Appendix E

Fortran Code of the Finite Element Program Containing the Concrete Plasticity Model

```
!**************************************************************************
!
!  PROGRAM: Column_Plasticity
!
!  PURPOSE:  Calculate stress vs. strain and moment-curvature curves for
!            a given column cross section and given axial load range.
!
!  NOTES:  Elements must always be numbered so that the top layer nodes
!          are first and the axis of the column runs through x=y=0.
!          Present code does not allow for load reversals as the concrete
!          model does not account for crack opening/closing.
!
!  WRITTEN BY: Julie Wolf (includes sections by John Hall)
!
!**************************************************************************

      PROGRAM MAIN
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      DIMENSION TITLE(20)
      REAL(KIND=16), ALLOCATABLE, DIMENSION(:) :: A,B,C
      REAL(KIND=16), ALLOCATABLE, DIMENSION(:,:) :: AXLDS
C
C Array AXLDS contains the vector of axial load and moment values
C Array A contains mesh data (COOR, ID, and R)
C Array B contains element data
C Array C contains solution data (H, F, RTPT, DX, XT, XTPT, ELRT ELRTPT)
C
      INTEGER(KIND=8), ALLOCATABLE, DIMENSION(:) :: NTYPE
C
C Array NTYPE designates the plane of the node
C      0 - bottom plane
C      1 - top plane
C Used for determining degrees of freedom
C
      REAL(KIND=16) MX,MY,MSTOP,MMAX
      COMMON /INFO/ N5TEMP
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NPAR(8)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      COMMON /TOL/ TOLF,TOLM,NITMAX
      COMMON /DEBUG/ IDEBUG
      OPEN (5,FILE='input.txt')
      OPEN (6,FILE='output.txt')
```

```
      OPEN (7,FILE='storage',FORM='UNFORMATTED')
      OPEN (8,FILE='MC_curves.txt')
C
C Read and print control information.
C
      READ(5,1000) TITLE
 1000 FORMAT(20A4)
      READ(5,*) NNP,NEG1,NEG10,NEG11,NEG100,NEG110,NUMAX
      ALLOCATE(NTYPE(NNP))
      ALLOCATE(AXLDS(NUMAX,2))
      AXLDS=0.Q0
      NTYPE=0
      DO 50 K=1,NUMAX
      READ(5,*) AXLDS(K,1),AXLDS(K,2)
   50 CONTINUE
      WRITE(6,1001) TITLE,NNP,NEG1,NEG10,NEG11,NEG100,NEG110,NUMAX
 1001 FORMAT(5(/),1X,20A4,4(/),1X,'NNP =',I5,/,1X,
     *    'NEG1 (Longitudinal Rebar) =',I5,(/),1X,
     *    'NEG10 (Lower Transverse Rebar) =',I5,(/),1X,
     *    'NEG11 (Upper Transverse Rebar) =',I5,(/),1X
     *    'NEG100 (Top Layer Concrete) =',I5,(/),1X,
     *    'NEG110 (Interior Concrete) =',I5,(/),1X,
     *    'NUMAX =',I5,(/),1X,'AXLDS =')
      DO 100 I=1,NUMAX
      WRITE(6,1002) AXLDS(I,1),AXLDS(I,2)
 1002 FORMAT(10X,E12.5,4X,E12.5)
  100 CONTINUE
      READ(5,*) NSSTEPS,NHSTEPS,NITMAX,ANGLE,TOLF,TOLM,MSTOP,MMAX
      WRITE(6,1003) NSSTEPS,NHSTEPS,NITMAX,ANGLE,TOLF,TOLM,MSTOP,MMAX
 1003 FORMAT(2(/),1X,'NSSTEPS =',I5,/,1X,'NHSTEPS =',I5,/,1X,
     *   'NITMAX = ',I5,/,1X,
     *   'ANGLE (IN DEGREES) OF MOMENT APPLICATION = ',E12.5,/,1X,
     *   'FORCE TOLERANCE = ',E12.5,/,1X,'MOMENT TOLERANCE = ',E12.5,/,
     *   1X,'SMALLEST MOMENT INCREMENT ATTEMPTED = ',E12.5,/,1X,
     *   'MAXIMUM MOMENT ATTEMPTED = ',E12.5)
      IF (NHSTEPS .GT. NSSTEPS .AND. NUMAX .EQ. 1
     *  .AND. AXLDS(1,2) .EQ. 0.) THEN
          WRITE(6,*) 'Number of hydrostatic steps must be less than',
     *      ' or equal to the number of static steps'
          STOP
      ENDIF
C
C Read and print nodal data.  Number equations.
      N1=1
      N2=N1+NNP*3 ! COOR
      N3=N2+NNP*3 ! ID
      N4=N3+NNP*3 ! R
      ALLOCATE(A(N4-1))
      A=0.Q0
      ZCOR=0.
C
C Generates and prints nodal coordinates.
      CALL READC(A(N1),NNP,ZCOR,NTYPE)
C
C Generates the nodal boundary conditions.  Numbers the equations and
C prints the equation numbers.  Computes NEQ.
      CALL READID(A(N2),A(N1),NNP,NEQ,NTYPE)
C
C Generates and prints nodal loads and specified displacements.
      CALL READR(A(N3),A(N2),NNP)
C
C Read and print element data.  Calculate assembly arrays.  Calculate
C and print half-bandwidth.
C
      MBAND=1
      MAXNEL1=0
      MAXNEL10=0
      MAXNEL11=0
      MAXNEL100=0
      MAXNEL110=0
      MAXNINT=1
C
      N5=1
      N6=1
      N7=1
```

```
      N8=1
      N9=1
      N10=1
      N11=1
      N1101=1
      N1102=1
      N1103=1
      N1104=1
      N12=1
      N1201=1
      N1202=1
      N1203=1
      N1204=1
      N13=NEQ*(8*6*2+1)+1
      WRITE(6,2001)
 2001 FORMAT(4(/),1X,'ELEMENT DATA')
      IGO=1
      REWIND 7
      DO 200 IEG=1,NEG1
      READ(5,*) NPAR
      IF(NPAR(1).NE.1) THEN
        WRITE(6,2002)
 2002   FORMAT(2(/),'Incorrect number of longitudinal rebar groups',
     *  ' (Type 1 elements)')
        STOP
      END IF
      CALL TRUSSA(A,B,C)
      IF(N5TEMP.GT.N5) N5=N5TEMP
  200 CONTINUE
      DO 201 IEG=1,NEG10
      READ(5,*) NPAR
      IF(NPAR(1).NE.10) THEN
        WRITE(6,2003)
 2003   FORMAT(2(/),'Incorrect number of lower transverse rebar groups',
     *      ' (Type 10 elements)')
        STOP
      ENDIF
      CALL TRUSSB(A,B,C)
      IF(N5TEMP.GT.N5) N5=N5TEMP
  201 CONTINUE
      DO 202 IEG=1,NEG11
      READ(5,*) NPAR
      IF(NPAR(1).NE.11) THEN
        WRITE(6,2004)
 2004   FORMAT(2(/),'Incorrect number of upper transverse rebar groups',
     *      ' (Type 11 elements)')
        STOP
      ENDIF
      CALL TRUSSC(A,B,C)
      IF(N5TEMP.GT.N5) N5=N5TEMP
  202 CONTINUE
      DO 203 IEG=1,NEG100
      READ(5,*) NPAR
      IF(NPAR(1).NE.100) THEN
        WRITE(6,2005)
 2005   FORMAT(2(/),'Incorrect number of solid concrete groups',
     *      ' (Type 100 elements)')
        STOP
      END IF
      CALL SOLIDA(A,B,C)
      IF(N5TEMP.GT.N5) N5=N5TEMP
  203 CONTINUE
      DO 204 IEG=1,NEG110
      READ(5,*) NPAR
      IF(NPAR(1).NE.110) THEN
        WRITE(6,2006)
 2006   FORMAT(2(/),'Incorrect number of solid concrete groups',
     *      'Type 110 elements)')
        STOP
      END IF
      CALL SOLIDB(A,B,C)
      IF(N5TEMP.GT.N5) N5=N5TEMP
  204 CONTINUE
      WRITE(6,2007) MBAND,NEQ
 2007 FORMAT(4(/),1X,'MBAND = ',I5,/,1X,'NEQ = ',I5)
```

```
C
C Assemble nodal loads.  Calculate and assemble element matrices and
C   vectors.
      N6=1+NEQ*MBAND                                          ! H
      N7=N6+NEQ                                               ! F
      N8=N7+NEQ                                               ! RTPT
      N9=N8+NEQ                                               ! DX
      N10=N9+NEQ                                              ! XT
      N11=N10+NEQ                                             ! XTPT
      N1101=N11+4*MAXNEL1*NEG1                                ! ELRT1
      N1102=N1101+4*MAXNEL10*NEG10                            ! ELRT10
      N1103=N1102+4*MAXNEL11*NEG11                            ! ELRT11
      N1104=N1103+(MAXNINT*(6*2+3)+1)*MAXNEL100*NEG100  ! ELRT100
      N12=N1104+(MAXNINT*(6*2+3)+1)*MAXNEL110*NEG110    ! ELRT110
      N1201=N12+4*MAXNEL1*NEG1                                ! ELRTPT1
      N1202=N1201+4*MAXNEL10*NEG10                            ! ELRTPT10
      N1203=N1202+4*MAXNEL11*NEG11                            ! ELRTPT11
      N1204=N1203+(MAXNINT*(6*2+3)+1)*MAXNEL100*NEG100  ! ELRTPT100
      N13=N1204+(MAXNINT*(6*2+3)+1)*MAXNEL110*NEG110    ! ELRTPT110
      ALLOCATE(B(N5-1))
      ALLOCATE(C(N13-1))
C
C
C Loop over all axial load cases
      DO 300 K=1,NUMAX
      B=0.Q0
      C=0.Q0
      MX=0.Q0
      MY=0.Q0
      DM=1000.Q0*TOLM
C
C Transfer nodal loads from R to F.
      CALL FILLF(A(N2),A(N3),C(N6),AXLDS(K,1),MX,MY,NNP,NEQ)
C
C Transfer initial element lengths to ELRTPT
      IGO=2
      REWIND 7
      DO 400 IEG=1,NEG1
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSA(A,B,C)
  400 CONTINUE
      DO 401 IEG=1,NEG10
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSB(A,B,C)
  401 CONTINUE
      DO 402 IEG=1,NEG11
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSC(A,B,C)
  402 CONTINUE
      DO 403 IEG=1,NEG100
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDA(A,B,C)
  403 CONTINUE
      DO 404 IEG=1,NEG110
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDB(A,B,C)
  404 CONTINUE
      WRITE(6,3000) AXLDS(K,1)
      WRITE(*,3000) AXLDS(K,1)
 3000 FORMAT(4(/),1X,'SOLUTION INFORMATION FOR AXIAL LOAD = ',E12.5)
      WRITE(8,3001) AXLDS(K,1)
 3001 FORMAT(/,1X,'AXIAL LOAD =',E19.12,/)
      WRITE(8,3002)
 3002 FORMAT(10x,'CURVATURE-X',18X,'MX',9X,'CURVATURE-Y',18X,'MY')
C
C Solve for axial load only
      CALL STANAL(A(N1),A(N2),A(N3),B,C(1),C(N6),C(N7),C(N8),C(N9),
     *   C(N10),C(N11),C(N12),AXLDS(K,1),NTYPE,NSSTEPS,NHSTEPS,ZCOR,
     *   NITER)
      TEMP=QSQRT(C(N11-2)*C(N11-2)+C(N11-1)*C(N11-1))
      WRITE(8,3003) C(N11-2),0.,C(N11-1),0.
 3003 FORMAT(1X,6(1X,E19.12))
c Skip moment curvature calcs if no applied moment
      IF (AXLDS(K,2) .EQ. 0.) GO TO 300
C
C Create moment-curvature curve
```

```
      WRITE(6,3004) AXLDS(K,2)
      WRITE(*,3004) AXLDS(K,2)
 3004 FORMAT(4(/),1X,'SOLUTION INFORMATION FOR MOMENT = ',E12.5)
      MX=AXLDS(K,2)*QCOSD(ANGLE)
      MY=AXLDS(K,2)*QSIND(ANGLE)
      CALL FILLF(A(N2),A(N3),C(N6),AXLDS(K,1),MX,MY,NNP,NEQ)
      CALL MCANAL(A(N1),A(N2),A(N3),B,C(1),C(N6),C(N7),C(N8),C(N9),
     *   C(N10),C(N11),C(N12),AXLDS(K,2),NTYPE,NSSTEPS,NHSTEPS,ZCOR,
     *   NITER)
  300 CONTINUE
      STOP
      END PROGRAM MAIN

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

      SUBROUTINE ASSMBL(H,F,HE,FE,IDL,NEQ,MBAND,NED,M1,M2)
C Assembles HE into H when M1.NE.0.  Assembles FE into F when M2.NE.0.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION H(NEQ,MBAND),F(NEQ),HE(NED,NED),FE(NED*M2),IDL(NED)
      IF(M1.EQ.0) GO TO 2
C Assemble HE into H.
      DO 10 I=1,NED
      IROW=IDL(I)
      IF(IROW.EQ.0) GO TO 10
      DO 20 J=1,NED
      ICOL=IDL(J)-IROW+1
      IF(ICOL.GT.0) H(IROW,ICOL)=H(IROW,ICOL)+HE(I,J)
   20 CONTINUE
   10 CONTINUE
    2 IF(M2.EQ.0) RETURN
C Assemble FE into F.
      DO 30 I=1,NED
      IROW=IDL(I)
      IF(IROW.EQ.0) GO TO 30
      F(IROW)=F(IROW)+FE(I)
   30 CONTINUE
      RETURN
      END

      SUBROUTINE BAND(IDL,NED,NEL,MBAND)
C Computes MBAND.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION IDL(NED,NEL)
      DO 10  N=1,NEL
      DO 20 I=1,NED
      IROW=IDL(I,N)
      IF(IROW.EQ.0) GO TO 20
      DO 30 J=1,NED
      ICOL=IDL(J,N)
      IF(ICOL.EQ.0) GO TO 30
      NDIFF=IROW-ICOL+1
      IF(NDIFF.GT.MBAND) MBAND=NDIFF
   30 CONTINUE
   20 CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE BSOLVE(A,B,N,NBD,MA,MB)
C symmetric banded equation solver for AX=B where A is N*NBD and B is N*1
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION A(N,NBD),B(N)
      IF(MA.EQ.0) GO TO 200
C forward reduction on A
      AVOD=0.Q0
      DO 100 I=1,N
      AVOD=AVOD+QABS(A(I,1))
  100 CONTINUE
      AVOD=AVOD/N
```

```
      TOL=AVOD* 1.Q-12
      DMIN=1.Q+30
      DO 110 IROW=1,N
c add below line to prevent negative stiffness
      IF(A(IROW,1).LT.DMIN) DMIN=A(IROW,1)
      IF(DMIN.LT.TOL) GO TO 300
      IF(IROW.EQ.N) GO TO 110
      MIN=KMIN0(NBD,N-IROW+1)
      DO 120 J=2,MIN
      IF (A(IROW,J).EQ.0.Q0) GO TO 120
      ST=A(IROW,J)/A(IROW,1)
      IJ1=IROW+J-1
      DO 130 K=J,MIN
      KJ1=K-J+1
      A(IJ1,KJ1)=A(IJ1,KJ1)-ST*A(IROW,K)
  130 CONTINUE
      A(IROW,J)=ST
  120 CONTINUE
  110 CONTINUE
  200 IF(MB.EQ.0) RETURN
C forward reduction and back substitution on B
      IF(N.EQ.1) B(1)=B(1)/A(1,1)
      IF(N.EQ.1) RETURN
      N1=N-1
      DO 210 I=1,N1
      MIN=KMIN0(NBD,N-I+1)
      DO 220 J=2,MIN
      JJ=I+J-1
      B(JJ)=B(JJ)-B(I)*A(I,J)
  220 CONTINUE
  210 CONTINUE
      DO 230 I=1,N
      B(I)=B(I)/A(I,1)
  230 CONTINUE
      DO 240 I=1,N1
      II=N-I+1
      MIN=KMIN0(NBD,II)
      DO 250 J=2,MIN
      JJ=II-J+1
      B(JJ)=B(JJ)-B(II)*A(JJ,J)
  250 CONTINUE
  240 CONTINUE
      RETURN
  300 WRITE(6,3000)
 3000 FORMAT(2(/),5X,'DMIN IS LESS THAN TOL; EXECUTION STOPPED.')
      WRITE(6,3001) AVOD,DMIN,TOL,IROW
 3001 FORMAT(5X,'AVOD =',1X,E12.5,4X,'DMIN =',1X,E12.5,4X,'TOL =',1X,
     * E12.5,4X,'IROW =',I5)
      STOP
      END

      SUBROUTINE CORGEN(COOR,NNP,NTYPE)
C Generates matrix COOR
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),COOR1(3),NTYPE(NNP)
      L=0
      LG=0
   10 READ(5,*) N,NG,(COOR1(J),J=1,3),NTYPE(N)
      IF(N.EQ.0) RETURN
      DO 20 J=1,3
      COOR(N,J)=COOR1(J)
   20 CONTINUE
      NB=L+LG
      IF((LG.LE.0).OR.(NB.GE.N)) GO TO 40
      NE=N-1
      NSPACE=(N-L-1)/LG+1
      DO 30 J=1,3
      CINC=(COOR(N,J)-COOR(L,J))/NSPACE
      DO 31 I=NB,NE,LG
      COOR(I,J)=COOR(I-LG,J)+CINC
      NTYPE(I)=NTYPE(I-LG)
   31 CONTINUE
   30 CONTINUE
   40 L=N
```

```
      LG=NG
      GO TO 10
      END

      SUBROUTINE DFCALC(DF,DQ,X,R,THETA,PSIMAX,ALPHA,GAMMA,ETA,PHI,
     *  OMEGA,PSI,BETA,DBETADPSI,YST,DFDEP,SXX,SYY,SZZ,SXY,SYZ,SXZ)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION DF(6,1),DQ(6,1)
      REAL(KIND=16) PSIMAX,PSI,PHI,X,R,THETA,J2,J3
C
C Calculate all derivatives related to the peak loading surface
C  peak parameters
      RT = -0.5502721390459450Q0+QSQRT(-1.43439820150635Q0*X
     *          +0.0924044582071Q0*X**2.Q0+0.302799427010 2Q0)
      RC = -0.3519588917484 1Q0+QSQRT(-1.971053263759 4Q0*X
     *          +0.3102749931823 9Q0*X**2.Q0+0.1238750614807 69Q0)
      CALL RCALC(RT,RC,THETA,RPEAK)
      DRTDX = 0.5Q0/(RT+0.5502721390459450Q0)*(-1.434398201506 35Q0
     *              +0.1848089164142Q0*X)
      DRCDX = 0.5Q0/(RC+0.3519588917484 1Q0)*(-1.971053263759 4Q0
     *              +0.6205499863647 8Q0*X)
      CALL DRCALC(RC,RT,DRCDX,DRTDX,THETA,DRPEAKDX,DRPEAKDTH)
      IF (X .GE. 0.) THEN
          RT = 0.Q0
          RC = 0.Q0
          RPEAK = 0.Q0
          DRTDX = 0.Q0
          DRCDX = 0.Q0
          DRPEAKDX = 0.Q0
          DRPEAKDTH = 0.Q0
      ENDIF
      DFPEAKDX = -DRPEAKDX
      DFPEAKDTH = -DRPEAKDTH
C
C Calculate all derivatives related to the yield loading surface
      IF (PSI .LT. PSIMAX) THEN
          TEMPA = QSQRT(2.Q0/3.Q0)*0.45Q0*(1.Q0+QSQRT(3.Q0)/0.45Q0)
          TEMPB = QSQRT(2.Q0/3.Q0)*0.522Q0*(1.Q0+QSQRT(3.Q0)/1.044Q0)
          TEMPC = -X/(1.Q0-X)
          RT = TEMPC*TEMPB
          RC = TEMPC*TEMPA
          CALL RCALC(RT,RC,THETA,RYIELD)
c          DRTDX = 0.5Q0/(RT+0.88222488954238 1Q0)
c     *                 *(-1.47Q0+0.262233966735318Q0*X)
c          DRCDX = 0.5Q0/(RC+1.10899432823906 0Q0)
c     *                 *(-3.61Q0+0.356609103684252Q0*X)
          TEMPC = (1.Q0-X)**2.Q0
          DRTDX = -TEMPB/TEMPC
          DRCDX = -TEMPA/TEMPC
          CALL DRCALC(RC,RT,DRCDX,DRTDX,THETA,DRYIELDDX,DRYIELDDTH)
          IF (X .GE. 0.) THEN
              RT = 0.Q0
              RC = 0.Q0
              RYIELD = 0.Q0
              DRCDX = 0.Q0
              DRTDX = 0.Q0
              DRYIELDDX = 0.Q0
              DRYIELDDTH = 0.Q0
          ENDIF
          DFYIELDDX = -DRYIELDDX
          DFYIELDDTH = -DRYIELDDTH
          DFDX = BETA*DFPEAKDX + (1.Q0-BETA)*DFYIELDDX
          DFDTH = BETA*DFPEAKDTH + (1.Q0-BETA)*DFYIELDDTH
C
C Vary Psi relationship here
C
          DPSIDEP = 1.Q0/(PHI+ALPHA*(QABS(X))**GAMMA)
          DFDEP = (RYIELD-RPEAK)*DBETADPSI*DPSIDEP
C
C Calculate all derivatives related to the residual loading surface
      ELSE IF (PSI .GT. PSIMAX) THEN
          RTT = -0.7416615013716 67Q0+QSQRT(-0.9764196664448 9Q0*X
     *              +0.0565121387786362Q0*X**2.Q0+0.550061782616876 0Q0)
```

```
          RCC = -0.474373208368885Q0+QSQRT(-1.34173004980343OQ0*X
     *              +0.189756033577570Q0*X**2.Q0+0.2250299408189Q0)
C Parameters to have round residual surface
c          RTT = RCC
C
          IF (X .GE. 0.) THEN
              RTT = 0.Q0
              RCC = 0.Q0
          ENDIF
          IF (RCC .GE. RC .AND. RTT .GE. RT) THEN
              DFDX = DFPEAKDX
              DFDTH = DFPEAKDTH
              RRES = RPEAK
          ELSE
              IF (RT .GT. RTT) THEN
                DRTDX = 0.5Q0/(RTT+0.74166150137166 7Q0)*
     *                    (-0.9764196664448 9Q0+0.1130242775572724Q0*X)
              ENDIF
              IF (RC .GT. RCC) THEN
                DRCDX = 0.5Q0/(RCC+0.474373208368885Q0)*
     *                    (-1.34173004980343OQ0+0.37951206715514Q0*X)
              ENDIF
C Parameters to have round residual surface
c              IF (RT .GT. RTT) THEN
c                DRTDX = DRCDX
c              ENDIF
C
              RT = QMIN1(RT,RTT)
              RC = QMIN1(RC,RCC)
              CALL RCALC(RT,RC,THETA,RRES)
              CALL DRCALC(RC,RT,DRCDX,DRTDX,THETA,DRRESDX,DRRESDTH)
              IF (X .GE. 0.) THEN
                RC = 0.Q0
                RT = 0.Q0
                RRES = 0.Q0
                DRCDX = 0.Q0
                DRTDX = 0.Q0
                DRRESDX = 0.Q0
                DRRESDTH = 0.Q0
              ENDIF
              DFRESDX = -DRRESDX
              DFRESDTH = -DRRESDTH
              DFDX = BETA*DFPEAKDX + (1.Q0-BETA)*DFRESDX
              DFDTH = BETA*DFPEAKDTH + (1.Q0-BETA)*DFRESDTH
          ENDIF
C
C Vary Psi relationship here
C
          DPSIDEP = 1.Q0/(PHI+ALPHA*(QABS(X))**GAMMA)
          DFDEP = (RRES-RPEAK)*DBETADPSI*DPSIDEP
      ELSE
          DFDX = DFPEAKDX
          DFDTH = DFPEAKDTH
          DFDEP = 0.Q0
      ENDIF
      DQDX = OMEGA*DFDX
      DQDTH = OMEGA*DFDTH
C
C Calculate all derivatives related to the invariants
      CALL PRIN(X,R,THETA,S1,S2,S3)
      X = X*YST
      R = R*YST
      J2 = R**2.Q0/2.Q0
      if (idebug .eq. 1) write(11,*) 'J2 = ',J2
      DJ2DSXX = (2.Q0*SXX-SYY-SZZ)/3.Q0
      DJ2DSYY = (-SXX+2.Q0*SYY-SZZ)/3.Q0
      DJ2DSZZ = (-SXX-SYY+2.Q0*SZZ)/3.Q0
      DJ2DSXY = 2.Q0*SXY
      DJ2DSYZ = 2.Q0*SYZ
      DJ2DSXZ = 2.Q0*SXZ

      TEMPA = X/QSQRT(3.Q0)
      J3 = (SXX-TEMPA)*(SYY-TEMPA)*(SZZ-TEMPA)-((SXX-TEMPA)*SYZ**2.Q0
     *        +SXY**2.Q0*(SZZ-TEMPA)+SXZ**2.Q0*(SYY-TEMPA))
     *        +2.Q0*SXY*SYZ*SXZ
      DJ3DSXX = (4.Q0*SYY*SZZ-2.Q0*SXX*(-SXX+SYY+SZZ)
```

```
     *          -(SYY**2.Q0+SZZ**2.Q0))/9.Q0-(2.Q0*SYZ**2.Q0
     *          -(SXY**2.Q0+SXZ**2.Q0))/3.Q0
      DJ3DSYY = (4.Q0*SXX*SZZ-2.Q0*SYY*(SXX-SYY+SZZ)
     *          -(SXX**2.Q0+SZZ**2.Q0))/9.Q0
     *          -(2.Q0*SXZ**2.Q0-(SXY**2.Q0+SYZ**2.Q0))/3.Q0
      DJ3DSZZ = (4.Q0*SXX*SYY-2.Q0*SZZ*(SXX+SYY-SZZ)
     *          -(SXX**2.Q0+SYY**2.Q0))/9.Q0
     *          -(2.Q0*SXY**2.Q0-(SXZ**2.Q0+SYZ**2.Q0))/3.Q0
      DJ3DSXY = 2.Q0*SXZ*SYZ+2.Q0/3.Q0*SXY*(SXX+SYY-2.Q0*SZZ)
      DJ3DSYZ = 2.Q0*SXY*SXZ+2.Q0/3.Q0*SYZ*(-2.Q0*SXX+SYY+SZZ)
      DJ3DSXZ = 2.Q0*SXY*SYZ+2.Q0/3.Q0*SXZ*(SXX-2.Q0*SYY+SZZ)
      TEMPA = QSIN(3.Q0*THETA)
      DTHDJ2 = 0.75Q0*QSQRT(3.Q0)*J3/(J2**2.5Q0)/TEMPA
      DTHDJ3 = -QSQRT(3.Q0)/2.Q0/(J2**1.5Q0)/TEMPA
      IF (TEMPA .LT. 1.Q-6) THEN
        DTHDJ2 = QSQRT(3.Q0)/2.Q0*J3/(J2**(2.5Q0))
        DTHDJ3 = -1.Q0/QSQRT(3.Q0)/(J2**(1.5Q0))
      ENDIF
      DTHDSXX = DTHDJ3*DJ3DSXX+DTHDJ2*DJ2DSXX
      DTHDSYY = DTHDJ3*DJ3DSYY+DTHDJ2*DJ2DSYY
      DTHDSZZ = DTHDJ3*DJ3DSZZ+DTHDJ2*DJ2DSZZ
      DTHDSXY = DTHDJ3*DJ3DSXY+DTHDJ2*DJ2DSXY
      DTHDSYZ = DTHDJ3*DJ3DSYZ+DTHDJ2*DJ2DSYZ
      DTHDSXZ = DTHDJ3*DJ3DSXZ+DTHDJ2*DJ2DSXZ

      TEMPSXX = DJ2DSXX/R/YST
      TEMPSYY = DJ2DSYY/R/YST
      TEMPSZZ = DJ2DSZZ/R/YST
      TEMPSXY = DJ2DSXY/R/YST
      TEMPSYZ = DJ2DSYZ/R/YST
      TEMPSXZ = DJ2DSXZ/R/YST
C
C Correct for case where R=0
      IF (R .EQ. 0. .OR. X .GE. 0.) THEN
          TEMPSXX = 0.Q0
          TEMPSYY = 0.Q0
          TEMPSZZ = 0.Q0
          TEMPSXY = 0.Q0
          TEMPSYZ = 0.Q0
          TEMPSXZ = 0.Q0
      ENDIF
C
C Calculated final yield and loading surface derivatives
      DF(1,1) = TEMPSXX+DFDX/QSQRT(3.Q0)/YST+DFDTH*DTHDSXX
      DF(2,1) = TEMPSYY+DFDX/QSQRT(3.Q0)/YST+DFDTH*DTHDSYY
      DF(3,1) = TEMPSZZ+DFDX/QSQRT(3.Q0)/YST+DFDTH*DTHDSZZ
      DF(4,1) = TEMPSXY+DFDTH*DTHDSXY
      DF(5,1) = TEMPSYZ+DFDTH*DTHDSYZ
      DF(6,1) = TEMPSXZ+DFDTH*DTHDSXZ
      DQ(1,1) = TEMPSXX+DQDX/QSQRT(3.Q0)/YST+DQDTH*DTHDSXX
      DQ(2,1) = TEMPSYY+DQDX/QSQRT(3.Q0)/YST+DQDTH*DTHDSYY
      DQ(3,1) = TEMPSZZ+DQDX/QSQRT(3.Q0)/YST+DQDTH*DTHDSZZ
      DQ(4,1) = TEMPSXY+DQDTH*DTHDSXY
      DQ(5,1) = TEMPSYZ+DQDTH*DTHDSYZ
      DQ(6,1) = TEMPSXZ+DQDTH*DTHDSXZ
      X = X/YST
      R = R/YST
      RETURN
      END

      SUBROUTINE DRCALC(RC,RT,DRCDX,DRTDX,THETA,DRDX,DRDTH)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      REAL(KIND=16) RC,RT,DRCDX,DRTDX,THETA,TEMPA,TEMPB,DRDX,DRDTH
      TEMPA = 4.Q0*(RC**2.Q0-RT**2.Q0)*(QCOS(THETA))**2.Q0
     *          +(RC-2.Q0*RT)**2.Q0
      TEMPB = QSQRT(4.Q0*(RC**2.Q0-RT**2.Q0)*(QCOS(THETA))**2.Q0
     *          +5.Q0*RT**2.Q0-4.Q0*RT*RC)
      DRDX = (2.Q0*DRCDX*(RC**2.Q0-RT**2.Q0)*QCOS(THETA)+4.Q0*RC*
     *  (RC*DRCDX-RT*DRTDX)*QCOS(THETA)+DRCDX*(2.Q0*RT-RC)*TEMPB+RC
     *  *(2.Q0*DRTDX-DRCDX)*TEMPB+(RC*(2.Q0*RT-RC)*(8.Q0*(RC*DRCDX
     *  -RT*DRTDX)*(QCOS(THETA))**2.Q0+10.Q0*RT*DRTDX-4.Q0*DRTDX*RC
     *  -4.Q0*RT*DRCDX))/(2.Q0*TEMPB))/TEMPA-((2.Q0*RC*(RC**2.Q0
     *  -RT**2.Q0)*QCOS(THETA)+RC*(2.Q0*RT-RC)*TEMPB)*(8.Q0*(RC*DRCDX
     *  -RT*DRTDX)*(QCOS(THETA))**2.Q0+2.Q0*(RC-2.Q0*RT)*(DRCDX
```

```
     *  -2.Q0*DRTDX)))/(TEMPA**2.Q0)
      TEMPA = RC**2.Q0 - RT**2.Q0
      TEMPB = 2.Q0*RT - RC
      DRDTH = 2.Q0*QSIN(THETA)*RC*TEMPA/((4.Q0*(QCOS(THETA))**2.Q0*TEMPA
     *  +TEMPB**2.Q0)**2.Q0)/QSQRT((4.Q0*(QCOS(THETA))**2.Q0-1.Q0)*TEMPA
     *  +TEMPB**2.Q0)*((4.Q0*(QCOS(THETA))**2.Q0*TEMPA-TEMPB**2.Q0)
     *  *QSQRT((4.Q0*(QCOS(THETA))**2.Q0-1.Q0)*TEMPA+TEMPB**2.Q0)
     *  +8.Q0*(QCOS(THETA))**3.Q0*TEMPA*TEMPB+2.Q0*QCOS(THETA)
     *  *TEMPB**3.Q0-4.Q0*QCOS(THETA)*TEMPA*TEMPB)
      IF (QSIN(3.Q0*THETA) .LT. 1.Q-6) THEN
        DRDTH = 1.5Q0*RC*TEMPA/(TEMPB**2.Q0)
      ENDIF
      RETURN
      END

      SUBROUTINE FCALC(F,ELAS,SIG0,DSIG,X,R,THETA,PSI,PSIMAX,BETA,YST)
C
C Subroutine to calculate F (the loading function) given the current
C  stress state (SIG0) the stress increment (DSIG) and current hardening
C  parameter (PSI).  ELAS indicates the ratio of DSIG which is to be
C  applied.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION SIG0(6),DSIG(6)
      REAL(KIND=16) F,ELAS,SIG0,DSIG
      REAL(KIND=16) SXX,SYY,SZZ,SXY,SYZ,SXZ,I1,I2,J2,J3,X,R,THETA
      REAL(KIND=16) RT,RC,RYIELD,RPEAK,RRES,PSIMAX,PSI,BETA,YST
      SXX = SIG0(1)+ELAS*DSIG(1)
      SYY = SIG0(2)+ELAS*DSIG(2)
      SZZ = SIG0(3)+ELAS*DSIG(3)
      SXY = SIG0(4)+ELAS*DSIG(4)
      SYZ = SIG0(5)+ELAS*DSIG(5)
      SXZ = SIG0(6)+ELAS*DSIG(6)
      I1 = SXX+SYY+SZZ
      I2 = (SXX*SYY+SYY*SZZ+SZZ*SXX)-SXY**2.Q0-SYZ**2.Q0-SXZ**2.Q0
      TEMP = I1/3.Q0
      J2 = 3.Q0*TEMP**2.Q0-I2
      J3 = (SXX-TEMP)*(SYY-TEMP)*(SZZ-TEMP)-((SXX-TEMP)*SYZ**2.Q0
     *      +SXY**2.Q0*(SZZ-TEMP)+SXZ**2.Q0*(SYY-TEMP))+2.Q0*SXY*SYZ*SXZ
      X = I1/QSQRT(3.Q0)/YST
      R = QSQRT(2.Q0*J2)/YST
      TEMP = 1.5Q0*QSQRT(3.Q0)*J3/(J2**(1.5Q0))
      IF (TEMP .LT. -1.) TEMP=-1.Q0
      IF (TEMP .GT. 1.) TEMP=1.Q0
      THETA = QACOS(TEMP)/3.Q0
      CALL PRIN(X,R,THETA,S1,S2,S3)
      IF (X .GE. 0.) THEN
            F = R
            GO TO 10
      ENDIF
C Below two expressions use fit for peak loading surface

      RT = -0.550272139045945Q0+QSQRT(-1.43439820150635Q0*X
     *          +0.0924044582071Q0*X**2.Q0+0.302799427010 2Q0)
      RC = -0.31959889174841Q0+QSQRT(-1.971053263759 4Q0*X
     *          +0.3102749931823 9Q0*X**2.Q0+0.12387506148076 9Q0)
      CALL RCALC(RT,RC,THETA,RPEAK)
      IF (PSI .LT. PSIMAX) THEN
C
C Below two expressions use fit for yield loading surface
            TEMPA = QSQRT(2.Q0/3.Q0)*0.45Q0*(1.Q0+QSQRT(3.Q0)/0.45Q0)
            TEMPB = QSQRT(2.Q0/3.Q0)*0.522Q0*(1.Q0+QSQRT(3.Q0)/1.044Q0)
            TEMPC = -X/(1.Q0-X)
            RT = TEMPC*TEMPB
            RC = TEMPC*TEMPA
            CALL RCALC(RT,RC,THETA,RYIELD)
            F = R-BETA*(RPEAK-RYIELD)-RYIELD
      ELSE IF (PSI .GT. PSIMAX) THEN
            RT = -0.741661501371667Q0+QSQRT(-0.9764196664448 9Q0*X
     *             +0.0565121387786362Q0*X**2.Q0+0.550061782616876 0Q0)
            RC = -0.474373208368885Q0+QSQRT(-1.341730049803430Q0*X
     *             +0.189756033577570Q0*X**2.Q0+0.22502994081818 9Q0)
C Parameters to have round residual surface
c            RT = RC
```

```
C
            CALL RCALC(RT,RC,THETA,RRES)
            RRES = QMIN1(RRES,RPEAK)
            F = R-BETA*(RPEAK-RRES)-RRES
         ELSE F = R-RPEAK
         ENDIF
      10 RETURN
         END

         SUBROUTINE FILLF(ID,R,F,AXLD,MX,MY,NNP,NEQ)
C Transfers nodal loads from R to F.
         IMPLICIT REAL(KIND=16) (A-H,O-Z)
         IMPLICIT INTEGER(KIND=8) (I-N)
         COMMON /DEBUG/ IDEBUG
         DIMENSION ID(NNP,3),R(NNP,3),F(NEQ)
         REAL(KIND=16) MX,MY
         DO 10 I=1,NNP
         DO 11 J=1,3
         IROW=ID(I,J)
         IF(IROW.EQ.0) GO TO 11
         F(IROW)=R(I,J)
      11 CONTINUE
      10 CONTINUE
         F(NEQ-2)=AXLD
         F(NEQ-1)=MX
         F(NEQ)=MY
         RETURN
         END

         SUBROUTINE IDGEN(ID,NNP,NTYPE)
C Generates vector ID
         IMPLICIT REAL(KIND=16) (A-H,O-Z)
         IMPLICIT INTEGER(KIND=8) (I-N)
         COMMON /DEBUG/ IDEBUG
         DIMENSION ID(NNP,3),ID1(3),NTYPE(NNP)
      10 READ(5,*) N,NE,NG,(ID1(J),J=1,3)
         IF(N.EQ.0) GO TO 40
         DO 20 J=1,3
         ID(N,J)=ID1(J)
      20 CONTINUE
         NB=N+NG
         IF((NG.LE.0).OR.(NB.GT.NE)) GO TO 10
         DO 30 I=NB,NE,NG
         DO 31 J=1,3
         ID(I,J)=ID1(J)
      31 CONTINUE
      30 CONTINUE
         GO TO 10
      40 DO 50 I=1,NNP
         IF (NTYPE(I) .EQ. 1) ID(I,3)=0
      50 CONTINUE
         END

         SUBROUTINE JACCOMP (COOR,LM,XJ,XI,P,DP,DET,NNP,NEN,IJK)
C Computes the Jacobian matrix, it's inverse, and it's determinant
         IMPLICIT REAL(KIND=16) (A-H,O-Z)
         IMPLICIT INTEGER(KIND=8) (I-N)
         COMMON /DEBUG/ IDEBUG
         DIMENSION COOR(NNP,3),LM(NEN),XJ(3,3),XI(3,3),P(4,NEN),DP(3,NEN)
C
         XJ=0.Q0
         DO 10 I=1,NEN
         J1 = LM(I)
         IF (J1.EQ.0) GO TO 10
         DO 11 J=1,3
            XJ(1,J) = XJ(1,J) + P(2,I)*COOR(J1,J)
            XJ(2,J) = XJ(2,J) + P(3,I)*COOR(J1,J)
            XJ(3,J) = XJ(3,J) + P(4,I)*COOR(J1,J)
      11 CONTINUE
      10 CONTINUE
C XJ is the Jacobian matrix at the current integration point.
C
         XI(1,1) =  XJ(2,2)*XJ(3,3) - XJ(3,2)*XJ(2,3)
         XI(2,2) =  XJ(1,1)*XJ(3,3) - XJ(3,1)*XJ(1,3)
         XI(3,3) =  XJ(1,1)*XJ(2,2) - XJ(2,1)*XJ(1,2)
         XI(1,2) = -XJ(1,2)*XJ(3,3) + XJ(3,2)*XJ(1,3)
```

```
      XI(2,1) = -XJ(2,1)*XJ(3,3) + XJ(3,1)*XJ(2,3)
      XI(1,3) =  XJ(1,2)*XJ(2,3) - XJ(2,2)*XJ(1,3)
      XI(3,1) =  XJ(2,1)*XJ(3,2) - XJ(3,1)*XJ(2,2)
      XI(2,3) = -XJ(1,1)*XJ(2,3) + XJ(2,1)*XJ(1,3)
      XI(3,2) = -XJ(1,1)*XJ(3,2) + XJ(3,1)*XJ(1,2)
      DET = XJ(1,1)*XI(1,1) + XJ(1,2)*XI(2,1) + XJ(1,3)*XI(3,1)
C DET is the determinant of XJ.
      IF (DET.LE.0.) WRITE(6,1000) IJK
      IF (DET.LE.0.) STOP
      DO 20 I=1,3
      DO 21 J=1,3
      XI(I,J) = XI(I,J)/DET
   21 CONTINUE
   20 CONTINUE
C XI is the inverse of XJ.
C
      DP = 0.Q0
      DO 30 I=1,3
      DO 31 J=1,NEN
      DP(I,J) = XI(I,1)*P(2,J) + XI(I,2)*P(3,J) + XI(I,3)*P(4,J)
   31 CONTINUE
   30 CONTINUE
C DP contains terms of the nodal displacement (pressure) to strain (pressure)
C    gradient) transformation matrix at the current integration point (X,Y,Z axes).
 1000 FORMAT (//,3X,'DETERMINANT OF JACOBIAN NON-POSITIVE IN ELEMENT ',
     1 I5,' EXECUTION STOPPED')
      RETURN
      END

      SUBROUTINE LCLCOR(COOR,LM,COORL,NNP,NEN)
C Transfers nodal coordinates for an element from COOR to COORL.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),LM(NEN),COORL(NEN,3)
      DO 10 I=1,NEN
      LI=LM(I)
      DO 11 J=1,3
      IF(LI.EQ.0) COORL(I,J)=0.Q0
      IF(LI.NE.0) COORL(I,J)=COOR(LI,J)
   11 CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE LCLID(ID,LM,IDL,NEDN,NED,NEN,NEL,NEQ,NNP)
C Transfers equation numbers for all elements from ID to IDL.
C   For elements having half their nodes in the top plane
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION ID(NNP,3),LM(NEN,NEL),IDL(NED,NEL)
      DO 10 N=1,NEL
      DO 20 I=1,NEN/2
      LI=LM(I,N)
      DO 21 J=1,NEDN-1
      IF(LI.EQ.0) IDL((I-1)*(NEDN-1)+J,N)=0
      IF(LI.NE.0) IDL((I-1)*(NEDN-1)+J,N)=ID(LI,J)
   21 CONTINUE
   20 CONTINUE
      DO 30 I=NEN/2+1,NEN
      LI=LM(I,N)
      DO 31 J=1,NEDN
      IF(LI.EQ.0) IDL((I-1)*NEDN+J-NEN/2,N)=0
      IF(LI.NE.0) IDL((I-1)*NEDN+J-NEN/2,N)=ID(LI,J)
   31 CONTINUE
   30 CONTINUE
      DO 40 I=1,3
         IDL(NED-I+1,N)=NEQ-I+1
   40 CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE LCLID2(ID,LM,IDL,NNP,NEDN,NEN,NEL)
```

```
C Transfers equation numbers for all elements from ID to IDL.
C  For elements which do not lie in the top plane.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION ID(NNP,6),LM(NEN,NEL),IDL(NEDN,NEN,NEL)
      DO 10 N=1,NEL
      DO 11 I=1,NEN
      LI=LM(I,N)
      DO 12 J=1,NEDN
      IF(LI.EQ.0) IDL(J,I,N)=0
      IF(LI.NE.0) IDL(J,I,N)=ID(LI,J)
   12 CONTINUE
   11 CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE LCLID3(ID,LM,IDL,NEDN,NED,NEN,NEL,NEQ,NNP)
C Transfers equation numbers for all elements from ID to IDL.
C   For elements having all their nodes in the top plane
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION ID(NNP,3),LM(NEN,NEL),IDL(NED,NEL)
      DO 10 N=1,NEL
      DO 20 I=1,NEN
      LI=LM(I,N)
      DO 21 J=1,NEDN-1
      IF(LI.EQ.0) IDL((I-1)*(NEDN-1)+J,N)=0
      IF(LI.NE.0) IDL((I-1)*(NEDN-1)+J,N)=ID(LI,J)
   21 CONTINUE
   20 CONTINUE
      DO 30 I=1,3
         IDL(NED-I+1,N)=NEQ-I+1
   30 CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE LCLX(X,COORL,IDL,XL,DX,NEDN,NEN,NED,NEQ,IFLAG)
C Transfers displacements for an element from X to XL.
C   For elements having half their nodes in the top plane
C   If IFLAG=0 calculates displacements assuming current coordinates
C   If IFLAG=1 calculates displacements assuming coordinates of previous time step
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION X(NEQ),COORL(NEN,3),XL(NEN,NEDN),IDL(NED),DX(NEQ)
      DO 10 I=1,NEN/2
         DO 11 J=1,NEDN-1
            IDI=IDL((I-1)*(NEDN-1)+J)
            IF(IDI.EQ.0) XL(I,J)=0.Q0
            IF(IDI.NE.0) XL(I,J)=X(IDI)
   11    CONTINUE
   10 CONTINUE
      DO 20 I=NEN/2+1,NEN
         DO 21 J=1,NEDN
            IDI=IDL((I-1)*NEDN+J-NEN/2)
            IF(IDI.EQ.0) XL(I,J)=0.Q0
            IF(IDI.NE.0) XL(I,J)=X(IDI)
   21    CONTINUE
   20 CONTINUE
      DO 30 I=1,NEN/2
         ID1=IDL((I-1)*(NEDN-1)+1)
         ID2=IDL((I-1)*(NEDN-1)+2)
         XL(I,3)=X(NEQ-2) + (COORL(I,2)-DX(ID2)*IFLAG) * X(NEQ-1)
     *      - (COORL(I,1)-DX(ID1)*IFLAG) * X(NEQ)
   30 CONTINUE
      RETURN
      END

      SUBROUTINE LCLX2(X,IDL,XL,NEDN,NEN,NED,NEQ)
C Transfers displacements for an element from X to XL.
C   For elements which do not lie in the top plane
```

```
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION X(NEQ),XL(NEN,NEDN),IDL(NEDN,NEN)
      DO 10 I=1,NEN
         DO 11 J=1,NEDN
            IDI=IDL(J,I)
            IF(IDI.EQ.0) XL(I,J)=0.QO
            IF(IDI.NE.0) XL(I,J)=X(IDL(J,I))
   11    CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE LMGEN(LM,MAT,NEN,NEL)
C Generates vector MAT and matrix LM
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION LM(NEN,NEL),MAT(NEL),LM1(NEN)
   10 READ(5,*) N,NE,NG,MAT1,(LM1(I),I=1,NEN),LMG
      IF(N.EQ.0) RETURN
      MAT(N)=MAT1
      DO 20 I=1,NEN
      LM(I,N)=LM1(I)
   20 CONTINUE
      NB=N+NG
      IF((NG.LE.0).OR.(NB.GT.NE)) GO TO 10
      DO 30 J=NB,NE,NG
      MAT(J)=MAT1
      DO 31 I=1,NEN
      IF(LM1(I).EQ.0) LM(I,J)=0
      IF(LM1(I).NE.0) LM(I,J)=LM(I,J-NG)+LMG
   31 CONTINUE
   30 CONTINUE
      GO TO 10
      END

      SUBROUTINE MCANAL(COOR,ID,R,B,H,F,RTPT,DX,XT,XTPT,ELRT,ELRTPT,
     *   MOMENT,NTYPE,NSSTEPS,NHSTEPS,ZCOR,NITER)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NPAR(8)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      COMMON /TOL/ TOLF,TOLM,NITMAX
      COMMON /DEBUG/ IDEBUG
      COMMON /MISC/ PSIMAX,KAPPA
      DIMENSION COOR(NNP,3),ID(NNP,3),R(NNP,3),B(N5-1),H(NEQ,MBAND),
     *   F(NEQ),RTPT(NEQ),DX(NEQ),XT(NEQ),XTPT(NEQ),
     *   ELRT(N12-N11),ELRTPT(N13-N12),NTYPE(NNP)
      REAL(KIND=16) MOMENT,KAPPA
      TIME=0.QO
      DT=QABS(MOMENT/NHSTEPS)
C Calculate elastic stiffness matrix and factor
C Remove below code to iterate with tangent stiffness
C
      IGO=3
      H=0.QO
      REWIND 7
      DO 10 IEG=1,NEG1
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSA(COOR,B,H)
   10 CONTINUE
      DO 20 IEG=1,NEG10
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSB(COOR,B,H)
   20 CONTINUE
      DO 30 IEG=1,NEG11
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSC(COOR,B,H)
   30 CONTINUE
      DO 40 IEG=1,NEG100
```

```
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDA(COOR,B,H)
   40 CONTINUE
      DO 50 IEG=1,NEG110
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDB(COOR,B,H)
   50 CONTINUE
c add large stiffness in two rotational DOFs
      H(NEQ-1,1)=H(NEQ-1,1)+1.0Q17
c      H(NEQ,1)=H(NEQ,1)+1.0Q17
c
      CALL BSOLVE(H,DX,NEQ,MBAND,1,0)
      DO 100 N=1,NHSTEPS
      TIME=TIME+DT
      FRAC=QABS(TIME/MOMENT)
C  Set previous time step displacement and element responses equal
C     to current time step values
C
      XT = XTPT
      ELRT = ELRTPT
      NITER=0
      DO 200 K=1,NITMAX
      DO 205 I=1,NEQ-2
         DX(I)=F(I)-RTPT(I)
  205 CONTINUE
      DO 206 I=NEQ-1,NEQ
         DX(I)=F(I)*FRAC-RTPT(I)
  206 CONTINUE
      DO 210 I=1,NNP
         DO 211 J=1,3
            IDI=ID(I,J)
            IF(IDI.EQ.0) GO TO 211
            IF(QABS(DX(IDI)).GT.TOLF) GO TO 1
  211    CONTINUE
  210 CONTINUE
      IF(QABS(DX(NEQ-2)).GT.TOLF) GO TO 1
      IF(QABS(DX(NEQ-1)).GT.TOLM) GO TO 1
      IF(QABS(DX(NEQ)).GT.TOLM) GO TO 1
      GO TO 2
    1 CALL BSOLVE(H,DX,NEQ,MBAND,0,1)
      DO 230 I=1,NNP
         DO 231 J=1,3
            IDI=ID(I,J)
            IF (IDI.EQ.0) GO TO 231
            COOR(I,J)=COOR(I,J) + DX(IDI)
  231    CONTINUE
  230 CONTINUE
      DO 240 I=1,NEQ
         XTPT(I) = XTPT(I) + DX(I)
         DX(I) = XTPT(I)- XT(I)
  240 CONTINUE
      DO 250 I=1,NNP
         IF (NTYPE(I).EQ.1) COOR(I,3)=ZCOR+XTPT(NEQ-2)
     *      +COOR(I,2)*XTPT(NEQ-1)-COOR(I,1)*XTPT(NEQ)
  250 CONTINUE
      RTPT=0.Q0
      IGO=4
      REWIND 7
      DO 260 IEG=1,NEG1
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSA(COOR,B,H)
  260 CONTINUE
      DO 261 IEG=1,NEG10
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSB(COOR,B,H)
  261 CONTINUE
      DO 262 IEG=1,NEG11
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSC(COOR,B,H)
  262 CONTINUE
      DO 263 IEG=1,NEG100
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDA(COOR,B,H)
  263 CONTINUE
      DO 264 IEG=1,NEG110
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
```

```
      CALL SOLIDB(COOR,B,H)
  264 CONTINUE
c add large FORCE in two rotational DOFs
      RTPT(NEQ-1)=RTPT(NEQ-1)+1.0Q17*DX(NEQ-1)
      RTPT(NEQ)=RTPT(NEQ)+1.0Q17*DX(NEQ)
c
      NITER=K
  200 CONTINUE

    2 WRITE(6,2000) TIME,NITER
      WRITE(*,2000) TIME,NITER
 2000 FORMAT(1X,'MOMENT =',E12.5,5X,'NITER =',I5)
      IF (NITER .EQ. NITMAX) WRITE(6,2001) F(NEQ-1)*FRAC-RTPT(NEQ-1)
 2001 FORMAT(6X, 'MOMENT IMBALANCE = ',E12.5)
      WRITE(8,2005) XTPT(NEQ-1),F(NEQ-1)*FRAC-mrotx,XTPT(NEQ),
     *  F(NEQ)*FRAC-mroty
 2005 FORMAT(1X,6(1X,E19.12))
  100 CONTINUE
      WRITE(6,2003)
 2003 FORMAT(2(/),25X,'POST MOMENT ELEMENT RESPONSES',/)
      CALL PPICT(COOR,ID,B,H,XTPT,2,NTYPE,TIME)
      RETURN
      END

      SUBROUTINE PPICT(COOR,ID,B,C,XTPT,ND,NTYPE,TIME)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NPAR(8)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),ID(NNP,3),B(N5-1),XTPT(NEQ),NTYPE(NNP)
      IGO=5
      REWIND 7
      DO 100 IEG=1,NEG1
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSA(COOR,B,C)
  100 CONTINUE
      DO 101 IEG=1,NEG10
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSB(COOR,B,C)
  101 CONTINUE
      DO 102 IEG=1,NEG11
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSC(COOR,B,C)
  102 CONTINUE
      DO 103 IEG=1,NEG100
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDA(COOR,B,C)
  103 CONTINUE
      DO 104 IEG=1,NEG110
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDB(COOR,B,C)
  104 CONTINUE
      IF (ND.EQ.1) RETURN
      WRITE(6,1000)
 1000 FORMAT(/,3X,'NODE',8X,'XDISP',8X,'YDISP',8X,'ZDISP')
      DO 200 I=1,NNP
      XDISP=0.Q0
      YDISP=0.Q0
      ZDISP=0.Q0
      ID1=ID(I,1)
      ID2=ID(I,2)
      ID3=ID(I,3)
      IF(ID1.NE.0) XDISP=XTPT(ID1)
      IF(ID2.NE.0) YDISP=XTPT(ID2)
      IF(ID3.NE.0 .AND. NTYPE(I).EQ.0) ZDISP=XTPT(ID3)
      IF(NTYPE(I).EQ.1) THEN
         ZDISP=XTPT(NEQ-2)+COOR(I,2)*XTPT(NEQ-1)-COOR(I,1)*XTPT(NEQ)
      ENDIF
      WRITE(6,2000) I,XDISP,YDISP,ZDISP
 2000 FORMAT(3X,I4,3(1X,E12.5))
  200 CONTINUE
```

```
      WRITE(6,2001)
 2001 FORMAT(/,3X,'TOP PLANE',/,9X,'ZDISP',2X,'CURVATURE-X',
     *  2X,'CURVATURE-Y')
      WRITE(6,2002) XTPT(NEQ-2),XTPT(NEQ-1),XTPT(NEQ)
 2002 FORMAT(1X,3(1X,E12.5),4(/))
      RETURN
      END

      SUBROUTINE PRIN(X,R,THETA,S1,S2,S3)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      REAL(KIND=16) X,R,THETA,PI
C SIGM = sigma_mean = mean stress
C SQJ2 = square root of the invariant J2
C S1,S2,S3 = principal stresses with S1>S2>S3
      PI = 3.14159265358979323846264338327950
      SIGM = X/QSQRT(3.Q0)
      SQJ2 = R/QSQRT(2.Q0)
      TEMP = 2.Q0/QSQRT(3.Q0)*SQJ2
      TEMP2 = 2.Q0/3.Q0*PI
      S1 = SIGM + TEMP*QCOS(THETA)
      S2 = SIGM + TEMP*QCOS(THETA - TEMP2)
      S3 = SIGM + TEMP*QCOS(THETA + TEMP2)
      RETURN
      END

      SUBROUTINE RCALC(RT,RC,THETA,R)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      REAL(KIND=16) R,RT,RC,THETA
      R=(2.Q0*RC*(RC**2.Q0-RT**2.Q0)*QCOS(THETA)+RC*(2.Q0*RT-RC)
     *      *QSQRT(4.Q0*(RC**2.Q0-RT**2.Q0)*(QCOS(THETA))**2.Q0
     *      +5.Q0*RT**2.Q0-4.Q0*RT*RC))/(4.Q0*(RC**2.Q0-RT**2.Q0)
     *      *(QCOS(THETA))**2.Q0+(RC-2.Q0*RT)**2.Q0)
      RETURN
      END

      SUBROUTINE READC(COOR,NNP,ZCOR,NTYPE)
C Generates and prints nodal coordinates.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),NTYPE(NNP)
      CALL CORGEN(COOR,NNP,NTYPE)
      WRITE(6,1000)
 1000 FORMAT(4(/),1X,'NODAL COORDINATES',/,6X,'NODE',14X,'X',12X,'Y',
     * 12X,'Z',4X,'TYPE')
      DO 10 I=1,NNP
      WRITE(6,1001) I,(COOR(I,J),J=1,3),NTYPE(I)
      IF (COOR(I,3).GT.ZCOR) ZCOR=COOR(I,3)
   10 CONTINUE
 1001 FORMAT(6X,I4,2X,3(1X,E12.5),7X,I1)
      RETURN
      END


      SUBROUTINE READID(ID,COOR,NNP,NEQ,NTYPE)
C Generates the nodal boundary conditions.  Numbers the equations and
C prints the equation numbers.  Computes NEQ.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION ID(NNP,3), COOR(NNP,3),NTYPE(NNP)
      ID=1
      CALL IDGEN(ID,NNP,NTYPE)
C Number the nonzero terms of ID from 1 to NEQ.  Set NEQ.
      NEQ=0
      DO 20 I=1,NNP
      DO 21 J=1,3
      IF(ID(I,J).EQ.0) GO TO 21
      NEQ=NEQ+1
      ID(I,J)=NEQ
   21 CONTINUE
   20 CONTINUE
```

```
C Add final 3 DOF's to account for top plane vertical displacement
C  and two rotations
      NEQ=NEQ+3
      WRITE(6,2000)
 2000 FORMAT(4(/),1X,'EQUATION NUMBERS',/,6X,'NODE',4X,'DOF 1',2X,
     * 'DOF 2',2X,'DOF 3')
      DO 30 I=1,NNP
      WRITE(6,3000) I,(ID(I,J),J=1,3)
   30 CONTINUE
 3000 FORMAT(6X,I4,2X,6(3X,I4))
      RETURN
      END

      SUBROUTINE READR(R,ID,NNP)
C Generates and prints nodal loads and specified displacements.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION R(NNP,3)
      R=0.Q0
      CALL RGEN(R,ID,NNP)
      WRITE(6,1000)
 1000 FORMAT(4(/),1X,'SPECIFIED NODAL LOADS',/,6X,
     * 'NODE',10X,'DOF 1',8X,'DOF 2',8X,'DOF 3')
      DO 10 I=1,NNP
      WRITE(6,1001) I,(R(I,J),J=1,3)
 1001 FORMAT(6X,I4,2X,6(1X,E12.5))
   10 CONTINUE
      RETURN
      END

      SUBROUTINE RECT (P,R,S,T)
C RECT COMPUTES SHAPE FUNCTION VALUES AND DERIVATIVES FOR AN 8 NODE ELEMENT
C AT THE LOCAL COORDINATE LOCATION R,S,T.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION P(4,8)
C
      DO 10 I=1,4
C I=1 FOR SHAPE FUNCTION VALUES; =2 FOR R DERIVATIVES; =3 FOR S DERIVATIVES;
C =4 FOR T DERIVATIVES.
      IF (I.EQ.1) GO TO 4
      IF (I.EQ.2) GO TO 5
      IF (I.EQ.3) GO TO 6
      IF (I.EQ.4) GO TO 7
    4 AR = 1.Q0 + R
      SR = 1.Q0 - R
      AS = 1.Q0 + S
      SS = 1.Q0 - S
      AT = 1.Q0 + T
      ST = 1.Q0 - T
      GO TO 8
    5 AR = 1.Q0
      SR = -1.Q0
      GO TO 8
    6 AR = 1.Q0 + R
      SR = 1.Q0 - R
      AS = 1.Q0
      SS = -1.Q0
      GO TO 8
    7 AS = 1.Q0 + S
      SS = 1.Q0 - S
      AT = 1.Q0
      ST = -1.Q0
    8 P(I,1) = 0.125Q0*SR*AS*AT
      P(I,2) = 0.125Q0*SR*SS*AT
      P(I,3) = 0.125Q0*AR*SS*AT
      P(I,4) = 0.125Q0*AR*AS*AT
      P(I,5) = 0.125Q0*SR*AS*ST
      P(I,6) = 0.125Q0*SR*SS*ST
      P(I,7) = 0.125Q0*AR*SS*ST
      P(I,8) = 0.125Q0*AR*AS*ST
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RGEN(R,ID,NNP)
C Generates vector R
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION R(NNP,3),ID(NNP,3),R1(3)
   10 READ(5,*) N,NE,NG,(R1(J),J=1,3)
      IF(N.EQ.0) RETURN
      DO 20 J=1,3
      R(N,J)=R1(J)
C      IF(R1(J).NE.0 .AND. ID(N,J).EQ.1) THEN
C         WRITE(6,1000) N,J
C 1000    FORMAT(/,1X,'NODAL LOADS ARE IGNORED IN THIS PROGRAM.',/,
C     *      'NODE = ',I4,5X,'DOF = ',I1,/)
C         R(N,J)=0
C      ENDIF
   20 CONTINUE
      NB=N+NG
      IF((NG.LE.0).OR.(NB.GT.NE)) GO TO 10
      DO 30 I=NB,NE,NG
      DO 31 J=1,3
      R(I,J)=R1(J)
   31 CONTINUE
   30 CONTINUE
      GO TO 10
      END

      SUBROUTINE SOLIDA(A,B,C)
C Sets up storage for subroutine SOLID1
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      COMMON /INFO/ N5TEMP
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NEN,NINT,NDUM(3)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      REAL(KIND=16), ALLOCATABLE, DIMENSION(:) :: BSUB
      DIMENSION A(N4-1),B(N5-1),C(N13-1)
C
      IF (MAXNEL100.LT.NEL) MAXNEL100=NEL
      IF (NINT.EQ.2) MAXNINT=8
      IF (NINT.EQ.3) MAXNINT=4
      N401=1   +NUMAT                     ! E
      N402=N401+NUMAT                     ! POISSON'S RATIO
      N403=N402+NUMAT                     ! f'c
      N404=N403+NEL                       ! MAT
      N405=N404+NEN*NEL                   ! LM
      N5TEMP=N405+(NEN*3-NEN/2+3)*NEL     ! IDL
      NEND=N5TEMP-1
      IF (IGO.EQ.1) THEN
         ALLOCATE(BSUB(NEND))
         BSUB=0.Q0
         CALL SOLID1(A(N1),A(N2),A(N3),BSUB(1),BSUB(N401),BSUB(N402),
     *      BSUB(N403),BSUB(N404),BSUB(N405),C(1),C(N6),C(N7),C(N8),
     *      C(N9),C(N10),C(N1103),C(N1203))
         WRITE(7) NEND,NELTYP,NEL,NUMAT,NEN,NINT,(NDUM(I),I=1,3),
     *      (BSUB(I),I=1,NEND)
         DEALLOCATE(BSUB)
      ELSE
         CALL SOLID1(A(N1),A(N2),A(N3),B(1),B(N401),B(N402),B(N403),
     *      B(N404),B(N405),C(1),C(N6),C(N7),C(N8),C(N9),C(N10),
     *      C(N1103),C(N1203))
      ENDIF
      RETURN
      END

      SUBROUTINE SOLID1(COOR,ID,R,E,PR,YST,MAT,LM,IDL,H,F,RTPT,DX,XT,
     *    XTPT,ELRT,ELRTPT)
C Performs computations for 6 or 8 node solid elements.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
```

```
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NEN,NINT,NDUM(3)
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),ID(NNP,3),R(NNP,3),E(NUMAT),PR(NUMAT),
     *   YST(NUMAT),MAT(NEL),LM(NEN,NEL),H(NEQ,MBAND),F(NEQ),RTPT(NEQ),
     *   IDL(NEN*3-NEN/2+3,NEL),DX(NEQ),XT(NEQ),XTPT(NEQ),
     *   ELRT(MAXNINT*(6*2+3)+1,MAXNEL100,NEG100),
     *   ELRTPT(MAXNINT*(6*2+3)+1,MAXNEL100,NEG100)
      DIMENSION HE(NEN*3-NEN/2+3,NEN*3-NEN/2+3),RE(NEN*3-NEN/2+3),
     *   COORL(NEN,3),XP(NEN*3),T(NEN*3,NEN*3-NEN/2+3),
     *   TT(NEN*3-NEN/2+3,NEN*3),TEMP(NEN*3,NEN*3-NEN/2+3),
     *   HEP(NEN*3,NEN*3),REP(NEN*3),XTL(NEN,3),XTPTL(NEN,3),
     *   DXL(NEN*3),DEPS(6),DSIG(6),DTT(6,6,MAXNINT,MAXNEL100,NEG100)
C
      NEDN=3
      NLED=NEN*NEDN
      NGED=NLED-NEN/2+3
C
      IF (NEN.NE.6 .AND. NEN.NE.8) THEN
         WRITE(6,1000)
 1000    FORMAT('Invalid number of nodes for solid element. ',
     *      'Program terminated.')
         STOP
      ENDIF
      IF (IGO.EQ.1) GOTO 1 ! Input element data
      IF (IGO.EQ.2) GOTO 2 ! Determine initial element volumes
      IF (IGO.EQ.3) GOTO 3 ! calculate the tangent stiffness, H
      IF (IGO.EQ.4) GOTO 4 ! calculate RTPT and ELRTPT
      IF (IGO.EQ.5) GOTO 5 ! Print the system picture
      WRITE(6,1001)
 1001 FORMAT('Fatal error in subroutine SOLID1. Invalid value for IGO.')
      STOP
C
C Read and print element data.  Calculate assembly arrays and half-
C    bandwidth
C
    1 WRITE(6,1002) IEG
 1002 FORMAT(//,2X,'TOP LAYER SOLID ELEMENT GROUP NUMBER',I4,10X)
      WRITE(6,1003) NEL,NUMAT,NEN,NINT
 1003 FORMAT(/,4X,'NEL =',I5,4X,'NUMAT =',I3,4X,'NEN = ',I2,
     *   4X,'NINT = ',I2)
      WRITE(6,1004)
 1004 FORMAT(/,4X,'MATERIAL/GEOMETRIC SETS',/,9X,'SET',14X,'E',
     *   4X,'POISSONS RATIO',10X,'YST')
      DO 100 I=1,NUMAT
      READ(5,*) E(I),PR(I),YST(I)
      WRITE(6,1005) I,E(I),PR(I),YST(I)
 1005 FORMAT(8X,I4,3X,E12.5,6X,E12.5,1X,E12.5)
  100 CONTINUE
C
C Generates vector MAT and matrix LM
      CALL LMGEN(LM,MAT,NEN,NEL)
      WRITE(6,1006)
 1006 FORMAT(/,4X,'MATERIAL SET NUMBERS AND CONNECTIVITY VECTORS',/,
     *   9X,'ELE',3X,'SET',5X,'NODE NUMBERS')
      DO 110 N=1,NEL
C
C Transfers nodal coordinates for an element from COOR to COORL.
      WRITE(6,1007) N,MAT(N),(LM(I,N),I=1,NEN)
 1007 FORMAT(8X,I4,2X,I4,5X,8(2X,I4))
  110 CONTINUE
C
C Transfers equation numbers for all elements from ID to IDL.
C   For elements having half their nodes in the top plane
      CALL LCLID(ID,LM,IDL,NEDN,NGED,NEN,NEL,NEQ,NNP)
C
C Calculate MBAND
      CALL BAND(IDL,NGED,NEL,MBAND)
      RETURN
C
C Set initial volumes for elements, initialize YFLAG, and
C    initialize DTT to the elastic stiffness matrix
C
    2 DTT = 0.Q0
```

```
C Add line below to iterate with the tangent stiffness
c      WRITE(9) DTT
       DO 200 N=1,NEL
       CALL SOLSTF(HEP,ELRT,ELRTPT,DXL,REP,COOR,LM(1,N),E(MAT(N)),
     *  PR(MAT(N)),YST(MAT(N)),NINT,N,NEN,0,MAXNEL100,NEG100)
 200   CONTINUE
       RETURN
C
C Calculate and assemble element matrices and vectors
C
    3  DO 300 N=1,NEL
c       if (idebug .eq. 1) write(11,3100) n,nel
c 3100 format('Calculating stiffness for solid element ',i5,' of ',i5)
       M=MAT(N)
       CALL SOLSTF(HEP,ELRT,ELRTPT,DXL,REP,COOR,LM(1,N),E(M),PR(M),
     *   YST(M),NINT,N,NEN,1,MAXNEL100,NEG100)
C
C Transfers nodal coordinates for an element from COOR to COORL.
       CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
C
C Define transformation matrix
C
       T=0.Q0
       TT=0.Q0
       TEMP=0.Q0
          T(1,1)=1.Q0
          T(2,2)=1.Q0
          T(4,3)=1.Q0
          T(5,4)=1.Q0
          T(7,5)=1.Q0
          T(8,6)=1.Q0
          T(10,7)=1.Q0
          T(11,8)=1.Q0
       IF (NEN.EQ.6) THEN
          T(12,9)=1.Q0
          T(13,10)=1.Q0
          T(14,11)=1.Q0
          T(15,12)=1.Q0
          T(16,13)=1.Q0
          T(17,14)=1.Q0
          T(18,15)=1.Q0
          T(3,16)=1.Q0
          T(3,17)=COORL(1,2)
          T(3,18)=-COORL(1,1)
          T(6,16)=1.Q0
          T(6,17)=COORL(2,2)
          T(6,18)=-COORL(2,1)
          T(9,16)=1.Q0
          T(9,17)=COORL(3,2)
          T(9,18)=-COORL(3,1)
       ENDIF
       IF (NEN.EQ.8) THEN
          T(13,9)=1.Q0
          T(14,10)=1.Q0
          T(15,11)=1.Q0
          T(16,12)=1.Q0
          T(17,13)=1.Q0
          T(18,14)=1.Q0
          T(19,15)=1.Q0
          T(20,16)=1.Q0
          T(21,17)=1.Q0
          T(22,18)=1.Q0
          T(23,19)=1.Q0
          T(24,20)=1.Q0
          T(3,21)=1.Q0
          T(3,22)=COORL(1,2)
          T(3,23)=-COORL(1,1)
          T(6,21)=1.Q0
          T(6,22)=COORL(2,2)
          T(6,23)=-COORL(2,1)
          T(9,21)=1.Q0
          T(9,22)=COORL(3,2)
          T(9,23)=-COORL(3,1)
          T(12,21)=1.Q0
          T(12,22)=COORL(4,2)
```

```
         T(12,23)=-COORL(4,1)
      ENDIF
      DO 310 I=1,NEN*3
         DO 320 J=1,NEN*3-NEN/2+3
            TT(J,I)=T(I,J)
  320    CONTINUE
  310 CONTINUE
C
C Transform to global coordinates
C
      TEMP=MATMUL(HEP,T)
      HE=MATMUL(TT,TEMP)
      CALL ASSMBL(H,F,HE,RE,IDL(1,N),NEQ,MBAND,NGED,1,0)
  300 CONTINUE
      RETURN
C
    4 DO 400 N=1,NEL
      M=MAT(N)
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
C
C Transfer displacements for an element from X to XL.
C   For elements having half their nodes in the top plane
      CALL LCLX(XT,COORL,IDL(1,N),XTL,DX,NEDN,NEN,NGED,NEQ,1)
      CALL LCLX(XTPT,COORL,IDL(1,N),XTPTL,DX,NEDN,NEN,NGED,NEQ,0)
      DO 410 I=1,NEN
         DXL(I*3-2) = XTPTL(I,1)-XTL(I,1)
         DXL(I*3-1) = XTPTL(I,2)-XTL(I,2)
         DXL(I*3)   = XTPTL(I,3)-XTL(I,3)
  410 CONTINUE
      CALL SOLSTF(HEP,ELRT,ELRTPT,DXL,REP,COOR,LM(1,N),E(M),PR(M),
     *   YST(M),NINT,N,NEN,2,MAXNEL100,NEG100)
C
C Define transformation matrix
C
      T=0.Q0
      TT=0.Q0
      TEMP=0.Q0
         T(1,1)=1.Q0
         T(2,2)=1.Q0
         T(4,3)=1.Q0
         T(5,4)=1.Q0
         T(7,5)=1.Q0
         T(8,6)=1.Q0
         T(10,7)=1.Q0
         T(11,8)=1.Q0
      IF (NEN.EQ.6) THEN
         T(12,9)=1.Q0
         T(13,10)=1.Q0
         T(14,11)=1.Q0
         T(15,12)=1.Q0
         T(16,13)=1.Q0
         T(17,14)=1.Q0
         T(18,15)=1.Q0
         T(3,16)=1.Q0
         T(3,17)=COORL(1,2)
         T(3,18)=-COORL(1,1)
         T(6,16)=1.Q0
         T(6,17)=COORL(2,2)
         T(6,18)=-COORL(2,1)
         T(9,16)=1.Q0
         T(9,17)=COORL(3,2)
         T(9,18)=-COORL(3,1)
      ENDIF
      IF (NEN.EQ.8) THEN
         T(13,9)=1.Q0
         T(14,10)=1.Q0
         T(15,11)=1.Q0
         T(16,12)=1.Q0
         T(17,13)=1.Q0
         T(18,14)=1.Q0
         T(19,15)=1.Q0
         T(20,16)=1.Q0
         T(21,17)=1.Q0
         T(22,18)=1.Q0
```

```
         T(23,19)=1.Q0
         T(24,20)=1.Q0
         T(3,21)=1.Q0
         T(3,22)=COORL(1,2)
         T(3,23)=-COORL(1,1)
         T(6,21)=1.Q0
         T(6,22)=COORL(2,2)
         T(6,23)=-COORL(2,1)
         T(9,21)=1.Q0
         T(9,22)=COORL(3,2)
         T(9,23)=-COORL(3,1)
         T(12,21)=1.Q0
         T(12,22)=COORL(4,2)
         T(12,23)=-COORL(4,1)
      ENDIF
      DO 420 I=1,NEN*3
         DO 430 J=1,NEN*3-NEN/2+3
            TT(J,I)=T(I,J)
  430    CONTINUE
  420 CONTINUE
C
C Transform to global coordinates
C
      RE=MATMUL(TT,REP)
      CALL ASSMBL(H,RTPT,HE,RE,IDL(1,N),NEQ,MBAND,NGED,0,1)
  400 CONTINUE
      RETURN
C
    5 WRITE(6,5000) IEG
 5000 FORMAT(//,2X,'SOLID CONCRETE ELEMENT GROUP NUMBER ',I4)
      DO 500 N=1,NEL
      M=MAT(N)
      WRITE(6,5001) N,NEN,ELRTPT(MAXNINT*(6*2+3)+1,N,IEG)
 5001 FORMAT(/,4X,'ELEMENT NUMBER',I4,/,'NEN =',I2,/,'VOLUME = ',
     *   E12.5,//)
      IF (MAXNINT.EQ.1) THEN
         WRITE(6,5002)
 5002    FORMAT(2X,'STRESS VECTOR',4X,'STRAIN VECTOR')
         DO 510 I=1,6
         WRITE(6,5003) ELRTPT(I,N,IEG),ELRTPT(6+I,N,IEG)
 5003    FORMAT(3X,E12.5,5X,E12.5,5X,E12.5)
  510    CONTINUE
         WRITE(6,5004) ELRTPT(13,N,IEG)
 5004    FORMAT(/,2X,'PSI = ',E12.5)
      ENDIF
      IF (MAXNINT.EQ.4) THEN
         WRITE(6,5005)
 5005    FORMAT(4X,'STRESS VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4')
         DO 520 I=1,6
         WRITE(6,5006) ELRTPT(I,N,IEG),ELRTPT(6+I,N,IEG),
     *      ELRTPT(2*6+I,N,IEG),ELRTPT(3*6+I,N,IEG)
 5006    FORMAT(8(2X,E12.5))
  520    CONTINUE
         WRITE(6,5007)
 5007    FORMAT(//,4X,'STRAIN VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4')
         DO 530 I=1,6
         WRITE(6,5006) ELRTPT(4*6+I,N,IEG),ELRTPT(5*6+I,N,IEG),
     *      ELRTPT(6*6+I,N,IEG),ELRTPT(7*6+I,N,IEG)
  530    CONTINUE
        WRITE(6,5008)
 5008   FORMAT(//,4X,'FINAL VALUE OF PSI',/,6X,'INT PT 1',6X,'INT PT 2',
     *      6X,'INT PT 3',6X,'INT PT 4')
        WRITE(6,5006) ELRTPT(8*6+1,N,IEG),ELRTPT(8*6+2,N,IEG),
     *      ELRTPT(8*6+3,N,IEG),ELRTPT(8*6+4,N,IEG)
      ENDIF
      IF (MAXNINT.EQ.8) THEN
         WRITE(6,5009)
 5009    FORMAT(4X,'STRESS VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4',6X,'INT PT 5',6X,'INT PT 6',6X,
     *      'INT PT 7',6X,'INT PT 8')
         DO 540 I=1,6
         WRITE(6,5006) ELRTPT(I,N,IEG),ELRTPT(6+I,N,IEG),
     *      ELRTPT(2*6+I,N,IEG),ELRTPT(3*6+I,N,IEG),ELRTPT(4*6+I,N,IEG),
     *      ELRTPT(5*6+I,N,IEG),ELRTPT(6*6+I,N,IEG),ELRTPT(7*6+I,N,IEG)
```

```
  540    CONTINUE
         WRITE(6,5010)
 5010    FORMAT(//,4X,'STRAIN VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4',6X,'INT PT 5',6X,'INT PT 6',6X,
     *      'INT PT 7',6X,'INT PT 8')
         DO 550 I=1,6
         WRITE(6,5006) ELRTPT(8*6+I,N,IEG),ELRTPT(9*6+I,N,IEG),
     *      ELRTPT(10*6+I,N,IEG),ELRTPT(11*6+I,N,IEG),
     *      ELRTPT(12*6+I,N,IEG),ELRTPT(13*6+I,N,IEG),
     *      ELRTPT(14*6+I,N,IEG),ELRTPT(15*6+I,N,IEG)
  550    CONTINUE
        WRITE(6,5011)
 5011   FORMAT(//,4X,'FINAL VALUE OF PSI',/,6X,'INT PT 1',6X,'INT PT 2',
     *      6X,'INT PT 3',6X,'INT PT 4',6X,'INT PT 5',6X,'INT PT 6',6X,
     *      'INT PT 7',6X,'INT PT 8')
        WRITE(6,5006) ELRTPT(16*6+1,N,IEG),ELRTPT(16*6+2,N,IEG),
     *      ELRTPT(16*6+3,N,IEG),ELRTPT(16*6+4,N,IEG),
     *      ELRTPT(16*6+5,N,IEG),ELRTPT(16*6+6,N,IEG),
     *      ELRTPT(16*6+7,N,IEG),ELRTPT(16*6+8,N,IEG)
      ENDIF
  500 CONTINUE
      RETURN
      END

      SUBROUTINE SOLIDB(A,B,C)
C Sets up storage for subroutine SOLID10
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      COMMON /INFO/ N5TEMP
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NEN,NINT,NDUM(3)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      REAL(KIND=16), ALLOCATABLE, DIMENSION(:) :: BSUB
      DIMENSION A(N4-1),B(N5-1),C(N13-1)
C
      IF (MAXNEL110.LT.NEL) MAXNEL110=NEL
      IF (NINT.EQ.2) MAXNINT=8
      IF (NINT.EQ.3) MAXNINT=4
      N401=1   +NUMAT                    ! E
      N402=N401+NUMAT                    ! POISSON'S RATIO
      N403=N402+NUMAT                    ! f'c
      N404=N403+NEL                      ! MAT
      N405=N404+NEN*NEL                  ! LM
      N5TEMP=N405+NEN*3*NEL              ! IDL
      NEND=N5TEMP-1
      IF (IGO.EQ.1) THEN
         ALLOCATE(BSUB(NEND))
         BSUB=0.Q0
         CALL SOLID10(A(N1),A(N2),A(N3),BSUB(1),BSUB(N401),BSUB(N402),
     *      BSUB(N403),BSUB(N404),BSUB(N405),C(1),C(N6),C(N7),C(N8),
     *      C(N9),C(N10),C(N1104),C(N1204))
         WRITE(7) NEND,NELTYP,NEL,NUMAT,NEN,NINT,(NDUM(I),I=1,3),
     *      (BSUB(I),I=1,NEND)
         DEALLOCATE(BSUB)
      ELSE
         CALL SOLID10(A(N1),A(N2),A(N3),B(1),B(N401),B(N402),B(N403),
     *      B(N404),B(N405),C(1),C(N6),C(N7),C(N8),C(N9),C(N10),
     *      C(N1104),C(N1204))
      ENDIF
      RETURN
      END

      SUBROUTINE SOLID10(COOR,ID,R,E,PR,YST,MAT,LM,IDL,H,F,RTPT,DX,XT,
     *    XTPT,ELRT,ELRTPT)
C Performs computations for 6 or 8 node interior solid elements.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NEN,NINT,NDUM(3)
      COMMON /DEBUG/ IDEBUG
```

```
      DIMENSION COOR(NNP,3),ID(NNP,3),R(NNP,3),E(NUMAT),PR(NUMAT),
     *   YST(NUMAT),MAT(NEL),LM(NEN,NEL),H(NEQ,MBAND),F(NEQ),RTPT(NEQ),
     *   IDL(3,NEN,NEL),DX(NEQ),XT(NEQ),XTPT(NEQ),
     *   ELRT(MAXNINT*(6*2+3)+1,MAXNEL110,NEG110),
     *   ELRTPT(MAXNINT*(6*2+3)+1,MAXNEL110,NEG110)
      DIMENSION HE(NEN*3,NEN*3),RE(NEN*3),COORL(NEN,3),XP(NEN*3),
     *   XTL(NEN,3),XTPTL(NEN,3),DXL(NEN*3),DEPS(6),DSIG(6),
     *   DTT(6,6,MAXNINT,MAXNEL110,NEG110)
C
      NEDN=3
      NED=NEN*NEDN
C
      IF (NEN.NE.6 .AND. NEN.NE.8) THEN
         WRITE(6,1000)
 1000    FORMAT('Invalid number of nodes for interior solid element. ',
     *      'Program terminated.')
         STOP
      ENDIF
      IF (IGO.EQ.1) GOTO 1 ! Input element data
      IF (IGO.EQ.2) GOTO 2 ! Determine initial element volumes
      IF (IGO.EQ.3) GOTO 3 ! calculate the tangent stiffness, H
      IF (IGO.EQ.4) GOTO 4 ! calculate RTPT and ELRTPT
      IF (IGO.EQ.5) GOTO 5 ! Print the system picture
      WRITE(6,1001)
 1001 FORMAT('Fatal error in subroutine SOLID10. '
     *   'Invalid value for IGO.')
      STOP
C
C Read and print element data.  Calculate assembly arrays and half-
C    bandwidth
C
    1 WRITE(6,1002) IEG
 1002 FORMAT(//,2X,'INTERIOR SOLID ELEMENT GROUP NUMBER',I4,10X)
      WRITE(6,1003) NEL,NUMAT,NEN,NINT
 1003 FORMAT(/,4X,'NEL =',I5,4X,'NUMAT =',I3,4X,'NEN = ',I2,
     *   4X,'NINT = ',I2)
      WRITE(6,1004)
 1004 FORMAT(/,4X,'MATERIAL/GEOMETRIC SETS',/,9X,'SET',14X,'E',
     *   4X,'POISSONS RATIO',10X,'YST')
      DO 100 I=1,NUMAT
      READ(5,*) E(I),PR(I),YST(I)
      WRITE(6,1005) I,E(I),PR(I),YST(I)
 1005 FORMAT(8X,I4,3X,E12.5,6X,E12.5,1X,E12.5)
  100 CONTINUE
C
C Generates vector MAT and matrix LM
      CALL LMGEN(LM,MAT,NEN,NEL)
      WRITE(6,1006)
 1006 FORMAT(/,4X,'MATERIAL SET NUMBERS AND CONNECTIVITY VECTORS',/,
     *   9X,'ELE',3X,'SET',5X,'NODE NUMBERS')
      DO 110 N=1,NEL
C
C Transfers nodal coordinates for an element from COOR to COORL.
      WRITE(6,1007) N,MAT(N),(LM(I,N),I=1,NEN)
 1007 FORMAT(8X,I4,2X,I4,5X,8(2X,I4))
  110 CONTINUE
C
C Transfers equation numbers for all elements from ID to IDL.
C   For elements which do not lie in the top plane
      CALL LCLID2(ID,LM,IDL,NNP,NEDN,NEN,NEL)
C
C Calculate MBAND
      CALL BAND(IDL,NED,NEL,MBAND)
      RETURN
C
C Set initial volumes for elements, initialize YFLAG, and
C    initialize DTT to the elastic stiffness matrix
C
    2 DTT = 0.Q0
C Add line below to iterate with the tangent stiffness
c      WRITE(9) DTT
      DO 200 N=1,NEL
      CALL SOLSTF(HE,ELRT,ELRTPT,DXL,RE,COOR,LM(1,N),E(MAT(N)),
     *  PR(MAT(N)),YST(MAT(N)),NINT,N,NEN,0,MAXNEL110,NEG110)
 200  CONTINUE
```

```
      RETURN
C
C Calculate and assemble element matrices and vectors
C
    3 DO 300 N=1,NEL
c      if (idebug .eq. 1) write(11,3100) n,nel
c 3100 format('Calculating stiffness for solid element ',i5,' of ',i5)
      M=MAT(N)
      CALL SOLSTF(HE,ELRT,ELRTPT,DXL,RE,COOR,LM(1,N),E(M),PR(M),
     *   YST(M),NINT,N,NEN,1,MAXNEL110,NEG110)
      CALL ASSMBL(H,F,HE,RE,IDL(1,1,N),NEQ,MBAND,NED,1,0)
  300 CONTINUE
      RETURN
C
    4 DO 400 N=1,NEL
c      if (idebug .eq. 1) write(11,4100) n,nel
c 4100 format('Calculating forces for interior solid element ',i5,
c     *  ' of ',i5)
      M=MAT(N)
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
C
C Transfer displacements for an element from X to XL.
C   For elements which do not lie in the top plane
      CALL LCLX2(XT,IDL(1,1,N),XTL,NEDN,NEN,NED,NEQ)
      CALL LCLX2(XTPT,IDL(1,1,N),XTPTL,NEDN,NEN,NED,NEQ)
      DO 410 I=1,NEN
         DXL(I*3-2) = XTPTL(I,1)-XTL(I,1)
         DXL(I*3-1) = XTPTL(I,2)-XTL(I,2)
         DXL(I*3)   = XTPTL(I,3)-XTL(I,3)
  410 CONTINUE
      CALL SOLSTF(HE,ELRT,ELRTPT,DXL,RE,COOR,LM(1,N),E(M),PR(M),
     *   YST(M),NINT,N,NEN,2,MAXNEL110,NEG110)
      CALL ASSMBL(H,RTPT,HE,RE,IDL(1,1,N),NEQ,MBAND,NED,0,1)
  400 CONTINUE
      RETURN
C
    5 WRITE(6,5000) IEG
 5000 FORMAT(//,2X,'INTERIOR SOLID CONCRETE ELEMENT GROUP NUMBER ',I4)
      DO 500 N=1,NEL
      M=MAT(N)
      WRITE(6,5001) N,NEN,ELRTPT(MAXNINT*(6*2+3)+1,N,IEG)
 5001 FORMAT(/,4X,'ELEMENT NUMBER',I4,/,'NEN =',I2,/,'VOLUME = ',
     *   E12.5,//)
      IF (MAXNINT.EQ.1) THEN
         WRITE(6,5002)
 5002    FORMAT(2X,'STRESS VECTOR',4X,'STRAIN VECTOR')
         DO 510 I=1,6
         WRITE(6,5003) ELRTPT(I,N,IEG),ELRTPT(6+I,N,IEG)
 5003    FORMAT(3X,E12.5,5X,E12.5,5X,E12.5)
  510    CONTINUE
         WRITE(6,5004) ELRTPT(13,N,IEG)
 5004    FORMAT(/,2X,'PSI = ',E12.5)
      ENDIF
      IF (MAXNINT.EQ.4) THEN
         WRITE(6,5005)
 5005    FORMAT(4X,'STRESS VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4')
         DO 520 I=1,6
         WRITE(6,5006) ELRTPT(I,N,IEG),ELRTPT(6+I,N,IEG),
     *      ELRTPT(2*6+I,N,IEG),ELRTPT(3*6+I,N,IEG)
 5006    FORMAT(8(2X,E12.5))
  520    CONTINUE
         WRITE(6,5007)
 5007    FORMAT(//,4X,'STRAIN VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4')
         DO 530 I=1,6
         WRITE(6,5006) ELRTPT(4*6+I,N,IEG),ELRTPT(5*6+I,N,IEG),
     *      ELRTPT(6*6+I,N,IEG),ELRTPT(7*6+I,N,IEG)
  530    CONTINUE
        WRITE(6,5008)
 5008   FORMAT(//,4X,'FINAL VALUE OF PSI',/,6X,'INT PT 1',6X,'INT PT 2',
     *      6X,'INT PT 3',6X,'INT PT 4')
        WRITE(6,5006) ELRTPT(8*6+1,N,IEG),ELRTPT(8*6+2,N,IEG),
```

```
     *      ELRTPT(8*6+3,N,IEG),ELRTPT(8*6+4,N,IEG)
            ENDIF
            IF (MAXNINT.EQ.8) THEN
               WRITE(6,5009)
 5009       FORMAT(4X,'STRESS VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4',6X,'INT PT 5',6X,'INT PT 6',6X,
     *      'INT PT 7',6X,'INT PT 8')
               DO 540 I=1,6
               WRITE(6,5006) ELRTPT(I,N,IEG),ELRTPT(6+I,N,IEG),
     *      ELRTPT(2*6+I,N,IEG),ELRTPT(3*6+I,N,IEG),ELRTPT(4*6+I,N,IEG),
     *      ELRTPT(5*6+I,N,IEG),ELRTPT(6*6+I,N,IEG),ELRTPT(7*6+I,N,IEG)
  540       CONTINUE
               WRITE(6,5010)
 5010       FORMAT(//,4X,'STRAIN VECTORS',/,6X,'INT PT 1',6X,'INT PT 2',6X,
     *      'INT PT 3',6X,'INT PT 4',6X,'INT PT 5',6X,'INT PT 6',6X,
     *      'INT PT 7',6X,'INT PT 8')
               DO 550 I=1,6
               WRITE(6,5006) ELRTPT(8*6+I,N,IEG),ELRTPT(9*6+I,N,IEG),
     *      ELRTPT(10*6+I,N,IEG),ELRTPT(11*6+I,N,IEG),
     *      ELRTPT(12*6+I,N,IEG),ELRTPT(13*6+I,N,IEG),
     *      ELRTPT(14*6+I,N,IEG),ELRTPT(15*6+I,N,IEG)
  550       CONTINUE
              WRITE(6,5011)
 5011       FORMAT(//,4X,'FINAL VALUE OF PSI',/,6X,'INT PT 1',6X,'INT PT 2',
     *      6X,'INT PT 3',6X,'INT PT 4',6X,'INT PT 5',6X,'INT PT 6',6X,
     *      'INT PT 7',6X,'INT PT 8')
              WRITE(6,5006) ELRTPT(16*6+1,N,IEG),ELRTPT(16*6+2,N,IEG),
     *      ELRTPT(16*6+3,N,IEG),ELRTPT(16*6+4,N,IEG),
     *      ELRTPT(16*6+5,N,IEG),ELRTPT(16*6+6,N,IEG),
     *      ELRTPT(16*6+7,N,IEG),ELRTPT(16*6+8,N,IEG)
            ENDIF
  500    CONTINUE
            RETURN
            END

            SUBROUTINE SOLSTF(STF,ELRT,ELRTPT,DXL,REP,COOR,LM,EE,PR,YST,
     *      NINT,IJK,NEN,IFLAG,MAXNEL,NEG)
            IMPLICIT REAL(KIND=16) (A-H,O-Z)
            IMPLICIT INTEGER(KIND=8) (I-N)
            COMMON /INFO1/ NNP,NEQ,MBAND,IGO
            COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     *      MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
            COMMON /DEBUG/ IDEBUG
            COMMON /TOL/ TOLF,TOLM,NITMAX
            COMMON /MISC/ PSIMAX,KAPPA
            DIMENSION STF(NEN*3,NEN*3),DXL(NEN*3),REP(NEN*3),COOR(NNP,3),
     *      ELRT(MAXNINT*(6*2+3)+1,MAXNEL,NEG),LM(NEN),
     *      ELRTPT(MAXNINT*(6*2+3)+1,MAXNEL,NEG)
            DIMENSION RI(25),SI(25),TI(25),WI(25),BT(NEN*3,6),B(6,NEN*3),
     *      P(4,NEN),DP(3,NEN),XJ(3,3),XI(3,3),D(6,6),DT(6,6),DINV(6,6),
     *      DEPS(6),DSIG(6),DF(6,1),DQ(6,1),DFT(1,6),TEMP(6,1),DEPSP(6),
     *      DTT(6,6,MAXNINT,MAXNEL,NEG)
            REAL(KIND=16) PSIMAX,KAPPA,PSI,PHI,F,ELAS,DEN,X,R,THETA,J2,J3,
     *      NDEPSP,TEMPPSI,D1,DET,VOL,PF
            INTEGER(KIND=8) FINT

C
C If IFLAG = 0 -> Calculate the element volume and initialize YFLAG.
C If IFLAG = 1 -> Calculate the stiffness matrix
C If IFLAG = 2 -> Calculate the stress, strain, and force vector
C
C RI,SI,TI AND WI CONTAIN THE R,S,T COORDINATES AND WEIGHTS, RESPECTIVELY, OF
C THE INTEGRATION POINTS FOR THE TRIANGULAR PRISM AND RECTANGULAR PRISM
C ELEMENTS IN THE FOLLOWING ORDER:  1 POINT (TRIANGULAR PRISM, NINT=1),
C 1 POINT (RECTANGULAR PRISM, NINT=1), 8 PTS (TP,4x2), 8 PTS (RP,2x2x2),
C 3 PTS (TP,3x1), 4 PTS (RP,2x2x1)
C
            RI(1) = 1.Q0/3.Q0
            RI(2) = 0.Q0
            RI(3) = RI(1)
            RI(4) = RI(1)
            RI(5) = 0.6Q0
            RI(6) = 0.2Q0
            RI(7) = 0.2Q0
            RI(8) = 0.6Q0
            RI(9) = 0.2Q0
```

```
      RI(10) = 0.2Q0
      RI(11) = -1.Q0/QSQRT(3.Q0)
      RI(12) = RI(11)
      RI(13) = 1.Q0/QSQRT(3.Q0)
      RI(14) = RI(13)
      RI(15) = RI(11)
      RI(16) = RI(11)
      RI(17) = RI(13)
      RI(18) = RI(13)
      RI(19) = 0.5Q0
      RI(20) = 0.Q0
      RI(21) = RI(19)
      RI(22) = RI(11)
      RI(23) = RI(11)
      RI(24) = RI(13)
      RI(25) = RI(13)
C
      SI(1) = RI(1)
      SI(2) = 0.Q0
      SI(3) = RI(1)
      SI(4) = RI(1)
      SI(5) = 0.2Q0
      SI(6) = 0.6Q0
      SI(7) = 0.2Q0
      SI(8) = 0.2Q0
      SI(9) = 0.6Q0
      SI(10) = 0.2Q0
      SI(11) = RI(13)
      SI(12) = RI(11)
      SI(13) = RI(11)
      SI(14) = RI(13)
      SI(15) = RI(13)
      SI(16) = RI(11)
      SI(17) = RI(11)
      SI(18) = RI(13)
      SI(19) = RI(19)
      SI(20) = RI(19)
      SI(21) = 0.Q0
      SI(22) = RI(13)
      SI(23) = RI(11)
      SI(24) = RI(11)
      SI(25) = RI(13)
C
      TI = 0.Q0
      TI(3) = RI(13)
      TI(4) = RI(11)
      TI(5) = RI(13)
      TI(6) = RI(13)
      TI(7) = RI(13)
      TI(8) = RI(11)
      TI(9) = RI(11)
      TI(10) = RI(11)
      TI(11) = RI(13)
      TI(12) = RI(13)
      TI(13) = RI(13)
      TI(14) = RI(13)
      TI(15) = RI(11)
      TI(16) = RI(11)
      TI(17) = RI(11)
      TI(18) = RI(11)
C
      WI = 1.Q0
      WI(2) = 8.Q0
      WI(3) = -9.Q0/32.Q0
      WI(4) = WI(3)
      WI(5) = 25.Q0/96.Q0
      WI(6) = WI(5)
      WI(7) = WI(5)
      WI(8) = WI(5)
      WI(9) = WI(5)
      WI(10) = WI(5)
      WI(19) = RI(1)
      WI(20) = RI(1)
      WI(21) = RI(1)
      WI(22) = 2.Q0
      WI(23) = WI(22)
```

```
      WI(24) = WI(22)
      WI(25) = WI(22)
C
C D is the elastic stiffness matrix
C
      C11 = EE*(1.Q0-PR)/((1.Q0+PR)*(1.Q0-2.Q0*PR))
      C22 = EE*PR/((1.Q0+PR)*(1.Q0-2.Q0*PR))
      C33 = EE/(2.Q0*(1.Q0+PR))
      D = 0.Q0
      D(1,1) = C11
      D(1,2) = C22
      D(1,3) = C22
      D(2,1) = C22
      D(2,2) = C11
      D(2,3) = C22
      D(3,1) = C22
      D(3,2) = C22
      D(3,3) = C11
      D(4,4) = C33
      D(5,5) = C33
      D(6,6) = C33
C
C Calculate the inverse of the elastic material matrix
      C4 = (C11**2.Q0+C11*C22-2.Q0*C22**2.Q0)
      C1 = (C11+C22)/C4
      C2 = -C22/C4
      C3 = 1.Q0/C33
      DINV = 0.Q0
      DINV(1,1) = C1
      DINV(1,2) = C2
      DINV(1,3) = C2
      DINV(2,1) = C2
      DINV(2,2) = C1
      DINV(2,3) = C2
      DINV(3,1) = C2
      DINV(3,2) = C2
      DINV(3,3) = C1
      DINV(4,4) = C3
      DINV(5,5) = C3
      DINV(6,6) = C3
C
C
      IF (IFLAG.EQ.0) VOL = 0.Q0
      IF (IFLAG.EQ.1) STF = 0.Q0
      IF (IFLAG.EQ.2) REP = 0.Q0
C
C  NINT=INTEGRATION SCHEME: 1 OR 2 POINT.
      IF ((NEN.EQ.6).AND.(NINT.EQ.1)) L1=1
      IF ((NEN.EQ.6).AND.(NINT.EQ.1)) L2=1
      IF ((NEN.EQ.8).AND.(NINT.EQ.1)) L1=2
      IF ((NEN.EQ.8).AND.(NINT.EQ.1)) L2=2
      IF ((NEN.EQ.6).AND.(NINT.EQ.2)) L1=3
      IF ((NEN.EQ.6).AND.(NINT.EQ.2)) L2=10
      IF ((NEN.EQ.8).AND.(NINT.EQ.2)) L1=11
      IF ((NEN.EQ.8).AND.(NINT.EQ.2)) L2=18
      IF ((NEN.EQ.6).AND.(NINT.EQ.3)) L1=19
      IF ((NEN.EQ.6).AND.(NINT.EQ.3)) L2=21
      IF ((NEN.EQ.8).AND.(NINT.EQ.3)) L1=22
      IF ((NEN.EQ.8).AND.(NINT.EQ.3)) L2=25
C  L1 TO L2 ARE THE GAUSS QUADRATURE POINT LOCATIONS.
C
      DO 100 L=L1,L2
C  L LOOPS OVER THE INTEGRATION POINTS
      P = 0.Q0
      IF (NEN.EQ.6) CALL TRI  (P,RI(L),SI(L),TI(L))
      IF (NEN.EQ.8) CALL RECT (P,RI(L),SI(L),TI(L))
      CALL JACCOMP(COOR,LM,XJ,XI,P,DP,DET,NNP,NEN,IJK)
      D1 = WI(L)*DET
      ELRTPT(MAXNINT*(6*2+2)+L-L1+1,IJK,IEG) = D1
      IF (IFLAG.EQ.0) THEN
            VOL = VOL + D1
            GO TO 100
      ENDIF
      B = 0.Q0
      DO 130 J=1,NEN
      J1 = 3*(J-1) + 1
```

```
      J2 = J1 + 1
      J3 = J1 + 2
      B(1,J1) = DP(1,J)
      B(2,J2) = DP(2,J)
      B(3,J3) = DP(3,J)
      B(4,J1) = DP(2,J)
      B(4,J2) = DP(1,J)
      B(5,J2) = DP(3,J)
      B(5,J3) = DP(2,J)
      B(6,J1) = DP(3,J)
      B(6,J3) = DP(1,J)
  130 CONTINUE
C B IS THE NODAL DISPLACEMENT TO STRAIN (X,Y,Z AXES) TRANSFORMATION MATRIX AT
C THE CURRENT INTEGRATION POINT.
C
      BT = 0.Q0
      DO 140 I=1,6
         DO 150 J=1,NEN*3
            BT(J,I)=B(I,J)
  150    CONTINUE
  140 CONTINUE
C BT IS THE TRANSPOSE OF B AT THE CURRENT INTEGRATION POINT.
C
      IF (IFLAG.EQ.1) THEN
         B = MATMUL(D,B)
         STF = STF + D1*MATMUL(BT,B)
         GO TO 100
      ENDIF
C THE CONTRIBUTION TO THE PORTIONS OF THE ROWS OF THE STIFFNESS MATRIX
C AT THE CURRENT INTEGRATION POINT HAS BEEN ADDED INTO STF.
C
C Define constants here
      OMEGA = 0.5Q0
      PHI = 0.5Q0
      ALPHA = 10.0Q0 !0.6Q1*5.Q3/YST
      GAMMA = 1.72Q0
      PSIMAX = 0.6Q-3
      FINT = 10
      TOLLS = 0.000001Q0
      KAPPA=0.6q0
      PSI = ELRT(MAXNINT*12+L-L1+1,IJK,IEG)
C
C Below two lines require beta as a function of psi
      BETA = ((PSI/PSIMAX)**KAPPA)
     *      *QEXP(1.Q0-((PSI/PSIMAX)**KAPPA))
      DBETADPSI = KAPPA*QEXP(1.Q0-((PSI/PSIMAX)**KAPPA))
     *  * (1.Q0 - ((PSI/PSIMAX)**KAPPA))
     *  / (PSIMAX*((PSI/PSIMAX)**(1.q0-KAPPA)))
C
C CE 108 pg. 9-19 step a. Calculate strain increment
      DEPS = MATMUL(B,DXL)
C
C CE 108 pg. 9-19 step b. Calculate stress increment assuming elastic behavior
      DSIG = MATMUL(D,DEPS)
C
C CE 108 pg. 9-19 step c. Compute current stress/strain state
      DO 200 I=1,6
         NSIG = (L-L1+1)*6-6+I
         NEPS = MAXNINT*6+(L-L1+1)*6-6+I
         ELRTPT(NSIG,IJK,IEG) = ELRT(NSIG,IJK,IEG)+DSIG(I)
         ELRTPT(NEPS,IJK,IEG) = ELRT(NEPS,IJK,IEG)+DEPS(I)
  200 CONTINUE
C
C Add this line to remove plasticity
c      GO TO 1
C
C CE 108 pg. 9-19 step d. Compute F and determine stress state
      CALL FCALC(F,1.Q0,ELRT((L-L1+1)*6-5,IJK,IEG),DSIG,
     *      X,R,THETA,PSI,PSIMAX,BETA,YST)
C
C If the current stress is below the loading surface and was not previously cracked/crushed
C  set loading type to elastic and tangent matrix to the elastic matrix
      IF (F .LT. 0.Q0 .AND.
     *     ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .GE. -5.Q0) THEN
          ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = 0.Q0
```

```
C            IF (F .GT. -TOLLS) ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = 10.Q0
             DT = D
             GO TO 1
      ENDIF
C
C If in axial tension or previously in axial tension, apply plane stress
       IF (ELRTPT((L-L1+1)*6-3,IJK,IEG) .GT. 0.Q0 .OR.
     *  (ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .LE. -5.Q0 .AND.
     *  ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .GE. -15.Q0)) THEN
             ELRTPT((L-L1+1)*6-3,IJK,IEG) = 0.Q0
             ELRTPT((L-L1+1)*6-1,IJK,IEG) = 0.Q0
             ELRTPT((L-L1+1)*6,IJK,IEG) = 0.Q0
             ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = -10.Q0
             C11 = EE/(1.Q0-PR**2.Q0)
             C22 = C22*PR
             C33 = C11*(1.Q0-PR)/2.Q0
             DT = 0.Q0
             DT(1,1) = C11
             DT(1,2) = C22
             DT(2,1) = C22
             DT(2,2) = C11
             DT(4,4) = C33
             DSIG = MATMUL(DT,DEPS)
             ELRTPT((L-L1+1)*6-5,IJK,IEG) = ELRT((L-L1+1)*6-5,IJK,IEG)
     *           +DSIG(1)
             ELRTPT((L-L1+1)*6-4,IJK,IEG) = ELRT((L-L1+1)*6-4,IJK,IEG)
     *           +DSIG(2)
             ELRTPT((L-L1+1)*6-2,IJK,IEG) = ELRT((L-L1+1)*6-2,IJK,IEG)
     *           +DSIG(4)
             GO TO 1
         ENDIF
C
C If concrete was previously in net tension => completely cracked so zero out stress
         IF (ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .LT. -15.Q0) THEN
             DO 205 J=1,6
                NSIG = (L-L1+1)*6-6+J
                ELRTPT(NSIG,IJK,IEG) = 0.Q0
  205        CONTINUE
             ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = -20.Q0
             DT=0.Q0
             GO TO 1
         ENDIF
C
C If you are at the origin and have not previously loaded the specimen at all,
C   assign elastic loading and skip plasticity.
       IF (F .EQ. 0.Q0 .AND.
      *  ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .GE. -1.Q0) THEN
             IF (X .EQ. 0.Q0 .AND.
     *         ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .LT. 5.Q0
     *           .AND. ELRT(MAXNINT*12+L-L1+1,IJK,IEG) .EQ. 0.Q0) THEN
                ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = 0.Q0
                DT = D
                GO TO 1
             ENDIF
       ENDIF
C
C CE 108 pg. 9-19 step e.  Determine ELAS based on previous stress state
       IF (ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .GT. 5.Q0 .OR.
     *       ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .LT. -5.Q0) THEN
             ELAS = 0.Q0
       ELSE
             CALL ZBRENT(0.Q0,1.Q0,ELAS,TOLLS,
     *           ELRT((L-L1+1)*6-5,IJK,IEG),DSIG,X,R,THETA,PSI,
     *           PSIMAX,BETA,YST)
             ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = 10.Q0
             IF (ELAS .EQ. 0.Q0 .AND.
     *         ELRT(MAXNINT*12+L-L1+1,IJK,IEG) .EQ. 0.Q0) THEN
                CALL FCALC(PF,0.Q0,ELRT((L-L1+1)*6-5,IJK,IEG),DSIG,
     *               X,R,THETA,PSI,PSIMAX,BETA,YST)
                IF (PF .LE. 0.Q0) THEN
                    ELAS = 1.Q0
                    ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = 0.Q0
                ENDIF
             ENDIF
       ENDIF
```

```
C
C CE 108 pg. 9-19 step f.  Push current stress state to current loading surface
      DO 210 I=1,6
         NSIG = (L-L1+1)*6-6+I
         ELRTPT(NSIG,IJK,IEG) = ELRT(NSIG,IJK,IEG)+ELAS*DSIG(I)
  210 CONTINUE
      IF (ELAS .EQ. 1.) THEN
        ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = 10.Q0
        GO TO 1
      ENDIF
      DEPS = DEPS*(1.Q0-ELAS)
      DEPS = DEPS/FINT
C
C CE 108 pg. 9-19 step g. Calculate elastic and plastic stress
      DO 220 I=1,FINT
            CALL FCALC(F,0.Q0,ELRTPT((L-L1+1)*6-5,IJK,IEG),DSIG,
     *          X,R,THETA,PSI,PSIMAX,BETA,YST)
C If in tension, zero out stresses and set flag
            IF (X .GT. 0.) THEN
                DO 230 J=1,6
                    NSIG = (L-L1+1)*6-6+J
                    ELRTPT(NSIG,IJK,IEG) = 0.Q0
  230           CONTINUE
                    ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG)= -20.Q0
                GO TO 1
            ENDIF
            NSIG = (L-L1+1)*6-6
            SXX = ELRTPT(NSIG+1,IJK,IEG)
            SYY = ELRTPT(NSIG+2,IJK,IEG)
            SZZ = ELRTPT(NSIG+3,IJK,IEG)
            SXY = ELRTPT(NSIG+4,IJK,IEG)
            SYZ = ELRTPT(NSIG+5,IJK,IEG)
            SXZ = ELRTPT(NSIG+6,IJK,IEG)
            CALL DFCALC(DF,DQ,X,R,THETA,PSIMAX,ALPHA,GAMMA,ETA,PHI,
     *          OMEGA,PSI,BETA,DBETADPSI,YST,DFDEP,SXX,SYY,SZZ,SXY,SYZ,
     *          SXZ)
            DO 240 J=1,6
                DFT(1,J) = DF(J,1)
  240       CONTINUE
C
C Calculate DT = the tangent material matrix
C   DEN is the denominator (scalar) of the DT expression
            TEMP = 0.Q0
            TEMP(4,1) = DQ(4,1)
            TEMP(5,1) = DQ(5,1)
            TEMP(6,1) = DQ(6,1)
            DEN = 0.Q0
            DO 250 J=1,6
                DEN = DEN + (TEMP(J,1)+DQ(J,1))*DQ(J,1)
  250       CONTINUE
            DEN = 2.Q0/3.Q0*DEN
            DEN = QSQRT(DEN)
            DEN = -DFDEP*DEN
            TEMP = 0.Q0
            TEMP = MATMUL(D,DQ)
            DO 260 J=1,6
                DEN = DEN + DF(J,1)*TEMP(J,1)
  260       CONTINUE
            DFT = MATMUL(DFT,D)
            DT = MATMUL(DQ,DFT)
            DT = MATMUL(D,DT)
            DT = DT/DEN
            DT = D - DT
            IF (DEN .EQ. 0.) THEN
                IF (X .EQ. 0. .AND.
     *            ELRT(MAXNINT*12+L-L1+1,IJK,IEG) .EQ. 0.Q0) THEN
                    DT = D
                    ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG)=0.Q0
                ELSE
                    DT = 0.Q0
                    ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG)=10.Q0
                END IF
            END IF
C  Compute the new stress for the given DT and increment in strain
            DSIG = MATMUL(DT,DEPS)
```

```
            DO 280 K = 1,6
                TEMP(K,1) = DSIG(K)
  280       CONTINUE
            IF (DEN .EQ. 0. .AND. X .EQ. 0.) THEN
                DSIG = FINT*DSIG
                DO 290 J=1,6
                    NSIG = (L-L1+1)*6-6+J
                    ELRTPT(NSIG,IJK,IEG) = ELRTPT(NSIG,IJK,IEG)+DSIG(J)
  290           CONTINUE
                GO TO 1
            ENDIF
C
C Set up iteration to return stress state to yield surface
            DO 300 J=1,NITMAX
                DEPSP = DEPS - MATMUL(DINV,DSIG)
                NDEPSP = 0.Q0
                DO 320 K=1,3
                    NDEPSP = NDEPSP + DEPSP(K)*DEPSP(K)
  320           CONTINUE
                DO 330 K=4,6
                    NDEPSP = NDEPSP + 2.Q0 * DEPSP(K)*DEPSP(K)
  330           CONTINUE
                NDEPSP = QSQRT(2.Q0/3.Q0*NDEPSP)
                AVEX = X/2.Q0 + (ELRTPT((L-L1+1)*6-5,IJK,IEG)
     *              + ELRTPT((L-L1+1)*6-4,IJK,IEG) + DSIG(1)
     *              + ELRTPT((L-L1+1)*6-3,IJK,IEG) + DSIG(2)
     *              + DSIG(3))/QSQRT(3.Q0)/YST/2.Q0
C
C Vary Psi relationship here
C
               IF (PSI .GE. PSIMAX) THEN
                    TEMPPSI = PSI + NDEPSP/
     *                  (PHI + ALPHA * (QABS(AVEX))**GAMMA)
               ELSE
                    TEMPPSI = PSI + NDEPSP/
     *                  (PHI + ALPHA * (QABS(AVEX))**GAMMA)
               ENDIF
C
C Below two lines require beta as a function of psi
                BETA = ((TEMPPSI/PSIMAX)**KAPPA)
     *              *QEXP(1.Q0-((TEMPPSI/PSIMAX)**KAPPA))
                DBETADPSI = KAPPA*QEXP(1.Q0-((TEMPPSI/PSIMAX)**KAPPA))
     *              * (1.Q0 - ((TEMPPSI/PSIMAX)**KAPPA))
     *              / (PSIMAX*((TEMPPSI/PSIMAX)**(1.q0-KAPPA)))
                IF (ELRT(MAXNINT*13+L-L1+1,IJK,IEG) .EQ. 0.) GO TO 2
                IF (PSI .LT. 1Q-5) GO TO 2
                CALL FCALC(F,1.Q0,ELRTPT((L-L1+1)*6-5,IJK,IEG),
     *              DSIG,X,R,THETA,TEMPPSI,PSIMAX,BETA,YST)
C If in axial compression but still net tension the concrete
C   must be cracked/crushed so zero out stresses
                IF (X .GT. 0.Q0) THEN
                    if (elrtpt((l-l1+1)*6-3,ijk,ieg) .le. 0.q0) then
                        DO 340 K=1,6
                            NSIG = (L-L1+1)*6-6+K
                            ELRTPT(NSIG,IJK,IEG) = 0.Q0
  340                   CONTINUE
                        ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = -20.Q0
                        DT=0.Q0
                       GO TO 1
                    ELSE IF (ELRT(MAXNINT*13+L-L1+1,IJK,IEG)
     *                  .EQ. -10.Q0
     *             .OR. ELRTPT((L-L1+1)*6-3,IJK,IEG) .GT. 0.Q0) THEN
                        ELRTPT((L-L1+1)*6-3,IJK,IEG) = 0.Q0
                        ELRTPT((L-L1+1)*6-1,IJK,IEG) = 0.Q0
                        ELRTPT((L-L1+1)*6,IJK,IEG) = 0.Q0
                        ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = -10.Q0
                        C11 = EE/(1.Q0-PR**2.Q0)
                        C22 = C22*PR
                        C33 = C11*(1.Q0-PR)/2.Q0
                        DT = 0.Q0
                        DT(1,1) = C11
                        DT(1,2) = C22
                        DT(2,1) = C22
                        DT(2,2) = C11
                        DT(4,4) = C33
```

```
                        DSIG = MATMUL(DT,DEPS)
                        ELRTPT((L-L1+1)*6-5,IJK,IEG) =
     *                      ELRT((L-L1+1)*6-5,IJK,IEG)+DSIG(1)
                        ELRTPT((L-L1+1)*6-4,IJK,IEG) =
     *                      ELRT((L-L1+1)*6-4,IJK,IEG)+DSIG(2)
                        ELRTPT((L-L1+1)*6-2,IJK,IEG) =
     *                      ELRT((L-L1+1)*6-2,IJK,IEG)+DSIG(4)
                        GO TO 1
                    ENDIF
                 ENDIF
                IF (J .EQ. NITMAX .AND. QABS(F) .GT. QABS(TEMPF)) THEN
                    DO 310 K = 1,6
                        DSIG(K) = TEMP(K,1)
  310               CONTINUE
                ENDIF
                IF (QABS(F) .LT. TOLLS) GO TO 2
                IF (J .EQ. NITMAX) GO TO 2
                IF (J .EQ. 1) TEMPF = F
                CALL DFCALC(DF,DQ,X,R,THETA,PSIMAX,ALPHA,GAMMA,ETA,PHI,
     *              OMEGA,TEMPPSI,BETA,DBETADPSI,YST,DFDEP,SXX,SYY,SZZ,
     *              SXY,SYZ,SXZ)
c                CALL DFCALC2(DF,ELRTPT((L-L1+1)*6-5,IJK,IEG),
c     *              TEMPPSI,PSIMAX,BETA,YST)
                DEN = 0.Q0
                DO 350 K = 1,6
                    DEN = DEN + DF(K,1)*DF(K,1)
  350           CONTINUE
                IF (DEN .EQ. 0.) GO TO 2
                DO 360 K = 1,6
                    DSIG(K) = DSIG(K) - DF(K,1)*F/DEN
  360           CONTINUE
  300       CONTINUE
    2       PSI = TEMPPSI
            DO 390 J=1,6
                NSIG = (L-L1+1)*6-6+J
                ELRTPT(NSIG,IJK,IEG) = ELRTPT(NSIG,IJK,IEG)+DSIG(J)
  390       CONTINUE
  220 CONTINUE
      ELRTPT(MAXNINT*12+L-L1+1,IJK,IEG) = PSI
      ELRTPT(MAXNINT*13+L-L1+1,IJK,IEG) = 10.Q0
C
C CE 108 pg. 9-19 step h. Add contribution to force vector
    1 DO 400 I=1,6
         DO 410 J=1,NEN*3
            NSIG=(L-L1+1)*6-6+I
            REP(J) = REP(J)+D1*BT(J,I)*ELRTPT(NSIG,IJK,IEG)
  410    CONTINUE
  400 CONTINUE
  100 CONTINUE
      IF (IFLAG.EQ.0) ELRTPT(MAXNINT*(6*2+3)+1,IJK,IEG)=VOL
      RETURN
      END


      SUBROUTINE STANAL(COOR,ID,R,B,H,F,RTPT,DX,XT,XTPT,ELRT,ELRTPT,
     *   AXLD,NTYPE,NSSTEPS,NHSTEPS,ZCOR,NITER)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NPAR(8)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      COMMON /TOL/ TOLF,TOLM,NITMAX
      COMMON /DEBUG/ IDEBUG
      COMMON /MISC/ PSIMAX,KAPPA
      DIMENSION COOR(NNP,3),ID(NNP,3),R(NNP,3),B(N5-1),H(NEQ,MBAND),
     *   F(NEQ),RTPT(NEQ),DX(NEQ),XT(NEQ),XTPT(NEQ),
     *   ELRT(N12-N11),ELRTPT(N13-N12),NTYPE(NNP)
      REAL(KIND=16) KAPPA
      TIME=0.Q0
      DT=QABS(AXLD/NSSTEPS)
C Calculate elastic stiffness matrix and factor
C Remove below code to iterate with tangent stiffness
C
```

```
      IGO=3
      H=0.Q0
      REWIND 7
      DO 10 IEG=1,NEG1
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSA(COOR,B,H)
   10 CONTINUE
      DO 20 IEG=1,NEG10
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSB(COOR,B,H)
   20 CONTINUE
      DO 30 IEG=1,NEG11
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSC(COOR,B,H)
   30 CONTINUE
      DO 40 IEG=1,NEG100
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDA(COOR,B,H)
   40 CONTINUE
      DO 50 IEG=1,NEG110
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDB(COOR,B,H)
   50 CONTINUE
c add large stiffness in two rotational DOFs
      H(NEQ-1,1)=H(NEQ-1,1)+1.0Q17
      H(NEQ,1)=H(NEQ,1)+1.0Q17
c
      CALL BSOLVE(H,DX,NEQ,MBAND,1,0)
      DO 100 N=1,NSSTEPS
      TIME=TIME+DT
      FRAC=QABS(TIME/AXLD)
      IF (N .LT. NHSTEPS) THEN
        FRAC2=(N*1.Q0)/(NHSTEPS*1.Q0)
      ELSE
        FRAC2 = 1.Q0
      ENDIF
      IF(AXLD.EQ.0.) FRAC=1.Q0
C  Set previous time step displacement and element responses equal
C     to current time step values
C
      XT = XTPT
      ELRT = ELRTPT
      NITER=0
      DO 200 K=1,NITMAX
      DO 205 I=1,NEQ-3
         DX(I)=F(I)*FRAC2-RTPT(I)
  205 CONTINUE
c removed P-delta moment (should be to NEQ, not to NEQ-2)
      DO 210 I=NEQ-2,NEQ-1
         DX(I)=F(I)*FRAC-RTPT(I)
  210 CONTINUE
        DO 220 I=1,NNP
         DO 221 J=1,3
            IDI=ID(I,J)
            IF(IDI.EQ.0) GO TO 221
            IF(QABS(DX(IDI)).GT.TOLF) GO TO 1
  221    CONTINUE
  220 CONTINUE
      IF(QABS(DX(NEQ-2)).GT.TOLF) GO TO 1
      IF(QABS(DX(NEQ-1)).GT.TOLM) GO TO 1
      IF(QABS(DX(NEQ)).GT.TOLM) GO TO 1
      GO TO 2
    1 CALL BSOLVE(H,DX,NEQ,MBAND,0,1)
      DO 240 I=1,NNP
         DO 241 J=1,3
            IDI=ID(I,J)
            IF (IDI.EQ.0) GO TO 241
            COOR(I,J)=COOR(I,J) + DX(IDI)
  241    CONTINUE
  240 CONTINUE
      DO 250 I=1,NEQ
         XTPT(I) = XTPT(I) + DX(I)
         DX(I) = XTPT(I)- XT(I)
  250 CONTINUE
      DO 260 I=1,NNP
         IF (NTYPE(I).EQ.1) COOR(I,3)=ZCOR+XTPT(NEQ-2)
```

```
     *      +COOR(I,2)*XTPT(NEQ-1)-COOR(I,1)*XTPT(NEQ)
  260 CONTINUE
      RTPT=0.Q0
      IGO=4
      REWIND 7
      DO 270 IEG=1,NEG1
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSA(COOR,B,H)
  270 CONTINUE
      DO 271 IEG=1,NEG10
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSB(COOR,B,H)
  271 CONTINUE
      DO 272 IEG=1,NEG11
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL TRUSSC(COOR,B,H)
  272 CONTINUE
      DO 273 IEG=1,NEG100
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDA(COOR,B,H)
  273 CONTINUE
      DO 274 IEG=1,NEG110
      READ(7) NEND,NPAR,(B(I),I=1,NEND)
      CALL SOLIDB(COOR,B,H)
  274 CONTINUE
c add large FORCE in two rotational DOFs
      RTPT(NEQ-1)=RTPT(NEQ-1)+1.0Q17*DX(NEQ-1)
      RTPT(NEQ)=RTPT(NEQ)+1.0Q17*DX(NEQ)
      NITER=K
  200 CONTINUE
    2 WRITE(6,2000) TIME,NITER
      WRITE(*,2000) TIME,NITER
 2000 FORMAT(1X,'LOAD =',E12.5,5X,'NITER =',I5)
      IF (NITER .EQ. NITMAX) WRITE(6,2001) F(NEQ-2)*FRAC-RTPT(NEQ-2)
 2001 FORMAT(6X, 'AXIAL LOAD IMBALANCE = ',E12.5)
  100 CONTINUE
      WRITE(6,2003)
 2003 FORMAT(2(/),25X,'POST AXIAL LOAD ELEMENT RESPONSES',/)
      CALL PPICT(COOR,ID,B,H,XTPT,2,NTYPE,TIME)
      RETURN
      END

      SUBROUTINE TOP(NTYPE,LM,NEN,NEL,NNP)
C Create array NTYPE which denotes top/bottom plane for each node
C  **Only call from element subroutines which have nodes in the top plane**
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION NTYPE(NNP),LM(NEN,NEL)
      DO 10 I=1,NEL
         DO 20 J=1,NEN/2
            NODE=LM(J,I)
            NTYPE(NODE)=1
   20    CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE TRI (P,CX,CY,T)
C TRI COMPUTES SHAPE FUNCTION VALUES AND DERIVATIVES FOR A 6 NODE ELEMENT
C AT THE LOCAL COORDINATE LOCATION CX,CY,T.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /DEBUG/ IDEBUG
      DIMENSION P(4,8)
C
      DO 10 I=1,4
C I=1 FOR SHAPE FUNCTION VALUES; =2 FOR CX DERIVATIVES; =3 FOR CY DERIVATIVES;
C =4 FOR T DERIVATIVES.
      IF (I.EQ.1) GO TO 4
      IF (I.EQ.2) GO TO 5
      IF (I.EQ.3) GO TO 6
      IF (I.EQ.4) GO TO 7
    4 R = CX
      S = CY
      V = 1.Q0 - R - S
```

```
      AT = 1.Q0 + T
      ST = 1.Q0 - T
      GO TO 8
    5 R = 1.Q0
      S = 0.Q0
      V = -1.Q0
      GO TO 8
    6 R = 0.Q0
      S = 1.Q0
      GO TO 8
    7 R = CX
      S = CY
      V = 1.Q0 - R - S
      AT = 1.Q0
      ST = -1.Q0
    8 P(I,1) = 0.5Q0*R*AT
      P(I,2) = 0.5Q0*S*AT
      P(I,3) = 0.5Q0*V*AT
      P(I,4) = 0.5Q0*R*ST
      P(I,5) = 0.5Q0*S*ST
      P(I,6) = 0.5Q0*V*ST
   10 CONTINUE
      RETURN
      END


      SUBROUTINE TRUSSA(A,B,C)
C Sets up storage for subroutine TRUSS1.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      REAL(KIND=16), ALLOCATABLE, DIMENSION(:) :: BSUB
      COMMON /INFO/ N5TEMP
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NDUM(5)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      COMMON /DEBUG/ IDEBUG
      DIMENSION A(N4-1),B(N5-1),C(N13-1)
C
      IF (MAXNEL1.LT.NEL) MAXNEL1=NEL
      N401=1   +NUMAT    ! E
      N402=N401+NUMAT    ! AREA
      N403=N402+NUMAT    ! YST
      N404=N403+NUMAT    ! YRT
      N405=N404+NEL      ! MAT
      N406=N405+2*NEL    ! LM
      N407=N406+8*NEL    ! IDL
      N5TEMP=N407+NEL    ! ALEN
      NEND=N5TEMP-1
      IF (IGO.EQ.1) THEN
          ALLOCATE(BSUB(NEND))
          BSUB=0.Q0
          CALL TRUSS1(A(N1),A(N2),A(N3),BSUB(1),BSUB(N401),BSUB(N402),
     *       BSUB(N403),BSUB(N404),BSUB(N405),BSUB(N406),BSUB(N407),
     *       C(1),C(N6),C(N7),C(N8),C(N9),C(N10),C(N11),C(N12))
          WRITE(7) NEND,NELTYP,NEL,NUMAT,(NDUM(I),I=1,5),
     *       (BSUB(I),I=1,NEND)
          DEALLOCATE(BSUB)
      ELSE
          CALL TRUSS1(A(N1),A(N2),A(N3),B(1),B(N401),B(N402),B(N403),
     *      B(N404),B(N405),B(N406),B(N407),C(1),C(N6),C(N7),
     *      C(N8),C(N9),C(N10),C(N11),C(N12))
      ENDIF
      RETURN
      END

      SUBROUTINE TRUSS1(COOR,ID,R,E,AREA,YST,YRT,MAT,LM,IDL,ALEN,
     *     H,F,RTPT,DX,XT,XTPT,ELRT,ELRTPT)
C Performs computations for longitudinal truss elements.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
```

```
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NDUM(5)
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),ID(NNP,3),R(NNP,3),E(NUMAT),
     * AREA(NUMAT),YST(NUMAT),YRT(NUMAT),MAT(NEL),LM(2,NEL),
     * IDL(8,NEL),ALEN(NEL),ELRT(4,MAXNEL1,NEG1),ELRTPT(4,MAXNEL1,NEG1),
     * H(NEQ,MBAND),F(NEQ),RTPT(NEQ),DX(NEQ),XT(NEQ),XTPT(NEQ)
      DIMENSION HEP(6,6),REP(6),COORL(2,3),XP(6),T(6,8),TT(8,6),
     * TEMP(6,8),HE(8,8),RE(8)
C
      NEN=2             ! Number of nodes per element
      NEDN=3            ! Number of degrees of freedom per node
      NLED=6            ! Number of local degrees of freedom per element
      NGED=NLED-NEN/2+3 !Number of global degrees of freedom per element
C
C NGED reflects removing the vertical degrees of freedom from the nodes
C   in the upper plane then adding back in the 3 top plane DOFs
C
      IF (IGO.EQ.1) GOTO 1 ! Input element data
      IF (IGO.EQ.2) GOTO 2 ! Transfer initial lengths to ELRT
      IF (IGO.EQ.3) GOTO 3 ! calculate the tangent stiffness, H
      IF (IGO.EQ.4) GOTO 4 ! calculate RTPT and ELRTPT
      IF (IGO.EQ.5) GOTO 5 ! Print the system picture
      WRITE(6,1000)
 1000 FORMAT('Fatal error in subroutine TRUSS1. Invalid value for IGO.')
      STOP
C
C Read and print element data.  Calculate assembly arrays and half-
C bandwidth.
    1 WRITE(6,1001) IEG
 1001 FORMAT(//,2X,'LONGITUDINAL REBAR ELEMENT GROUP NUMBER',I4,10X)
      WRITE(6,1002) NEL,NUMAT
 1002 FORMAT(/,4X,'NEL =',I4,4X,'NUMAT =',I3)
      WRITE(6,1003)
 1003 FORMAT(/,4X,'MATERIAL/GEOMETRIC SETS',/,9X,'SET',14X,'E',
     * 9X,'AREA',10X,'YST',10X,'YRT')
      DO 100 I=1,NUMAT
      READ(5,*) E(I),AREA(I),YST(I),YRT(I)
      WRITE(6,1004) I,E(I),AREA(I),YST(I),YRT(I)
 1004 FORMAT(8X,I4,2X,4(1X,E12.5))
  100 CONTINUE
C
C Generates vector MAT and matrix LM
      CALL LMGEN(LM,MAT,NEN,NEL)
      WRITE(6,1005)
 1005 FORMAT(/,4X,'MATERIAL SET NUMBERS AND CONNECTIVITY VECTORS',/,
     * 9X,'ELE',8X,'LENGTH',3X,'SET',5X,'NODE NUMBERS')
      DO 120 N=1,NEL
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      ALEN(N)=0.Q0
      DO 110 I=1,3
      XP(I)=COORL(1,I)-COORL(2,I)
      ALEN(N)=ALEN(N)+XP(I)*XP(I)
  110 CONTINUE
      ALEN(N)=QSQRT(ALEN(N))
      WRITE(6,1006) N,ALEN(N),MAT(N),(LM(I,N),I=1,NEN)
 1006 FORMAT(8X,I4,2X,E12.5,2X,I4,5X,2(2X,I4))
  120 CONTINUE
C
C Transfers equation numbers for all elements from ID to IDL.
C   For elements having half their nodes in the top plane
      CALL LCLID(ID,LM,IDL,NEDN,NGED,NEN,NEL,NEQ,NNP)
      CALL BAND(IDL,NGED,NEL,MBAND)
      RETURN
C
C Transfer initial element lengths to ELRTPT
    2 DO 200 N=1,NEL
        ELRTPT(4,N,IEG)=ALEN(N)
  200 CONTINUE
      RETURN
C
C Calculate and assemble element matrices and vectors
    3 DO 300 N=1,NEL
      M=MAT(N)
```

```
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      CLEN=0.Q0
      DO 310 I=1,3
      XP(I)=COORL(1,I)-COORL(2,I)
      CLEN=CLEN+XP(I)*XP(I)
  310 CONTINUE
      CLEN=QSQRT(CLEN)
      DO 320 I=1,3
      XP(I)=XP(I)/CLEN
      XP(I+3)=-XP(I)
  320 CONTINUE
C Define transformation matrix from local to global DOFs
C
      T=0.Q0
      T(1,1)=1.Q0
      T(2,2)=1.Q0
      T(4,3)=1.Q0
      T(5,4)=1.Q0
      T(6,5)=1.Q0
      T(3,6)=1.Q0
      T(3,7)=COORL(1,2)
      T(3,8)=-COORL(1,1)
      DO 330 I=1,8
         DO 331 J=1,6
            TT(I,J)=T(J,I)
  331 CONTINUE
  330 CONTINUE
      EAA=E(M)*AREA(M)/ALEN(N)
      ELS=QABS(E(M)*ELRTPT(2,N,IEG)) ! ELRTPT(2)=elastic strain
      IF(ELS.GE.YST(M)) EAA=EAA*YRT(M)
      DO 340 I=1,6
      DO 341 J=1,6
      HEP(I,J)=XP(I)*XP(J)*EAA
  341 CONTINUE
  340 CONTINUE
C Transform to global DOFs
C
      TEMP=MATMUL(HEP,T)
      HE=MATMUL(TT,TEMP)
      CALL ASSMBL(H,F,HE,RE,IDL(1,N),NEQ,MBAND,NGED,1,0)
  300 CONTINUE
      RETURN
C
C Compute and print stresses, strains, forces and displacements.
    4 DO 400 N=1,NEL
      M=MAT(N)
      PLEN=ELRT(4,N,IEG)        ! Length at previous time step
      EPSET=ELRT(2,N,IEG)       ! Elastic strain at previous time step
      EPSETPT=ELRTPT(2,N,IEG)   ! Elastic strain at current time step
      YEPS=YST(M)/E(M)          ! Yield strain
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      CLEN=0.Q0
      DO 410 I=1,3
         XP(I)=COORL(1,I)-COORL(2,I)
         CLEN=CLEN+XP(I)*XP(I)
  410 CONTINUE
      CLEN=QSQRT(CLEN)
      DO 420 I=1,3
         XP(I)=XP(I)/CLEN
         XP(I+3)=-XP(I)
  420 CONTINUE
C Determine if loading plastically (1), loading elastically (2),
C    loading transition (3), or unloading elastically (4)
      EPST=(PLEN-ALEN(N))/ALEN(N)   ! Previous total strain
      EPSTPT=(CLEN-ALEN(N))/ALEN(N) ! Current total strain
      DEPS=EPSTPT-EPST              ! Change in total strain
C Assume elastic then check assumption
      EPSETPT=EPSET+DEPS
      IF (QABS(EPSTPT).LT.QABS(EPST) .AND. QABS(EPSETPT).LE.YEPS) THEN
         IFLAG=4        ! Elastic unloading
      ELSEIF (QABS(EPSETPT).LE.YEPS) THEN
```

```
         IFLAG=2        ! Elastic loading
       ELSEIF (QABS(EPSET).LT.YEPS) THEN
         IFLAG=3        ! Transition loading
       ELSEIF (QABS(EPSET).GE.YEPS) THEN
         IFLAG=1        ! Plastic loading
       ELSE
         WRITE(6,4000) N,IEG
 4000    FORMAT(2(/),5X,'NO CASE FOUND FOR ELEMENT ',I4,
     *      ' OF LONGITUDINAL REBAR GROUP ',I4)
         STOP
       ENDIF
       IF (IFLAG.EQ.2 .OR. IFLAG.EQ.4) THEN
         ELRTPT(2,N,IEG)=ELRT(2,N,IEG)+DEPS
         ELRTPT(3,N,IEG)=ELRT(3,N,IEG)
       ENDIF
       IF (IFLAG.EQ.3 .OR. IFLAG.EQ.1) THEN
         IF (DEPS.GT.0) ELRTPT(2,N,IEG)=YEPS
         IF (DEPS.LT.0) ELRTPT(2,N,IEG)=-YEPS
         IF (DEPS.EQ.0) ELRTPT(2,N,IEG)=ELRT(2,N,IEG)
         ELRTPT(3,N,IEG)=ELRT(3,N,IEG)+DEPS
     *      -(ELRTPT(2,N,IEG)-ELRT(2,N,IEG))
       ENDIF
       ELRTPT(1,N,IEG)=ELRTPT(2,N,IEG)*E(M)+ELRTPT(3,N,IEG)*YRT(M)*E(M)
       ELRTPT(4,N,IEG)=CLEN
C Define transformation matrix from local to global DOFs
C
       TT=0.Q0
       TT(1,1)=1.Q0
       TT(2,2)=1.Q0
       TT(3,4)=1.Q0
       TT(4,5)=1.Q0
       TT(5,6)=1.Q0
       TT(6,3)=1.Q0
       TT(7,3)=COORL(1,2)
       TT(8,3)=-COORL(1,1)
       REP=0.Q0
       RE=0.Q0
       DO 430 I=1,6
         REP(I)=ELRTPT(1,N,IEG)*XP(I)*AREA(M)
  430  CONTINUE
       DO 440 I=1,8
         DO 450 J=1,6
           RE(I)=RE(I)+TT(I,J)*REP(J)
  450    CONTINUE
  440  CONTINUE
       CALL ASSMBL(H,RTPT,HE,RE,IDL(1,N),NEQ,MBAND,NGED,0,1)
  400  CONTINUE
       RETURN

    5  WRITE(6,5000) IEG
 5000  FORMAT(//,2X,'LONGITUDINAL REBAR ELEMENT GROUP NUMBER',I4,10X)
       DO 500 N=1,NEL
       M=MAT(N)
       WRITE(6,5001) N
 5001  FORMAT(/,4X,'ELEMENT NUMBER',I4,/,7X,'YIELD STRESS',10X,'STRESS',
     *    11X,'FORCE',6X,'YIELD STRAIN',6X,'ELASTIC STRAIN',6X,
     *    'PLASTIC STRAIN')
       FORCE=ELRTPT(1,N,IEG)*AREA(M)
       YEPS=YST(M)/E(M)
       WRITE(6,5002) YST(M),ELRTPT(1,N,IEG),FORCE,
     *    YEPS,ELRTPT(2,N,IEG),ELRTPT(3,N,IEG)
 5002  FORMAT(3X,3(4X,E12.5),6X,E12.5,2(8X,E12.5))
  500  CONTINUE
       RETURN
       END

       SUBROUTINE TRUSSB(A,B,C)
C Sets up storage for subroutine TRUSS10.
       IMPLICIT REAL(KIND=16) (A-H,O-Z)
       IMPLICIT INTEGER(KIND=8) (I-N)
       REAL(KIND=16), ALLOCATABLE, DIMENSION(:) :: BSUB
       COMMON /INFO/ N5TEMP
       COMMON /INFO1/ NNP,NEQ,MBAND,IGO
       COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
```

```
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NDUM(5)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      COMMON /DEBUG/ IDEBUG
      DIMENSION A(N4-1),B(N5-1),C(N13-1)
C
      IF (MAXNEL10.LT.NEL) MAXNEL10=NEL
      N401=1   +NUMAT    ! E
      N402=N401+NUMAT    ! AREA
      N403=N402+NUMAT    ! YST
      N404=N403+NUMAT    ! YRT
      N405=N404+NEL      ! MAT
      N406=N405+2*NEL    ! LM
      N407=N406+6*NEL    ! IDL
      N5TEMP=N407+NEL    ! ALEN
      NEND=N5TEMP-1
      IF (IGO.EQ.1) THEN
          ALLOCATE(BSUB(NEND))
          BSUB=0.Q0
          CALL TRUSS10(A(N1),A(N2),A(N3),BSUB(1),BSUB(N401),BSUB(N402),
     *       BSUB(N403),BSUB(N404),BSUB(N405),BSUB(N406),BSUB(N407),
     *       C(1),C(N6),C(N7),C(N8),C(N9),C(N10),C(N1101),C(N1201))
          WRITE(7) NEND,NELTYP,NEL,NUMAT,(NDUM(I),I=1,5),
     *       (BSUB(I),I=1,NEND)
          DEALLOCATE(BSUB)
      ELSE
          CALL TRUSS10(A(N1),A(N2),A(N3),B(1),B(N401),B(N402),B(N403),
     *      B(N404),B(N405),B(N406),B(N407),C(1),C(N6),C(N7),
     *      C(N8),C(N9),C(N10),C(N1101),C(N1201))
      ENDIF
      RETURN
      END

      SUBROUTINE TRUSS10(COOR,ID,R,E,AREA,YST,YRT,MAT,LM,IDL,ALEN,
     *     H,F,RTPT,DX,XT,XTPT,ELRT,ELRTPT)
C Performs computations for lower transverse truss elements.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NDUM(5)
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),ID(NNP,3),R(NNP,3),E(NUMAT),
     * AREA(NUMAT),YST(NUMAT),YRT(NUMAT),MAT(NEL),LM(2,NEL),
     * IDL(3,2,NEL),ALEN(NEL),ELRT(4,MAXNEL10,NEG10),
     * ELRTPT(4,MAXNEL10,NEG10),H(NEQ,MBAND),F(NEQ),RTPT(NEQ),DX(NEQ),
     * XT(NEQ),XTPT(NEQ)
      DIMENSION HE(6,6),RE(6),COORL(2,3),XP(6)
C
      NEN=2    ! Number of nodes per element
      NEDN=3   ! Number of degrees of freedom per node
      NED=6    ! Number of degrees of freedom per element
C
      IF (IGO.EQ.1) GOTO 1 ! Input element data
      IF (IGO.EQ.2) GOTO 2 ! Transfer initial lengths to ELRT
      IF (IGO.EQ.3) GOTO 3 ! calculate the tangent stiffness, H
      IF (IGO.EQ.4) GOTO 4 ! calculate RTPT and ELRTPT
      IF (IGO.EQ.5) GOTO 5 ! Print the system picture
      WRITE(6,1000)
 1000 FORMAT('Fatal error in subroutine TRUSS10. ',
     *   'Invalid value for IGO.')
      STOP
C
C Read and print element data.  Calculate assembly arrays and half-
C bandwidth.
    1 WRITE(6,1001) IEG
 1001 FORMAT(//,2X,'LOWER TRANSVERSE REBAR ELEMENT GROUP NUMBER',I4,10X)
      WRITE(6,1002) NEL,NUMAT
 1002 FORMAT(/,4X,'NEL =',I4,4X,'NUMAT =',I3)
      WRITE(6,1003)
 1003 FORMAT(/,4X,'MATERIAL/GEOMETRIC SETS',/,9X,'SET',14X,'E',
     * 9X,'AREA',10X,'YST',10X,'YRT')
      DO 100 I=1,NUMAT
      READ(5,*) E(I),AREA(I),YST(I),YRT(I)
```

```
      WRITE(6,1004) I,E(I),AREA(I),YST(I),YRT(I)
 1004 FORMAT(8X,I4,2X,4(1X,E12.5))
  100 CONTINUE
C
C Generates vector MAT and matrix LM
      CALL LMGEN(LM,MAT,NEN,NEL)
      WRITE(6,1005)
 1005 FORMAT(/,4X,'MATERIAL SET NUMBERS AND CONNECTIVITY VECTORS',/,
     * 9X,'ELE',8X,'LENGTH',3X,'SET',5X,'NODE NUMBERS')
      DO 120 N=1,NEL
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      ALEN(N)=0.Q0
      DO 110 I=1,3
      XP(I)=COORL(1,I)-COORL(2,I)
      ALEN(N)=ALEN(N)+XP(I)*XP(I)
  110 CONTINUE
      ALEN(N)=QSQRT(ALEN(N))
      WRITE(6,1006) N,ALEN(N),MAT(N),(LM(I,N),I=1,NEN)
 1006 FORMAT(8X,I4,2X,E12.5,2X,I4,5X,2(2X,I4))
  120 CONTINUE
C
C Transfers equation numbers for all elements from ID to IDL.
C  For elements which do not lie in the top plane.
      CALL LCLID2(ID,LM,IDL,NNP,NEDN,NEN,NEL)
      CALL BAND(IDL,NED,NEL,MBAND)
      RETURN
C
C Transfer initial element lengths to ELRTPT
    2 DO 200 N=1,NEL
         ELRTPT(4,N,IEG)=ALEN(N)
  200 CONTINUE
      RETURN
C
C Calculate and assemble element matrices and vectors
    3 DO 300 N=1,NEL
      M=MAT(N)
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      CLEN=0.Q0
      DO 310 I=1,3
      XP(I)=COORL(1,I)-COORL(2,I)
      CLEN=CLEN+XP(I)*XP(I)
  310 CONTINUE
      CLEN=QSQRT(CLEN)
      DO 320 I=1,3
      XP(I)=XP(I)/CLEN
      XP(I+3)=-XP(I)
  320 CONTINUE
      EAA=E(M)*AREA(M)/ALEN(N)
      ELS=QABS(E(M)*ELRTPT(2,N,IEG)) ! ELRTPT(2)=elastic strain
      IF(ELS.GE.YST(M)) EAA=EAA*YRT(M)
      DO 340 I=1,6
      DO 341 J=1,6
      HE(I,J)=XP(I)*XP(J)*EAA
  341 CONTINUE
  340 CONTINUE
      CALL ASSMBL(H,F,HE,RE,IDL(1,1,N),NEQ,MBAND,NED,1,0)
  300 CONTINUE
      RETURN
C
C Compute and print stresses, strains, forces and displacements.
    4 DO 400 N=1,NEL
      M=MAT(N)
      PLEN=ELRT(4,N,IEG)        ! Length at previous time step
      EPSET=ELRT(2,N,IEG)       ! Elastic strain at previous time step
      EPSETPT=ELRTPT(2,N,IEG)   ! Elastic strain at current time step
      YEPS=YST(M)/E(M)          ! Yield strain
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      CLEN=0.Q0
      DO 410 I=1,3
```

```
         XP(I)=COORL(1,I)-COORL(2,I)
         CLEN=CLEN+XP(I)*XP(I)
  410 CONTINUE
      CLEN=QSQRT(CLEN)
      DO 420 I=1,3
         XP(I)=XP(I)/CLEN
         XP(I+3)=-XP(I)
  420 CONTINUE
C Determine if loading plastically (1), loading elastically (2),
C    loading transition (3), or unloading elastically (4)
      EPST=(PLEN-ALEN(N))/ALEN(N)    ! Previous total strain
      EPSTPT=(CLEN-ALEN(N))/ALEN(N) ! Current total strain
      DEPS=EPSTPT-EPST               ! Change in total strain
C Assume elastic then check assumption
      EPSETPT=EPSET+DEPS
      IF (QABS(EPSTPT).LT.QABS(EPST) .AND. QABS(EPSETPT).LE.YEPS) THEN
         IFLAG=4         ! Elastic unloading
      ELSEIF (QABS(EPSETPT).LE.YEPS) THEN
         IFLAG=2         ! Elastic loading
      ELSEIF (QABS(EPSET).LT.YEPS) THEN
         IFLAG=3         ! Transition loading
      ELSEIF (QABS(EPSET).GE.YEPS) THEN
         IFLAG=1         ! Plastic loading
      ELSE
         WRITE(6,4000) N,IEG
 4000    FORMAT(2(/),5X,'NO CASE FOUND FOR ELEMENT ',I4,
     *      ' OF LOWER TRANSVERSE REBAR GROUP ',I4)
         STOP
      ENDIF
      IF (IFLAG.EQ.2 .OR. IFLAG.EQ.4) THEN
         ELRTPT(2,N,IEG)=ELRT(2,N,IEG)+DEPS
         ELRTPT(3,N,IEG)=ELRT(3,N,IEG)
      ENDIF
      IF (IFLAG.EQ.3 .OR. IFLAG.EQ.1) THEN
         IF (DEPS.GT.0) ELRTPT(2,N,IEG)=YEPS
         IF (DEPS.LT.0) ELRTPT(2,N,IEG)=-YEPS
         IF (DEPS.EQ.0) ELRTPT(2,N,IEG)=ELRT(2,N,IEG)
         ELRTPT(3,N,IEG)=ELRT(3,N,IEG)+DEPS
     *      -(ELRTPT(2,N,IEG)-ELRT(2,N,IEG))
      ENDIF
      ELRTPT(1,N,IEG)=ELRTPT(2,N,IEG)*E(M)+ELRTPT(3,N,IEG)*YRT(M)*E(M)
      ELRTPT(4,N,IEG)=CLEN
      RE=0.QO
      DO 430 I=1,6
         RE(I)=ELRTPT(1,N,IEG)*XP(I)*AREA(M)
  430 CONTINUE
      CALL ASSMBL(H,RTPT,HE,RE,IDL(1,1,N),NEQ,MBAND,NED,0,1)
  400 CONTINUE
      RETURN

    5 WRITE(6,5000) IEG
 5000 FORMAT(//,2X,'LOWER TRANSVERSE REBAR ELEMENT GROUP NUMBER',I4,10X)
      DO 500 N=1,NEL
      M=MAT(N)
      WRITE(6,5001) N
 5001 FORMAT(/,4X,'ELEMENT NUMBER',I4,/,7X,'YIELD STRESS',10X,'STRESS',
     *   11X,'FORCE',6X,'YIELD STRAIN',6X,'ELASTIC STRAIN',6X,
     *   'PLASTIC STRAIN')
      FORCE=ELRTPT(1,N,IEG)*AREA(M)
      YEPS=YST(M)/E(M)
      WRITE(6,5002) YST(M),ELRTPT(1,N,IEG),FORCE,
     *   YEPS,ELRTPT(2,N,IEG),ELRTPT(3,N,IEG)
 5002 FORMAT(3X,3(4X,E12.5),6X,E12.5,2(8X,E12.5))
  500 CONTINUE
      RETURN
      END

      SUBROUTINE TRUSSC(A,B,C)
C Sets up storage for subroutine TRUSS11.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      REAL(KIND=16), ALLOCATABLE, DIMENSION(:) :: BSUB
      COMMON /INFO/ N5TEMP
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
```

```
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NDUM(5)
      COMMON /POINTS/ N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N1101,N1102,
     * N1103,N1104,N12,N1201,N1202,N1203,N1204,N13
      COMMON /DEBUG/ IDEBUG
      DIMENSION A(N4-1),B(N5-1),C(N13-1)
C
      IF (MAXNEL11.LT.NEL) MAXNEL11=NEL
      N401=1   +NUMAT    ! E
      N402=N401+NUMAT    ! AREA
      N403=N402+NUMAT    ! YST
      N404=N403+NUMAT    ! YRT
      N405=N404+NEL      ! MAT
      N406=N405+2*NEL    ! LM
      N407=N406+7*NEL    ! IDL
      N5TEMP=N407+NEL    ! ALEN
      NEND=N5TEMP-1
      IF (IGO.EQ.1) THEN
          ALLOCATE(BSUB(NEND))
          BSUB=0.Q0
          CALL TRUSS11(A(N1),A(N2),A(N3),BSUB(1),BSUB(N401),BSUB(N402),
     *       BSUB(N403),BSUB(N404),BSUB(N405),BSUB(N406),BSUB(N407),
     *       C(1),C(N6),C(N7),C(N8),C(N9),C(N10),C(N1102),C(N1202))
          WRITE(7) NEND,NELTYP,NEL,NUMAT,(NDUM(I),I=1,5),
     *       (BSUB(I),I=1,NEND)
          DEALLOCATE(BSUB)
      ELSE
          CALL TRUSS11(A(N1),A(N2),A(N3),B(1),B(N401),B(N402),B(N403),
     *      B(N404),B(N405),B(N406),B(N407),C(1),C(N6),C(N7),
     *      C(N8),C(N9),C(N10),C(N1102),C(N1202))
      ENDIF
      RETURN
      END

      SUBROUTINE TRUSS11(COOR,ID,R,E,AREA,YST,YRT,MAT,LM,IDL,ALEN,
     *     H,F,RTPT,DX,XT,XTPT,ELRT,ELRTPT)
C Performs computations for upper transverse truss elements.
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      COMMON /INFO1/ NNP,NEQ,MBAND,IGO
      COMMON /INFO2/ IEG,NEG1,NEG10,NEG11,NEG100,NEG110,MAXNEL1,
     * MAXNEL10,MAXNEL11,MAXNEL100,MAXNEL110,MAXNINT
      COMMON /ELPAR/ NELTYP,NEL,NUMAT,NDUM(5)
      COMMON /DEBUG/ IDEBUG
      DIMENSION COOR(NNP,3),ID(NNP,3),R(NNP,3),E(NUMAT),
     * AREA(NUMAT),YST(NUMAT),YRT(NUMAT),MAT(NEL),LM(2,NEL),
     * IDL(7,NEL),ALEN(NEL),ELRT(4,MAXNEL11,NEG11),
     * ELRTPT(4,MAXNEL11,NEG11),H(NEQ,MBAND),F(NEQ),RTPT(NEQ),DX(NEQ),
     * XT(NEQ),XTPT(NEQ)
      DIMENSION HEP(6,6),REP(6),COORL(2,3),XP(6),T(6,7),TT(7,6),
     * TEMP(6,7),HE(7,7),RE(7)
C
      NEN=2    ! Number of nodes per element
      NEDN=3   ! Number of degrees of freedom per node
      NLED=6   ! Number of local degrees of freedom per element
      NGED=NLED-NEN+3  ! Number of global degrees of freedom per element
C
C NGED reflects removing the vertical degrees of freedom from both nodes
C   then adding back in the 3 top plane DOFs in place of each
C
      IF (IGO.EQ.1) GOTO 1 ! Input element data
      IF (IGO.EQ.2) GOTO 2 ! Transfer initial lengths to ELRT
      IF (IGO.EQ.3) GOTO 3 ! calculate the tangent stiffness, H
      IF (IGO.EQ.4) GOTO 4 ! calculate RTPT and ELRTPT
      IF (IGO.EQ.5) GOTO 5 ! Print the system picture
      WRITE(6,1000)
 1000 FORMAT('Fatal error in subroutine TRUSS11. ',
     *   'Invalid value for IGO.')
      STOP
C
C Read and print element data.  Calculate assembly arrays and half-
C bandwidth.
    1 WRITE(6,1001) IEG
 1001 FORMAT(//,2X,'UPPER TRANSVERSE REBAR ELEMENT GROUP NUMBER',I4,10X)
      WRITE(6,1002) NEL,NUMAT
```

```
 1002 FORMAT(/,4X,'NEL =',I4,4X,'NUMAT =',I3)
      WRITE(6,1003)
 1003 FORMAT(/,4X,'MATERIAL/GEOMETRIC SETS',/,9X,'SET',14X,'E',
     * 9X,'AREA',10X,'YST',10X,'YRT')
      DO 100 I=1,NUMAT
      READ(5,*) E(I),AREA(I),YST(I),YRT(I)
      WRITE(6,1004) I,E(I),AREA(I),YST(I),YRT(I)
 1004 FORMAT(8X,I4,2X,4(1X,E12.5))
  100 CONTINUE
C
C Generates vector MAT and matrix LM
      CALL LMGEN(LM,MAT,NEN,NEL)
      WRITE(6,1005)
 1005 FORMAT(/,4X,'MATERIAL SET NUMBERS AND CONNECTIVITY VECTORS',/,
     * 9X,'ELE',8X,'LENGTH',3X,'SET',5X,'NODE NUMBERS')
      DO 120 N=1,NEL
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      ALEN(N)=0.Q0
      DO 110 I=1,3
      XP(I)=COORL(1,I)-COORL(2,I)
      ALEN(N)=ALEN(N)+XP(I)*XP(I)
  110 CONTINUE
      ALEN(N)=QSQRT(ALEN(N))
      WRITE(6,1006) N,ALEN(N),MAT(N),(LM(I,N),I=1,NEN)
 1006 FORMAT(8X,I4,2X,E12.5,2X,I4,5X,2(2X,I4))
  120 CONTINUE
C
C Transfers equation numbers for all elements from ID to IDL.
C  For elements having all their nodes in the top plane.
      CALL LCLID3(ID,LM,IDL,NEDN,NGED,NEN,NEL,NEQ,NNP)
      CALL BAND(IDL,NGED,NEL,MBAND)
      RETURN
C
C Transfer initial element lengths to ELRTPT
    2 DO 200 N=1,NEL
         ELRTPT(4,N,IEG)=ALEN(N)
  200 CONTINUE
      RETURN
C
C Calculate and assemble element matrices and vectors
    3 DO 300 N=1,NEL
      M=MAT(N)
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      CLEN=0.Q0
      DO 310 I=1,3
      XP(I)=COORL(1,I)-COORL(2,I)
      CLEN=CLEN+XP(I)*XP(I)
  310 CONTINUE
      CLEN=QSQRT(CLEN)
      DO 320 I=1,3
      XP(I)=XP(I)/CLEN
      XP(I+3)=-XP(I)
  320 CONTINUE
C Define transformation matrix from local to global DOFs
C
      T=0.Q0
      T(1,1)=1.Q0
      T(2,2)=1.Q0
      T(4,3)=1.Q0
      T(5,4)=1.Q0
      T(3,5)=1.Q0
      T(3,6)=COORL(1,2)
      T(3,7)=-COORL(1,1)
      T(6,5)=1.Q0
      T(6,6)=COORL(2,2)
      T(6,7)=-COORL(2,1)
      DO 330 I=1,7
         DO 331 J=1,6
            TT(I,J)=T(J,I)
  331 CONTINUE
  330 CONTINUE
      EAA=E(M)*AREA(M)/ALEN(N)
```

```
      ELS=QABS(E(M)*ELRTPT(2,N,IEG)) ! ELRTPT(2)=elastic strain
      IF(ELS.GE.YST(M)) EAA=EAA*YRT(M)
      DO 340 I=1,6
      DO 341 J=1,6
      HEP(I,J)=XP(I)*XP(J)*EAA
  341 CONTINUE
  340 CONTINUE
C Transform to global DOFs
C
      TEMP=MATMUL(HEP,T)
      HE=MATMUL(TT,TEMP)
      CALL ASSMBL(H,F,HE,RE,IDL(1,N),NEQ,MBAND,NGED,1,0)
  300 CONTINUE
      RETURN
C
C Compute and print stresses, strains, forces and displacements.
    4 DO 400 N=1,NEL
      M=MAT(N)
      PLEN=ELRT(4,N,IEG)        ! Length at previous time step
      EPSET=ELRT(2,N,IEG)       ! Elastic strain at previous time step
      EPSETPT=ELRTPT(2,N,IEG)   ! Elastic strain at current time step
      YEPS=YST(M)/E(M)          ! Yield strain
C
C Transfers nodal coordinates for an element from COOR to COORL.
      CALL LCLCOR(COOR,LM(1,N),COORL,NNP,NEN)
      CLEN=0.Q0
      DO 410 I=1,3
         XP(I)=COORL(1,I)-COORL(2,I)
         CLEN=CLEN+XP(I)*XP(I)
  410 CONTINUE
      CLEN=QSQRT(CLEN)
      DO 420 I=1,3
         XP(I)=XP(I)/CLEN
         XP(I+3)=-XP(I)
  420 CONTINUE
C Determine if loading plastically (1), loading elastically (2),
C    loading transition (3), or unloading elastically (4)
      EPST=(PLEN-ALEN(N))/ALEN(N)   ! Previous total strain
      EPSTPT=(CLEN-ALEN(N))/ALEN(N) ! Current total strain
      DEPS=EPSTPT-EPST              ! Change in total strain
C Assume elastic then check assumption
      EPSETPT=EPSET+DEPS
      IF (QABS(EPSTPT).LT.QABS(EPST) .AND. QABS(EPSETPT).LE.YEPS) THEN
         IFLAG=4        ! Elastic unloading
      ELSEIF (QABS(EPSETPT).LE.YEPS) THEN
         IFLAG=2        ! Elastic loading
      ELSEIF (QABS(EPSET).LT.YEPS) THEN
         IFLAG=3        ! Transition loading
      ELSEIF (QABS(EPSET).GE.YEPS) THEN
         IFLAG=1        ! Plastic loading
      ELSE
         WRITE(6,4000) N,IEG
 4000    FORMAT(2(/),5X,'NO CASE FOUND FOR ELEMENT ',I4,
     *      ' OF UPPER TRANSVERSE REBAR GROUP ',I4)
         STOP
      ENDIF
      IF (IFLAG.EQ.2 .OR. IFLAG.EQ.4) THEN
         ELRTPT(2,N,IEG)=ELRT(2,N,IEG)+DEPS
         ELRTPT(3,N,IEG)=ELRT(3,N,IEG)
      ENDIF
      IF (IFLAG.EQ.3 .OR. IFLAG.EQ.1) THEN
         IF (DEPS.GT.0) ELRTPT(2,N,IEG)=YEPS
         IF (DEPS.LT.0) ELRTPT(2,N,IEG)=-YEPS
         IF (DEPS.EQ.0) ELRTPT(2,N,IEG)=ELRT(2,N,IEG)
         ELRTPT(3,N,IEG)=ELRT(3,N,IEG)+DEPS
     *      -(ELRTPT(2,N,IEG)-ELRT(2,N,IEG))
      ENDIF
      ELRTPT(1,N,IEG)=ELRTPT(2,N,IEG)*E(M)+ELRTPT(3,N,IEG)*YRT(M)*E(M)
      ELRTPT(4,N,IEG)=CLEN
C Define transformation matrix from local to global DOFs
C
      T=0.Q0
      TT(1,1)=1.Q0
      TT(2,2)=1.Q0
```

```
      TT(3,4)=1.Q0
      TT(4,5)=1.Q0
      TT(5,3)=1.Q0
      TT(6,3)=COORL(1,2)
      TT(7,3)=-COORL(1,1)
      TT(5,6)=1.Q0
      TT(6,6)=COORL(2,2)
      TT(7,6)=-COORL(2,1)
      REP=0.Q0
      RE=0.Q0
      DO 430 I=1,6
         REP(I)=ELRTPT(1,N,IEG)*XP(I)*AREA(M)
  430 CONTINUE
      DO 440 I=1,7
         DO 450 J=1,6
            RE(I)=RE(I)+TT(I,J)*REP(J)
  450    CONTINUE
  440 CONTINUE
      CALL ASSMBL(H,RTPT,HE,RE,IDL(1,N),NEQ,MBAND,NGED,0,1)
  400 CONTINUE
      RETURN

    5 WRITE(6,5000) IEG
 5000 FORMAT(//,2X,'UPPER TRANSVERSE REBAR ELEMENT GROUP NUMBER',I4,10X)
      DO 500 N=1,NEL
      M=MAT(N)
      WRITE(6,5001) N
 5001 FORMAT(/,4X,'ELEMENT NUMBER',I4,/,7X,'YIELD STRESS',10X,'STRESS',
     *   11X,'FORCE',6X,'YIELD STRAIN',6X,'ELASTIC STRAIN',6X,
     *   'PLASTIC STRAIN')
      FORCE=ELRTPT(1,N,IEG)*AREA(M)
      YEPS=YST(M)/E(M)
      WRITE(6,5002) YST(M),ELRTPT(1,N,IEG),FORCE,
     *   YEPS,ELRTPT(2,N,IEG),ELRTPT(3,N,IEG)
 5002 FORMAT(3X,3(4X,E12.5),6X,E12.5,2(8X,E12.5))
  500 CONTINUE
      RETURN
      END

      SUBROUTINE zbrent(x1,x2,ELAS,tol,SIG0,DSIG,X,R,THETA,PSI,
     *  PSIMAX,BETA,YST)
      IMPLICIT REAL(KIND=16) (A-H,O-Z)
      IMPLICIT INTEGER(KIND=8) (I-N)
      INTEGER(KIND=8) ITMAX
      REAL(KIND=16) tol,x1,x2,EPS
      PARAMETER (ITMAX=1000,EPS=1.q-16)
C  Using Brents method, find the root of a function func known to lie between x1 and x2.
C  The root, returned as zbrent, will be refined until its accuracy is tol.
C  Parameters: Maximum allowed number of iterations, and machine floating-point precision.
      INTEGER(KIND=8) iter
      REAL(KIND=16) a,b,c,d,e,fa,fb,fc,p,q,r,s,tol1,xm
      a=x1
      b=x2
      CALL FCALC(fa,a,SIG0,DSIG,X,R,THETA,PSI,PSIMAX,BETA,YST)
      CALL FCALC(fb,b,SIG0,DSIG,X,R,THETA,PSI,PSIMAX,BETA,YST)
      if((fa.gt.0..and.fb.gt.0.).or.(fa.lt.0..and.fb.lt.0.)) then
            write(6,*) 'root must be bracketed for zbrent'
            write(6,*) 'fcalc(a) = ',fa
            write(6,*) 'fcalc(b) = ',fb
            write(6,*) 'psi = ', psi
            write(6,*) 'program terminated by subroutine zbrent'
            if (fa .gt. 0.q0) elas = 0.q0
            if (fb .lt. 0.q0) elas = 1.q0
            stop
      endif
      c=b
      fc=fb
      do 11 iter=1,ITMAX
        if((fb.gt.0..and.fc.gt.0.).or.(fb.lt.0..and.fc.lt.0.))then
            c=a     !Rename a, b, c and adjust bounding interval d.
            fc=fa
            d=b-a
            e=d
        endif
        if(abs(fc).lt.abs(fb)) then
```

```
          a=b
          b=c
          c=a
          fa=fb
          fb=fc
          fc=fa
        endif
        tol1=2.Q0*EPS*abs(b)+0.5Q0*tol   !Convergence check.
        xm=.5Q0*(c-b)
        if(abs(xm).le.tol1 .or. fb.eq.0.)then
          ELAS=b
          return
        endif
        if(abs(e).ge.tol1 .and. abs(fa).gt.abs(fb)) then
          s=fb/fa   !Attempt inverse quadratic interpolation.
          if(a.eq.c) then
            p=2.Q0*xm*s
            q=1.Q0-s
          else
            q=fa/fc
            r=fb/fc
            p=s*(2.Q0*xm*q*(q-r)-(b-a)*(r-1.Q0))
            q=(q-1.Q0)*(r-1.Q0)*(s-1.Q0)
          endif
          if(p.gt.0.) q=-q   !Check whether in bounds.
          p=abs(p)
          if(2.Q0*p .lt. min(3.Q0*xm*q-abs(tol1*q),abs(e*q))) then
            e=d   !Accept interpolation.
            d=p/q
          else
            d=xm    !Interpolation failed, use bisection.
            e=d
          endif
        else    !Bounds decreasing too slowly, use bisection.
          d=xm
          e=d
        endif
        a=b    !Move last best guess to a.
        fa=fb
        if(abs(d) .gt. tol1) then    !Evaluate new trial root.
          b=b+d
        else
          b=b+sign(tol1,xm)
        endif
        CALL FCALC(fb,b,SIG0,DSIG,X,R,THETA,PSI,PSIMAX,BETA,YST)
     11 enddo
        pause 'zbrent exceeding maximum iterations'
        ELAS=b
        return
        END
```