APPENDIX R: User Defined Cohesive Model

The addition of a user cohesive material subroutine into LS-DYNA is relatively simple. The UMATiC subroutine is called every time step where i ranges from 41 to 50. Input for the material model follows the *MAT_USER_DEFINED_MATERIAL definition. The user has the option of providing either a scalar or vectorized subroutine. As discussed in the Remarks for the user-defined material, the first two material parameters are reserved to specify how the density is treated and the number of integration points required for the failure of the element.

The cohesive model calculates the tractions on the mid-surface of the element as a function of the differences of the displacements and velocities of the upper (defined by nodes 5-6-7-8) and lower surfaces (defined by nodes 1-2-3-4). The displacements, velocities, and the calculated tractions are in the local coordinate system of the element, where the first two components of the vectors are in the plane of the mid-surface and the third component is normal to the mid-surface.

A stiffness must also be calculated by the user for the explicit time step calculation in LS-DYNA. This stiffness must provide an upper bound on the stiffness in all three directions.

The material fails at an integration point when `ifail=.true`. For an element to be deleted from the calculation, the number of integration points specified by the second material parameter must fail. If the second parameter is zero, elements cannot fail regardless of the specification of IFAIL in the user-defined material input. For example, the user may choose to reject an implicit step is the displacement increment is too

For implicit analysis, the subroutine is called with `maketan=.true.` and the user must provide the elastic moduli in the three local directions in the respective diagonal terms of the `dsave` array. The parameter `reject`, if set to `.true.` by the user, will signal to LS-DYNA that the current implicit iteration is unacceptable. For example, the user may choose to reject an implicit step if the traction changes too much from the last time step. In this situation, LS-DYNA will print a warning message `Material model rejected current iterate' and retry the step with a smaller time step. If chosen carefully (by way of experimenting), this may result in a good trade-off between the number of implicit iterations per step and the step size for overall speed.

The following example is a vectorized model with two elastic constants and failure:

```
      subroutine umat41c(idpart,cm,lft,llt,fc,dx,dxdt,aux,ek,
     & ifail,dt1siz,crv,nnpcrv,nhxbwp,cma,maketan,dsave,ctmp,elsiz,
     & reject,ip,nip)
      include 'nlqparm'
c
c***  vector cohesive material user model example
c
```

```
c***  variables
c          idpart ---- Part ID
c          cm -------- material constants
c          lft,llt --- start and end of block
c          fc -------- components of the cohesive force
c          dx -------- components of the displacement
c          dxdt ------ components of the velocity
c          aux ------- history storage
c          ek -------- max.  stiffness/area for time step calculation
c          ifail ----- =.false.  not failed
c                      =.true.  failed
c          dt1siz ---- time step size
c          crv ------- curve array
c          nnpcrv ---- # points per curve for crv array
c          nhxbwp ---- internal element id array, lqfinv(nhxbwp(i),2)
c                      gives external element id
c          cma ------- additional memory for material data defined by
c                      LMCA in 2nd card, 6th field of *MAT_USER_DEFINED
c          maketan --- true for implicit
c          dsave ----- material stiffness array, define for implicit
c          ctmp ------ current temperature
c          elsiz ----- characteristic element size (=sqrt(area))
c          reject ---- set to .true.  if this implicit iterate is
c                      to be rejected for some reason (implicit only)
c          ip -------- integration point number
c          nip ------- total number of integration points
c
c***  dx, dxdt, and fc are in the local coordinate system:
c     components 1 and 2 are in the plane of the cohesive surface
c     component 3 is normal to the plane
c
c***  cm storage convention
c     (1) =0 density is per area
c         =1 density is per volume
c     (2) number of integration points for element deletion
c         =0 no deletion
c     (3:48) material model constants
c
      logical ifail,maketan,reject
      dimension cm(*),fc(nlq,*),dx(nlq,*),dxdt(nlq,*),
     &          aux(nlq,*),ek(*),ifail(*),dt1siz(*),crv(101,2,*),
     &          nhxbwp(*),cma(*),dsave(nlq,6,*),ctmp(*),elsiz(*)
      integer nnpcrv(*)
c
c
      et=cm(3)
      en=cm(4)
      eki=max(et,en)
      fcfail=cm(5)
c
      do i=lft,llt
        fc(i,1)=et*dx(i,1)
        fc(i,2)=et*dx(i,2)
        fc(i,3)=en*dx(i,3)
        ek(i)=eki
        ifail(i)=fc(i,3).gt.fcfail
      enddo
c
      if(maketan) then
        do i=lft,llt
          dsave(i,1,1)=et
          dsave(i,2,1)=0.
          dsave(i,3,1)=0.
          dsave(i,1,2)=0.
          dsave(i,2,2)=et
          dsave(i,3,2)=0.
          dsave(i,1,3)=0.
          dsave(i,2,3)=0.
          dsave(i,3,3)=en
        enddo
      endif
      return
      end
```

The second example implements the Tveergard-Hutchinson cohesive model with failure in both the vectorized (UMAT42C) and scalar (UMAT43C) forms. Note the LFT and LLT are passed to the scalar version, however their value is zero.

```
      subroutine umat42c(idpart,params,lft,llt,fTraction,jump_u,dxdt,
     & aux,ek,ifail,dt1siz,crv,nnpcrv,nhxbwp,cma,maketan,dsave,ctmp,elsiz,
     & reject,ip,nip)
      include 'nlqparm'
c
c***  vector cohesive material user model example
c
c     Tveergard-Hutchinson model based on:
c     tahoe/src/elements/cohesive_surface/cohesive_models/TvergHutch3DT.cpp
c
c     the declaration below is processed by the C preprocessor and
c     is real*4 or real*8 depending on whether LS-DYNA is single or double
c     precision
c
      REAL L,jump_u
      logical ifail,maketan,reject
      dimension params(*),fTraction(nlq,*),jump_u(nlq,*),dxdt(nlq,*),
     &          aux(nlq,*),ek(*),ifail(*),dt1siz(*),crv(101,2,*),
     &          nhxbwp(*),cma(*),dsave(nlq,6,*),ctmp(*),elsiz(*)
      integer nnpcrv(*)
c
      fsigma_max=params(3)
      fd_c_n=params(4)
      fd_c_t=params(5)
      fL_1=params(6)
      fL_2=params(7)
      fpenalty=params(8)
c
      fK=fpenalty*fsigma_max/(fL_1*fd_c_n)
c
      fac=min(fd_c_n/fd_c_t**2,1./fd_c_n)
c
      do i=lft,llt
      u_t1 = jump_u(i,1)
      u_t2 = jump_u(i,2)
      u_n = jump_u(i,3)
c
      r_t1 = u_t1/fd_c_t
      r_t2 = u_t2/fd_c_t
      r_n = u_n/fd_c_n
      L = sqrt(r_t1*r_t1 + r_t2*r_t2 + r_n*r_n)
c
      if (L .lt.  fL_1) then
         sigbyL=fsigma_max/fL_1
      else if (L .lt.  fL_2) then
         sigbyL = fsigma_max/L
      else if (L .lt.  1.) then
         sigbyL = fsigma_max*(1. - L)/(1. - fL_2)/L
      else
         sigbyL = 0.0
         ifail(i)=.true.
      endif
c
      fTraction(i,1) = sigbyL*r_t1*(fd_c_n/fd_c_t)
      fTraction(i,2) = sigbyL*r_t2*(fd_c_n/fd_c_t)
      fTraction(i,3) = sigbyL*r_n
c
c     penetration
      if (u_n .lt.  0) fTraction(i,3)=fTraction(i,3)+fK*u_n
c
c     approximate stiffness for time step
      if (u_n .lt.  0) then
        ek(i)=fac*sigbyL+fK
      else
        ek(i)=fac*sigbyL
      endif
c
      if (maketan) then
        dsave(i,1,1)=sigbyL/fd_c_t*(fd_c_n/fd_c_t)
        dsave(i,2,1)=0.
        dsave(i,3,1)=0.
        dsave(i,1,2)=0.
        dsave(i,2,2)=sigbyL/fd_c_t*(fd_c_n/fd_c_t)
        dsave(i,3,2)=0.
        dsave(i,1,3)=0.
        dsave(i,2,3)=0.
        dsave(i,3,3)=sigbyL/fd_c_n
        if (u_n.lt.0) dsave(i,3,3)=dsave(i,3,3)+fk
      endif
      enddo
c
      return
      end
c
c
c
      subroutine umat43c(idpart,params,lft,llt,fTraction,jump_u,dxdt,
     & aux,ek,ifail,dt1siz,crv,nnpcrv,nhxbwp,cma,maketan,dsave,ctmp,elsiz,
     & reject,ip,nip)
c
c***  scalar cohesive material user model example
c
c     Tveergard-Hutchinson model based on:
c     tahoe/src/elements/cohesive_surface/cohesive_models/TvergHutch3DT.cpp
c
c     the declaration below is processed by the C preprocessor and
c     is real*4 or real*8 depending on whether LS-DYNA is single or double
c     precision
c
      REAL L,jump_u
      logical ifail,maketan,reject
      dimension params(*),fTraction(nlq,*),jump_u(nlq,*),dxdt(nlq,*),
     &          aux(nlq,*),ek(*),ifail(*),dt1siz(*),crv(101,2,*),
     &          nhxbwp(*),cma(*),dsave(nlq,6,*),ctmp(*),elsiz(*)
      integer nnpcrv(*)
c
      fsigma_max=params(3)
      fd_c_n=params(4)
      fd_c_t=params(5)
      fL_1=params(6)
      fL_2=params(7)
      fpenalty=params(8)
```

```
c
      fK=fpenalty*fsigma_max/(fL_1*fd_c_n)
c
      fac=min(fd_c_n/fd_c_t**2,1./fd_c_n)
c
      u_t1 = jump_u(1)
      u_t2 = jump_u(2)
      u_n = jump_u(3)
c
      r_t1 = u_t1/fd_c_t
      r_t2 = u_t2/fd_c_t
      r_n = u_n/fd_c_n
      L = sqrt(r_t1*r_t1 + r_t2*r_t2 + r_n*r_n)
c
      if (L .lt.  fL_1) then
         sigbyL=fsigma_max/fL_1
      else if (L .lt.  fL_2) then
         sigbyL = fsigma_max/L
      else if (L .lt.  1.) then
         sigbyL = fsigma_max*(1. - L)/(1. - fL_2)/L
      else
         sigbyL = 0.0

      ifail=.true.
      endif
c
      fTraction(1) = sigbyL*r_t1*(fd_c_n/fd_c_t)
      fTraction(2) = sigbyL*r_t2*(fd_c_n/fd_c_t)
      fTraction(3) = sigbyL*r_n
c
c     penetration
      if (u_n .lt.  0) fTraction(3)=fTraction(3)+fK*u_n
c
c     approximate stiffness for time step
      if (u_n .lt.  0) then
        ek=fac*sigbyL+fK
      else
        ek=fac*sigbyL
      endif
c
      if (maketan) then
        dsave(1,1)=sigbyL/fd_c_t*(fd_c_n/fd_c_t)
        dsave(2,1)=0.
        dsave(3,1)=0.
        dsave(1,2)=0.
        dsave(2,2)=sigbyL/fd_c_t*(fd_c_n/fd_c_t)
        dsave(3,2)=0.
        dsave(1,3)=0.
        dsave(2,3)=0.
        dsave(3,3)=sigbyL/fd_c_n
        if (u_n.lt.0) dsave(3,3)=dsave(3,3)+fk
      endif
      return
      end
```