

APPENDIX S: User Defined Boundary Flux

A user defined boundary flux interface is provided in LS-DYNA where it is possible to define the thermal heat flux (power per surface area) in or out of a surface segment as an arbitrary function of temperature and history. The user may associate history variables with each individual flux interface and also use load curves.

The user flux interface is invoked using the keyword `*BOUNDARY_FLUX_OPTION`. This is accomplished with the parameter `NHISV`. When it is defined with a value greater than 0, the user subroutine

```
subroutine usrflux(f1,flp,...)
```

is called to compute the flux (`f1`) defined as heat (energy) per time and per surface area.

Other parameters that are passed to the user flux subroutine include the segment nodal temperatures at the previous (T_0) and current time (T_1), the segment nodal coordinates and the time integration parameter α . Also, the current thermal simulation time t , the time step Δt and average segment temperature (T_α) at time $t+\alpha\Delta t$ is provided together with the curve array for accessing defined load curves in the keyword input file. For computing load curve values, note that load curve IDs need to be transformed to internal numbers or the subroutine `crvval` should be used, see the appendix on user defined materials for details.

The segment coordinates available in the subroutine are such that the outward normal vector follows the well-known right-hand rule, thus segments corresponding to the lower surface of thick thermal shells are reversed before passed to the subroutine. For shells in general, the segment connectivity should follow the connectivity of the actual shell element to avoid problems.

Optionally, the user may define the derivative of the flux `f1` with respect to the average segment temperature (T_α) at time $t+\alpha\Delta t$, `flp`. This value is used in the nonlinear thermal solver for assembling the correct stiffness matrix and must be set by the user. If possible, it is recommended to use a value that reflects the nonlinearity of the flux model, otherwise the value 0 should be used.

An array of history variables, identical with the input parameters defined in the keyword input file, are passed to the subroutine that can be updated with time or kept constant throughout the simulation. An example of usage would be to integrate the flux with time to keep track of the dissipated energy per surface area in order to simulate the effects of spray cooling in hot-stamping.

```
subroutine usrflux(f1,flp,x,tnpl,tnl,nodes,
```

*APPENDIX S

```
.      alpha,atime,atemp,dt,time,fhsv,nfhsv,crv)
C*****
C|  LIVERMORE SOFTWARE TECHNOLOGY CORPORATION   (LSTC)
C|  -----
C|  COPYRIGHT © 2007 JOHN O. HALLQUIST, LSTC
C|  ALL RIGHTS RESERVED
C*****
c
c      User subroutine for boundary thermal flux
c
c      Purpose:  To define thermal flux parameter (heat per surface area and
c                time)
c
c      Variables:
c
c      fl          = flux intensity (output)
c      flp         = flux intensity derivative wrt atemp (output)
c      x(3,nodes)  = global segment coordinates (input)
c      tnpl(nodes) = temperatures at time time (input)
c      tnl(nodes)  = temperatures at time time-dt (input)
c      nodes       = number of nodes in segment (3,4 or 6) (input)
c      alpha       = time integration parameter (input)
c      atime       = time+(alpha-1)*dt
c      atemp       = average segment temperature at time atime
c      dt          = time step size (input)
c      time        = time at which the new temperature is sought (input)
c      fhsv(nfhsv) = flux history variables (input/output)
c      nfhsv       = number of flux history variables for this segment
c                  (input)
c      crv         = curve array (input)
c
c      include 'nlqparm'
c      dimension x(3,*),tnpl(*),tnl(*)
c      dimension fhsv(*),crv(lq1,2,*)
c
c      Define flux by linear convection
c      that optionally decays (in an ad-hoc way) as power
c      dissipates from surface
c
c      fhsv(1) = convection coefficient
c      fhsv(2) = ambient temperature
c      fhsv(3) = total amount of energy per surface area available
c      fhsv(4) = dissipated energy per surface area at current
c
c      hcon=fhsv(1)
c      tinf=fhsv(2)
c      flin=hcon*(tinf-atemp)
c      if (nfhsv.gt.2) then
c        q=(1.-fhsv(4)/fhsv(3))/
c          (1.+5*dt*flin/fhsv(3))
c        flp=-q*hcon
c        if (q.gt.1.) then
c          q=1.
c          flp=-hcon
c        elseif (q.lt.0.) then
c          q=0.
c          flp=0.
c        endif
c        fl=q*flin
c        fhsv(4)=fhsv(4)+dt*.5*fl
c        fhsv(4)=min(fhsv(3),fhsv(4))
c      else
c        fl=flin
c        flp=-hcon
c      endif
c
```

```
return  
end
```

