,1)
            YP   = XNODE(I,2)
            XI1  = XI*XP-1.0
            XI2  = XI1-1.0
            ETA1 = ETA*YP-1.0
            ETA2 = ETA1-1.0
            XI0  = (XI+XP)*(XI+XP)
            ETA0 = (ETA+YP)*(ETA+YP)
            XIP0 = XI+XP
            XIP1 = 3.0*XI*XP+XP*XP
            XIP2 = 3.0*XI*XP+2.0*XP*XP
            YIP0 = ETA+YP
            YIP1 = 3.0*ETA*YP+YP*YP
            YIP2 = 3.0*ETA*YP+2.0*YP*YP
C
            SFH(II)        = 0.0625*FNC(ETA0,ETA2)*FNC(XI0,XI2)
            DSFH(1,II)     = 0.0625*FNC(ETA0,ETA2)*XIP0*(XIP1-4.0)
            DSFH(2,II)     = 0.0625*FNC(XI0,XI2)*YIP0*(YIP1-4.0)
            DDSFH(1,II)    = 0.125*FNC(ETA0,ETA2)*(XIP2-2.0)
            DDSFH(2,II)    = 0.125*FNC(XI0,XI2)*(YIP2-2.0)
```

```
            DDSFH(3,II)   = 0.0625*(XIP1-4.0)*(YIP1-4.0)*XIP0*YIP0
C
            SFH(II+1)     = -0.0625*XP*FNC(XI0,XI1)*FNC(ETA0,ETA2)
            DSFH(1,II+1)  = -0.0625*FNC(ETA0,ETA2)*XP*XIP0*(XIP1-2.0)
            DSFH(2,II+1)  = -0.0625*FNC(XI0,XI1)*XP*YIP0*(YIP1-4.)
            DDSFH(1,II+1) = -0.125*FNC(ETA0,ETA2)*XP*(XIP2-1.0)
            DDSFH(2,II+1) = -0.125*FNC(XI0,XI1)*(YIP2-2.0)*XP
            DDSFH(3,II+1) = -0.0625*XP*XIP0*(XIP1-2.)*(YIP1-4.)*YIP0
C
            SFH(II+2)     = -0.0625*YP*FNC(XI0,XI2)*FNC(ETA0,ETA1)
            DSFH(1,II+2)  = -0.0625*FNC(ETA0,ETA1)*YP*XIP0*(XIP1-4.)
            DSFH(2,II+2)  = -0.0625*FNC(XI0,XI2)*YP*YIP0*(YIP1-2.)
            DDSFH(1,II+2) = -0.125*FNC(ETA0,ETA1)*YP*(XIP2-2.)
            DDSFH(2,II+2) = -0.125*FNC(XI0,XI2)*YP*(YIP2-1.0)
            DDSFH(3,II+2) = -0.0625*YP*YIP0*(YIP1-2.)*(XIP1-4.0)*XIP0
C
            SFH(II+3)     = 0.0625*XP*YP*FNC(XI0,XI1)*FNC(ETA0,ETA1)
            DSFH(1,II+3)  = 0.0625*FNC(ETA0,ETA1)*XP*YP*(XIP1-2.)*XIP0
            DSFH(2,II+3)  = 0.0625*FNC(XI0,XI1)*XP*YP*(YIP1-2.)*YIP0
            DDSFH(1,II+3) = 0.125*FNC(ETA0,ETA1)*XP*YP*(XIP2-1.)
            DDSFH(2,II+3) = 0.125*FNC(XI0,XI1)*XP*YP*(YIP2-1.0)
            DDSFH(3,II+3) = 0.0625*XP*YP*YIP0*XIP0*(YIP1-2.)*(XIP1-2.)
            II = I*NDF + 1
   60       CONTINUE
         ELSE
C
C     Non-conforming Hermite interpolation functions (Four-node element)
C
            II = 1
            DO 80 I = 1, NPE
            XP    = XNODE(I,1)
            YP    = XNODE(I,2)
            XI0   = XI*XP
            ETA0  = ETA*YP
            XIP1  = XI0+1
            ETAP1 = ETA0+1
            XIM1  = XI0-1
            ETAM1 = ETA0-1
            XID   = 3.0+2.0*XI0+ETA0-3.0*XI*XI-ETA*ETA-2.0*XI/XP
            ETAD  = 3.0+XI0+2.0*ETA0-XI*XI-3.0*ETA*ETA-2.0*ETA/YP
            ETAXI = 4.0+2.0*(XI0+ETA0)-3.0*(XI*XI+ETA*ETA)
     *                                -2.0*(ETA/YP+XI/XP)
C
            SFH(II) = 0.125*XIP1*ETAP1*(2.0+XI0+ETA0-XI*XI-ETA*ETA)
            DSFH(1,II)    = 0.125*XP*ETAP1*XID
            DSFH(2,II)    = 0.125*YP*XIP1*ETAD
            DDSFH(1,II)   = 0.250*XP*ETAP1*(XP-3.0*XI-1.0/XP)
            DDSFH(2,II)   = 0.250*YP*XIP1*(YP-3.0*ETA-1.0/YP)
            DDSFH(3,II)   = 0.125*XP*YP*ETAXI
C
            SFH(II+1)     = 0.125*XP*XIP1*XIP1*XIM1*ETAP1
            DSFH(1,II+1)  = 0.125*XP*XP*ETAP1*(3.0*XI0-1.0)*XIP1
            DSFH(2,II+1)  = 0.125*XP*YP*XIP1*XIP1*XIM1
            DDSFH(1,II+1) = 0.250*XP*XP*XP*ETAP1*(3.0*XI0+1.0)
            DDSFH(2,II+1) = 0.0
            DDSFH(3,II+1) = 0.125*XP*XP*YP*(3.0*XI0-1.0)*XIP1
C
            SFH(II+2)     = 0.125*YP*XIP1*ETAP1*ETAP1*ETAM1
            DSFH(1,II+2)  = 0.125*XP*YP*ETAP1*ETAP1*ETAM1
            DSFH(2,II+2)  = 0.125*YP*YP*XIP1*(3.0*ETA0-1.0)*ETAP1
            DDSFH(1,II+2) = 0.0
            DDSFH(2,II+2) = 0.250*YP*YP*YP*XIP1*(3.0*ETA0+1.0)
            DDSFH(3,II+2) = 0.125*XP*YP*YP*(3.0*ETA0-1.0)*ETAP1
            II = I*NDF + 1
   80       CONTINUE
         ENDIF
C
C     Compute the global first and second derivatives of the Hermite
C     interpolation functions.  The geometry is approximated using the
C     linear Lagrane interpolation functions (Subparametric formulation)
C
         DDSF(1,1) =   0.0D0
         DDSF(2,1) =   0.0D0
         DDSF(3,1) =   0.250D0
         DDSF(1,2) =   0.0D0
         DDSF(2,2) =   0.0D0
```

```
         DDSF(3,2) = - 0.250D0
         DDSF(1,3) =   0.0D0
         DDSF(2,3) =   0.0D0
         DDSF(3,3) =   0.250D0
         DDSF(1,4) =   0.0D0
         DDSF(2,4) =   0.0D0
         DDSF(3,4) = - 0.250D0
C
C     Compute global first derivatives of Hermite functions
C
         NN=NDF*NPE
         DO 110 I = 1, 2
         DO 100 J = 1, NN
         SUM = 0.0D0
         DO 90 K = 1, 2
         SUM = SUM + GJINV(I,K)*DSFH(K,J)
   90    CONTINUE
         GDSFH(I,J) = SUM
  100    CONTINUE
  110    CONTINUE
C
C     Compute global second derivatives of Hermite functions
C
         DO 140 I = 1, 3
         DO 130 J = 1, 2
         SUM = 0.0D0
         DO 120 K = 1, NPE
         SUM = SUM + DDSF(I,K)*ELXY(K,J)
  120    CONTINUE
         DJCB(I,J) = SUM
  130    CONTINUE
  140    CONTINUE
C
         DO 170 K = 1, 3
         DO 160 J = 1, NN
         SUM = 0.0D0
         DO 150 L = 1, 2
         SUM = SUM + DJCB(K,L)*GDSFH(L,J)
  150    CONTINUE
         DDSJ(K,J) = SUM
  160    CONTINUE
  170    CONTINUE
C
C     Compute the jacobian of the transformation
C
         GGJ(1,1)=GJ(1,1)*GJ(1,1)
         GGJ(1,2)=GJ(1,2)*GJ(1,2)
         GGJ(1,3)=2.0*GJ(1,1)*GJ(1,2)
         GGJ(2,1)=GJ(2,1)*GJ(2,1)
         GGJ(2,2)=GJ(2,2)*GJ(2,2)
         GGJ(2,3)=2.0*GJ(2,1)*GJ(2,2)
         GGJ(3,1)=GJ(2,1)*GJ(1,1)
         GGJ(3,2)=GJ(2,2)*GJ(1,2)
         GGJ(3,3)=GJ(2,1)*GJ(1,2)+GJ(1,1)*GJ(2,2)
         CALL INVERSE(GGJ,GGINV)
C
         DO 200 I = 1, 3
         DO 190 J = 1, NN
         SUM = 0.0D0
         DO 180 K = 1, 3
         SUM = SUM + GGINV(I,K)*(DDSFH(K,J)-DDSJ(K,J))
  180    CONTINUE
         GDDSFH(I,J) = SUM
  190    CONTINUE
  200    CONTINUE
      ENDIF
      RETURN
      END


      SUBROUTINE SHAPETRI(NPE,AL1,AL2,AL3,DET,ELXY)
C     ________________________________________________________________
C
C     Called in  ELKMFTRI  to evaluate the interpolation functions and
C     their global derivatives at the quadrature points for the linear
C     and quadratic (i.e., 3-node and 6-node) triangular elements.
```

```
C     ________________________________________________________________
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/SHP/SF(9),GDSF(2,9),SFH(16),GDSFH(2,16),GDDSFH(3,16)
      DIMENSION DSF(3,9),ELXY(9,2),GJ(2,2),GJINV(2,2)
C
C     Initialize the arrays
C
      DO 10 I = 1, NPE
      DSF(1,I) = 0.0D0
      DSF(2,I) = 0.0D0
      DSF(3,I) = 0.0D0
   10 CONTINUE
C
      IF(NPE.EQ.3) THEN
C
C     Linear Lagrane interpolation for three-node element
C
         SF(1) =  AL1
         SF(2) =  AL2
         SF(3) =  AL3
         DSF(1,1) = 1.0D0
         DSF(2,2) = 1.0D0
         DSF(3,3) = 1.0D0
      ELSE
C
C     Quadratic Lagrange interpolation functions for six-nde element
C
         SF(1) = AL1 * (2.0D0 * AL1 - 1)
         SF(2) = AL2 * (2.0D0 * AL2 - 1)
         SF(3) = AL3 * (2.0D0 * AL3 - 1)
         SF(4) = 4.0D0 * AL1 * AL2
         SF(5) = 4.0D0 * AL2 * AL3
         SF(6) = 4.0D0 * AL3 * AL1
         DSF(1,1) = 4.0D0 * AL1 - 1
         DSF(2,2) = 4.0D0 * AL2 - 1
         DSF(3,3) = 4.0D0 * AL3 - 1
         DSF(1,4) = 4.0D0 * AL2
         DSF(2,4) = 4.0D0 * AL1
         DSF(2,5) = 4.0D0 * AL3
         DSF(3,5) = 4.0D0 * AL2
         DSF(1,6) = 4.0D0 * AL3
         DSF(3,6) = 4.0D0 * AL1
      ENDIF
C
C     Compute the global derivatives of SF(I).  Note that the special
C     form of the jacobian for area coordinates,  AL3 = 1-AL1-AL2  is
C     substituted
C
      DO 60 I = 1,2
      DO 50 J = 1,2
      SUM = 0.0D0
      DO 40 K = 1, NPE
      SUM = SUM + (DSF(I,K) - DSF(3,K))*ELXY(K,J)
   40 CONTINUE
      GJ(I,J) = SUM
   50 CONTINUE
   60 CONTINUE
C
      DET = GJ(1,1)*GJ(2,2) - GJ(1,2)*GJ(2,1)
      GJINV(1,1) = GJ(2,2)/DET
      GJINV(2,2) = GJ(1,1)/DET
      GJINV(1,2) = -GJ(1,2)/DET
      GJINV(2,1) = -GJ(2,1)/DET
      DO 100 I = 1, 2
      DO 90 J = 1, NPE
      SUM = 0.0D0
      DO 80 K = 1, 2
      SUM = SUM + GJINV(I,K) * (DSF(K,J) - DSF(3,J))
   80 CONTINUE
      GDSF(I,J) = SUM
   90 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EQNSOLVR(NRM,NCM,NEQNS,NBW,BAND,RHS,IRES)
C     ________________________________________________________________
C
C     Called in MAIN to solve a  banded, symmetric, system of algebraic
C     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 TEMPORAL(NCOUNT,INTIAL,ITEM,NN)
C   ________________________________________________________________
C
C    Called in MAIN to compute the fully discretized equations for the
C    parabolic and hyperbolic differential equations in time using the
C    alfa-family and Newmark family of approximations, respectively.
C   ________________________________________________________________
```

```
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/STF/ELF(27),ELK(27,27),ELM(27,27),ELXY(9,2),ELU(27),
     1           ELV(27),ELA(27),A1,A2,A3,A4,A5
C
      IF(ITEM.EQ.1) THEN
C
C     The alfa-family of time approximation for parabolic equations
C
         DO 20 I=1,NN
         SUM=0.0
         DO 10 J=1,NN
         SUM=SUM+(ELM(I,J)-A2*ELK(I,J))*ELU(J)
   10    ELK(I,J)=ELM(I,J)+A1*ELK(I,J)
   20    ELF(I)=(A1+A2)*ELF(I)+SUM
      ELSE
C
C     The Newmark integration scheme for hyperbolic equations
C
         IF(NCOUNT.EQ.1 .AND. INTIAL.NE.0) THEN
            DO 40 I = 1,NN
            ELF(I)  = 0.0
            DO 40 J = 1,NN
            ELF(I)  = ELF(I)-ELK(I,J)*ELU(J)
   40       ELK(I,J)= ELM(I,J)
         ELSE
            DO 70 I = 1,NN
            SUM     = 0.0
            DO 60 J = 1,NN
            SUM     = SUM+ELM(I,J)*(A3*ELU(J)+A4*ELV(J)+A5*ELA(J))
   60       ELK(I,J)= ELK(I,J)+A3*ELM(I,J)
   70       ELF(I)  = ELF(I)+SUM
         ENDIF
      ENDIF
      RETURN
      END
```