```fortran
      SUBROUTINE UEL(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
     1 PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,
     2 DTIME,KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,
     3 PREDEF,NPREDF,LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,
     4 JPROPS,NJPROP,PERIOD)
C
      INCLUDE 'ABA_PARAM.INC'
      PARAMETER ( ZERO=0.D0,HALF=0.5D0,ONE=1.D0,TWO=2.D0,THREE=3.D0,
     *              FOUR=4.D0, FIVE=5.D0, SIX=6.D0, PI=3.14159D0)
C
      DIMENSION RHS(MLVARX,*),AMATRX(NDOFEL,NDOFEL),
     1 SVARS(NSVARS),ENERGY(8),PROPS(*),COORDS(MCRD,NNODE),
     2 U(NDOFEL),DU(MLVARX,*),V(NDOFEL),A(NDOFEL),TIME(2),
     3 PARAMS(3),JDLTYP(MDLOAD,*),ADLMAG(MDLOAD,*),
     4 DDLMAG(MDLOAD,*),PREDEF(2,NPREDF,NNODE),LFLAGS(*),
     5 JPROPS(*)
      DIMENSION SRESID(12)
      REAL Kg(12,12), Fg(12)
      REAL Tlb(6,12),Tgl(12,12),TlbT(12,6),TglT(12,12)
      REAL AM, r, Tr, h, hL, L, Ar, AL, I, S, rg, Ecp, E, Kv0,
     *      Fc, uc, Er, As, Is, Pe, Fcr, ucr, ke, k0, Kr, Kt
      REAL Fcrn, Fmax, umax, fcn, ucn
      INTEGER n
C
C     UEL SUBROUTINE FOR A ELASTOMERIC BEARING ELEMENT
C
C     SRESID -  stores the static residual at time t+dt
C     SVARS  -  In 1-6, contains the static residual at time t
C               upon entering the routine. SRESID is copied to
C               SVARS(1-6) after the dynamic residual has been
C               calculated.
C            -  For half-increment residual calculations: In 7-12,
C               contains the static residual at the beginning
C               of the previous increment. SVARS(1-6) are copied
C               into SVARS(7-12) after the dynamic residual has
C               been calculated.
C
C **************************************************************
C                  Parameters
C **************************************************************
      qRubber=PROPS(1)          !Yield strength of rubber
      uy=PROPS(2)               !Horizontal yield displacement
      G=PROPS(3)                !Shear modulus of rubber
      Kbulk=PROPS(4)            !Bulk modulus of rubber
      D1=PROPS(5)               !Internal diameter
      D2=PROPS(6)               !External diameter
      t=PROPS(7)                !Thickness of single shim plate
      ts=PROPS(8)               !Thickness of single rubber layer
      kc=PROPS(9)               !Cavitation parameter
      phi=PROPS(10)             !Maximum allowable damage in the rubber
      ac=PROPS(11)              !Strength degradation parameter
      sDratio=PROPS(12)         !Shear distance
      m=PROPS(13)               !Mass of the bearing
      cd=PROPS(14)              !viscous damping parameter
      tc=PROPS(15)              !Cover thickness
```

```fortran
      n=JPROPS(1)                !Total number of rubber layers
C
C *************************************************************
C              Derived parameters  k
C *************************************************************
C     Geometric and material parameters
      AM=m/TWO                           !Mass at one node
      rd=D2/D1                            !Diameter ratio
      Tr=n*t                             !Total rubber thickness
      h=n*t+(n-1)*ts                     !Height of rubber and steel shims
      L=n*t+(n-1)*ts                   !Total height of bearing
      Ar=pi*((D2+0.5*tc)**2-D1**2)/4         !Initial bonded area
      I=(pi/64)*((D2+0.5*tc)**4-D1**4)       !Moment of interia
      S=(D2-D1)/(4.0*t)                  !Shape factor
      F=(rd*rd+1.0)/((rd-1.0)*(rd-1.0)) + (1.0+rd)/((1.0-rd)*LOG(rd))   !Dia mod factor
      rg=SQRT(I/Ar)                       !Radius of gyration
C     Elastic stiffness
      Ecp=1.0/((1.0/(6.0*G*S*S*F))+(4.0/(3.0*Kbulk))) !Compression modulus
      E=3*G                              !Elastic modulus
      Kv0 = (Ar*Ecp/Tr)                    !Pre-cavitation stiffness
C     Cavitation parameters
      Fc=3*G*Ar                           !Initial yield stress
      uc=Fc/Kv0                          !intial yield strain
C     Buckling parameters
      Er=Ecp/3                           !Rotation modulus
      As=Ar*h/Tr                         !Adjusted shear area of bearing
      Is=I*h/Tr                          !Adjusted moment of interia
      Pe=pi*pi*Er*Is/(h**2)              !Euler buckling load of bearing
      Fcr=-SQRT(Pe*G*As)                 !Critical buckling load in compression
      ucr=Fcr/Kv0                        !Critical buckling deformation
C     Horizontal motion
      qYield=qRubber                     !Yield strength of bearing(approximation)
      ke=G*Ar/Tr                          !Elastic modulus of rubber
      k0=qRubber/uy                      !Initial stiffness of hysteretic(approx)
C     Rotation
      Kr=Er*Is/Tr
C     Torsion
      Kt=G*(2*Is)/Tr
C
C
      PRINT*, 'G', G, 'Kbulk', Kbulk, 'Ar', Ar, 'Tr', Tr,
     *          'h', h, 'S', S, 'F', F
      PRINT*, 'Horizontal Motion:'
      PRINT*, 'k0', k0, 'ke', ke, 'qYield', qYield
      PRINT*, 'Vertical Motion:'
      PRINT*, 'Ecp', Ecp, 'E', E, 'rg', rg, 'Kpre', Kv0,
     *        'Fc', Fc, 'uc', uc, 'Fcr', Fcr, 'ucr', ucr
C
C *************************************************************
C              Initialize AMATRX, RHS, SRESID
C *************************************************************
C
      DO K1 = 1, NDOFEL
        SRESID(K1) = ZEROF
        DO KRHS = 1, NRHS
```

```
            RHS(K1,KRHS) = ZERO
          END DO
          DO K2 = 1, NDOFEL
            AMATRX(K2,K1) = ZERO
          END DO
        END DO
C
C
        DO K1=1,12
          DO K2=1,12
            Kg(K1,K2)=ZERO
          ENDDO
        ENDDO
C
        DO K1=1,12
          Fg(K1)=ZERO
        ENDDO
C **************************************************************
C                Analysis cases not considered here
C **************************************************************
        IF (NDLOAD.NE.0) THEN
        WRITE(7,*) 'Error:Element loads not implemented'
        CALL XIT
        END IF
C
        IF (NRHS.EQ.2) THEN
        WRITE(7,*) 'Error:Riks solution not supported by element'
        CALL XIT
        END IF
C
C **************************************************************
C                Analysis Cases
C **************************************************************
C
C

        IF (LFLAGS(3).EQ.1) THEN
        Call Transformation
        Call ForceStiffness
C         Normal incrementation
          IF (LFLAGS(1).EQ.1 .OR. LFLAGS(1).EQ.2) THEN
C             *STATIC
            AMATRX(1:12,1:12) = Kg
            IF (LFLAGS(4).NE.0) THEN
              SRESID=MATMUL(Kg,DU(1:12,1))
              RHS(1:12,1)=RHS(1:12,1)-SRESID
            ELSE
              SRESID=Fg
              RHS(1:12,1)=RHS(1:12,1)-SRESID
            ENDIF
          ELSEIF (LFLAGS(1).EQ.11 .OR. LFLAGS(1).EQ.12) THEN
C             *DYNAMIC
            ALPHA=PARAMS(1)
            BETA=PARAMS(2)
            GAMMA=PARAMS(3)
```

```
C
            DADU = ONE/(BETA*DTIME**2)
            DVDU = GAMMA/(BETA*DTIME)
C
            DO K1=1,NDOFEL
                AMATRX(K1,K1)=AM*DADU
                RHS(K1,1) = RHS(K1,1)-AM*A(K1)
            ENDDO
            AMATRX(1:12,1:12)=AMATRX(1:12,1:12)+(ONE+ALPHA)*Kg
            SRESID=Fg
            RHS(1:12,1)=RHS(1:12,1)
     *                          -((ONE+ALPHA)*SRESID-ALPHA*SVARS(1:12))
            DO K1=1, NDOFEL
                SVARS(K1+12)=SVARS(K1)
                SVARS(K1)   =SRESID(K1)
            ENDDO
        ENDIF
      ELSEIF (LFLAGS(3).EQ.2) THEN
C       Stiffness Matrix
        AMATRX(1:12,1:12) = Kg
C     ELSEIF (LFLAGS(3).EQ.3) THEN  !For damping case
C       Not required if material of bearing do not provide damping
      ELSEIF (LFLAGS(3).EQ.4) THEN
C       Mass Matrix
        DO K1=1, NDOFEL
            AMATRX(K1,K1)= AM
        ENDDO
      ELSEIF (LFLAGS(3).EQ.5) THEN
C       Half-increment residual calculation
        ALPHA = PARAMS(1)
        SRESID=Fg
        RHS(1:12,1)=RHS(1:12,1)-AM*A-(ONE+ALPHA)*SRESID
     *            +HALF*ALPHA*(SVARS(1:12)+SVARS(13:24))
      ELSEIF (LFLAGS(3).EQ.6) THEN
C       Initial acceleration calculation
        DO K1=1, NDOFEL
            AMATRX(K1,K1)= AM
        ENDDO
        SRESID=Fg
        RHS(1:12,1)=RHS(1:12,1)-SRESID
        DO K1=1,NDOFEL
            SVARS(K1)=SRESID(K1)
        ENDDO
      ELSEIF (LFLAGS(3).EQ.100) THEN
C       Output for perturbations
C        AMATRX(1:12,1:12) = Kg
        IF (LFLAGS(1).EQ.1 .OR. LFLAGS(1).EQ.2) THEN
C           *STATIC
            SRESID=MATMUL(Kg,DU(1:12,1))
            RHS(1:12,1)=RHS(1:12,1)-SRESID
            DO KVAR = 1, NSVARS
            SVARS(KVAR) = ZERO
            END DO
            SVARS(1:12) = RHS(1:12,1)
        ELSEIF (LFLAGS(1).EQ.41) THEN
```

```
C            *FREQUENCY
             DO KRHS=1,NHRS
                 SRESID=MATMUL(Kg,DU(:,KRHS))
                 RHS(1:12,KRHS)=RHS(1:12,KRHS)-SRESID
             ENDDO
             DO KVAR=1,NSVARS
                 SVARS(KVAR)=ZERO
             ENDDO
             SVARS(1:12)=RHS(1:12,1)
         ENDIF
       ENDIF
C
C ***************************************************************
C                            SUBROUTINES
C ***************************************************************
       CONTAINS
C----------------------------------------------------------------
C            Subroutine for transformation matrices
C----------------------------------------------------------------
         SUBROUTINE Transformation
             REAL r, TEMP(3,3)
             r=SQRT((COORDS(1,2)-COORDS(1,1))**2+(COORDS(2,2)-
     *           COORDS(2,1))**2+(COORDS(3,2)-COORDS(3,1))**2)
C            Local to basic transformation matrix: Tlb
             DO K1=1, 6
                 DO K2=1,12
                 Tlb(K1,K2)=ZERO
                 ENDDO
             ENDDO
C            Populate the Tlb matrix
             DO K1=1, 6
                 Tlb(K1,K1)=-1.0
                 Tlb(K1,K1+6)=1.0
             ENDDO
             Tlb(2,6) = -sDratio*L;
             Tlb(2,12) = -(1.0 - sDratio)*L;
             Tlb(3,5) = sDratio*L;
             Tlb(3,11) = (1.0 - sDratio)*L;
C            Transpose of Tlb
             TlbT=TRANSPOSE(Tlb)
C
C            Local to basic transformation matrix: Tgl
             DO K1=1, 12
                 DO K2=1,12
                     Tgl(K1,K2)=ZERO
                 ENDDO
             ENDDO
C
             IF (ABS(COORDS(1,2)-COORDS(1,1)) .GE. 0.99*r) THEN
                 TEMP=RESHAPE((/1,0,0,0,1,0,0,0,1/),SHAPE(TEMP))
             ELSEIF (ABS(COORDS(2,2)-COORDS(2,1)) .GE. 0.99*r) THEN
                 TEMP=RESHAPE((/0,-1,0,1,0,0,0,0,1/),SHAPE(TEMP))
             ELSEIF (ABS(COORDS(3,2)-COORDS(3,1)) .GE. 0.99*r) THEN
                 TEMP=RESHAPE((/0,-1,0,0,0,-1,1,0,0/),SHAPE(TEMP))
             ELSE
```

```
                WRITE(7,*) 'Inclined orientation of
     *                           bearing is not supported'
                CALL XIT
            ENDIF
C           Populate the Tgl Matrix
            DO K1=1,4
                Tgl(3*(K1-1)+1:3*(K1-1)+3,3*(K1-1)+1:3*(K1-1)+3)=TEMP
            ENDDO
C           Transpose of Tgl
            TglT=TRANSPOSE(Tgl)
        END SUBROUTINE
C---------------------------------------------------------------------------
C               Subroutine for forces and stiffness matrices
C---------------------------------------------------------------------------
        SUBROUTINE ForceStiffness
            REAL ub(6), vb(6), dub(6), fb(6), Kb(6,6)
            REAL uh, vh, Delta, Ard, ratio
            REAL z(2), zC(2), dzdu(2,2), fbc(2), delta_z(2), Df(2,2)
            REAL DBL, tol
            REAL beta, gamma, zNrm, zCNrm, delta_zNrm, tmp1, tmp2, tmp3
            INTEGER iter, maxIter
            DO K1=1,6
                fb(K1)=ZERO
                ub(K1)=ZERO
                vb(K1)=ZERO
                dub(K1)=ZERO
                DO K2=1,6
                    Kb(K1,K2)=ZERO
                ENDDO
            ENDDO
            DO K1=1,2
                z(K1)=ZERO
                zC(K1)=ZERO
                delta_z(K1)=ZERO
                DO K2=1,2
                    dzdu(K1,K2)=ZERO
                    Df(K1,K2)=ZERO
                ENDDO
            ENDDO
            ub=MATMUL(Tlb,MATMUL(Tgl,U))
            vb=MATMUL(Tlb,MATMUL(Tgl,V))
            dub=MATMUL(Tlb,MATMUL(Tgl,DU(1:12,1)))
            zC=SVARS(26:27)
            z=SVARS(26:27)
            ke=SVARS(28)
            uh=SQRT(ub(2)**2+ub(3)**2)
            vh=SQRT(vb(2)**2+vb(3)**2)
            zCNrm=SQRT(zC(1)**2+zC(2)**2)
C           Update parameters
            Kv=Kv0*(1.0/(1.0+(3.0/pi**2)*((uh/rg)**2)))
            IF (uh .GT. ZERO) THEN
            uc=Fc/Kv
            ENDIF
            umax = MAX(uc, SVARS(25))
            Fmax=Fc*(1.0+(1.0/(Tr*kc))*(1-EXP(-kc*(umax-uc))))
```

```fortran
          Fcn=Fc*(1-phi*(1-EXP(-ac*(umax-uc)/uc)))
          ucn=Fcn/Kv
          Delta = 2*acos(uh/D2);
        Ard=(D2*D2/4)*(Delta-sin(Delta))
        ratio=Ard/Ar
        IF (ratio .GT. 0.2) THEN
            Fcrn=Fcr*Ard/Ar
        ELSE
            Fcrn=0.2*Fcr
        ENDIF
        ucr = Fcrn/Kv
C
C
C         axial force and stiffness in basic x direction
          IF (ub(1) .LE. ucr) THEN
              Kb(1,1)=Kv/1000
              fb(1)=Fcrn+Kb(1,1)*(ub(1)-ucr)
          ELSEIF (ub(1) .LE. ucn) THEN
              Kb(1,1)=Kv
              fb(1)=Kb(1,1)*ub(1)
          ELSEIF (ub(1) .LE. umax) THEN
              Kb(1,1)=(Fmax-fcn)/(umax-ucn)
              fb(1)=Fcn+Kb(1,1)*(ub(1)-ucn)
          ELSE
              Kb(1,1)=(Fc/Tr)*EXP(-kc*(ub(1)-uc))
              fb(1)=Fc*(1.0+(1.0/(Tr*kc))*(1-EXP(-kc*(ub(1)-uc))))
              umax=ub(1)
          ENDIF
C         Shear forces and stiffnesses in basic y and z direction
          DBL=1.0E-32
          IF (SQRT(dub(2)**2+dub(3)**2) .GT. ZERO) THEN
C         Calculate hysteretic evolution parameter z using Newton-Raphson
          iter=0
          maxIter=100
          tol=1E-7
          beta=0.1
          gamma=0.9
C
10        CONTINUE
          zNrm=SQRT(z(1)**2+z(2)**2)
          IF (zNrm .LE. DBL) THEN
              zNrm=DBL
          ENDIF
          tmp1=SIGN(gamma, z(1)*dub(2))+beta
          tmp2=SIGN(gamma, z(2)*dub(3))+beta
          tmp3=z(1)*dub(2)*tmp1+z(2)*dub(3)*tmp2
C         function and derivative
          fbc(1)= z(1)-zC(1)-(1.0/uy)*(dub(2)-z(1)*tmp3)
          fbc(2)= z(2)-zC(2)-(1.0/uy)*(dub(3)-z(2)*tmp3)
C
          Df(1,1)=1.0+(1.0/uy)*(2*z(1)*dub(2)*tmp1+z(2)*dub(3)*tmp2)
          Df(2,1)=(tmp1/uy)*z(2)*dub(2)
          Df(1,2)=(tmp2/uy)*z(1)*dub(3)
          Df(2,2)=1.0+(1.0/uy)*(z(1)*dub(2)*tmp1+2*z(2)*dub(3)*tmp2)
C
```

```
C           Issue wanring if diagonal of derivative Df is zero
          IF ((ABS(Df(1,1)) .LE. DBL) .OR. (ABS(Df(2,2)) .LE. DBL)) THEN
            WRITE(7,*) 'WARNING: LeadRubberX -zero Jacobian in Newton-
     *          Raphson scheme for hysteretic evolution parameter z'
            CALL XIT
            END IF
C
C           advance one step
            delta_z(1)=(fbc(1)*Df(2,2)-fbc(2)*Df(1,2))
     *                      /(Df(1,1)*Df(2,2)-Df(1,2)*Df(2,1))
            delta_z(2) = (fbc(1)*Df(2,1)-fbc(2)*Df(1,1))
     *                      /(Df(1,2)*Df(2,1)-Df(1,1)*Df(2,2))
            z = z-delta_z
            iter = iter+1
            delta_zNrm=SQRT(delta_z(1)**2+delta_z(2)**2)
           IF ((delta_zNrm .GE. tol) .AND. (iter .LT. maxIter)) GOTO 10

C
C           Issue wanring if Newton-Raphson scheme did not converge
            IF (iter .GE. maxIter) THEN
            WRITE(7,*) 'WARNING: LeadRubberX -The Newton-Raphson Scheme
     *          for z did not converge after ', iter, ' iterations
     *          and', delta_zNrm, 'Norm'
            CALL XIT
            END IF
C
C           // get derivative of hysteretic evolution parameter
            delta_z = z-zC;
            IF (ABS(dub(2)) .GT. ZERO)  THEN
            dzdu(1,1) = delta_z(1)/dub(2)
            dzdu(2,1) = delta_z(2)/dub(2)
            ENDIF
            IF (ABS(dub(3)) .GT. ZERO)  THEN
            dzdu(1,2) = delta_z(1)/dub(3)
            dzdu(2,2) = delta_z(2)/dub(3)
            ENDIF
C           set shear force
            fb(2)=cd*vb(2)+qYield*z(1)+ke*ub(2)
            fb(3)=cd*vb(3)+qYield*z(2)+ke*ub(3)
C           set tangent stiffness
            Kb(2,2) = cd/DTIME+qYield*dzdu(1,1) + ke
            Kb(2,3) = qYield*dzdu(1,2)
            Kb(3,2) = qYield*dzdu(2,1)
            Kb(3,3) = cd/DTIME+qYield*dzdu(2,2) + ke
C
            ELSE
            fb(2)=cd*vb(2)+qYield*z(1)+ke*ub(2)
            fb(3)=cd*vb(3)+qYield*z(2)+ke*ub(3)
C           set tangent stiffness
            Kb(2,2) = cd/DTIME+qYield*dzdu(1,1) + ke
            Kb(2,3) = qYield*dzdu(1,2)
            Kb(3,2) = qYield*dzdu(2,1)
            Kb(3,3) = cd/DTIME+qYield*dzdu(2,2) + ke
            ENDIF
            ke=(G*Ar/Tr)*(1-(fb(1)/Fcrn)**2)
```

```
C
C           get moment and stiffness in torsion about x, rotation about y, z
            Kb(4,4)=Kt
            Kb(5,5)=Kr
            Kb(6,6)=Kr
            fb(4)=Kb(4,4)*ub(4)
            fb(5)=Kb(5,5)*ub(5)
            fb(6)=Kb(6,6)*ub(6)
C           global stiffness matrix
            Kg=MATMUL(MATMUL(MATMUL(MATMUL(TglT,TlbT),Kb),Tlb),Tgl)
C           global force matrix
            Fg=MATMUL(TglT,MATMUL(TlbT,fb))
            SVARS(25)=umax
            SVARS(26:27)=z
            SVARS(28)=ke
        END SUBROUTINE
      END
```