APPENDIX B: User-Defined Equation-of-State

You can supply your own subroutines to define equation-of-state (EOS) models in LS-DYNA. To invoke a user-defined EOS, you must

1. Write a user EOS subroutine called by the LS-DYNA user-defined EOS interface.
2. Create a custom executable that includes the EOS subroutine.
3. Invoke that subroutine by defining a part in the keyword input deck that uses *EOS_USER_DEFINED with the appropriate input parameters.

We provide subroutine `ueoslib` and sample subroutines `ueos21s` and `ueos21v` in Fortran source files. This text serves as an introductory guide to implementing such a model. Note that the names of variables and subroutines below may differ from the actual ones depending on the platform and the current version of LS-DYNA.

General overview

When you define *EOS_USER_DEFINED for a part, LS-DYNA calls the subroutine `ueoslib` with the appropriate input data for the EOS update. LS-DYNA calls this subroutine twice for each integration point in each element. The first call requires the EOS to calculate the bulk modulus, and the second call updates the pressure and internal energy. You may modify this routine if necessary. This routine initializes the following data structures used by a specific *scalar* material subroutine:

`iflag` – mode flag
- EQ.-1: for initializing EOS constants
- EQ.0: for calculating the bulk modulus
- EQ.1: for the pressure and energy update

`cb` – bulk modulus
`pnew` – the new pressure
`rho0` – reference density
`hist` – array of user-defined history variables, NHV in length
`specen` – internal energy per unit reference volume
`df` – volume ratio, V/V_0
`v0` – the initial volume
`dvol` – volume increment
`pc` – pressure cut-off

If you activate the *vectorization* flag (IVECT = 1) on the EOS card, this routine generally stores these variables in vector blocks of length `nlq`, with vector indices ranging from

lft to llt, which allows for more efficient execution of the EOS routine. The data structures mentioned above for the vectorized case are:

cb(nlq)	–	bulk modulus
pnew(nlq)	–	the new pressure
hist(nlq,*)	–	array of user-defined history variables with NHV columns
specen(nlq)	–	internal energy per unit reference volume
df(nlq)	–	volume ratio, V/V0
v0(nlq)	–	the initial volume
dvol(nlq)	–	volume increment
pc(nlq)	–	pressure cut-off

The value of nlq is set as a parameter in the include file nlqparm, included at the top of the subroutine, and varies between machines and operating systems. Each entry in a vector block is associated with an element in the finite element mesh for a fixed integration point. The number of entries in the history variables array (indicated by * above) matches the number of history variables requested on the material card (NHV). All history variables are initially zero and are initialized within the user-define EOS subroutine on the first time step, when the logical variable first, passed through the argument list, is .TRUE. Furthermore, all user-defined EOS models require a bulk modulus, cb, for transmitting boundaries, contact interfaces, rigid body constraints, and time step calculations. In addition to the variables mentioned above, the following data can be supplied to the user-defined material routines, regardless of whether vectorization is used or not:

eosp(*)	–	array of material constants from the input file
tt	–	current time
crv(lq1,2,*)	–	array representation of curves defined in the keyword deck.

A user-defined EOS subroutine, ueos*XX*s in the scalar case or ueos*XX*v in the vector case, will be called for parts that point to *EOS_USER_DEFINED in the input deck. The letters *XX* represent a number between 21 and 30 that matches the input variable EOST in the *EOS_USER_DEFINED keyword. During the initialization phase, LS-DYNA calls the user-define EOS subroutine with iflag set to -1 to permit the initialization of constants in the user-defined EOS subroutine. Although fewer than 48 constants may be read into the array eosp during the input, you may use all 48 within the EOS subroutines. The user-defined subroutine should calculate the bulk modulus when iflag = 0 and update the pressure, internal energy, and history variables when iflag = 1.

See Appendix A for a discussion about using curves (*DEFINE_CURVE).

Example and Discussion

Here we provide a sample scalar user-defined subroutine for a Gruneisen EOS. Its vector counterpart immediately follows it. We close this section with a discussion about implementation and common pitfalls.

Scalar subroutine

```
      subroutine ueos21s(iflag,cb,pnew,hist,rho0,eosp,specen,
     &                   df,dvol,v0,pc,dt,tt,crv,nnpcrv,first)
      include 'nlqparm'
c
c***  example scalar user implementation of the Gruneisen EOS
c
c***  variables
c          iflag ----- =0 calculate bulk modulus
c                      =1 update pressure and energy
c          cb -------- bulk modulus
c          pnew ------ new pressure
c          hist ------ history variables
c          rho0 ------ reference density
c          eosp ------ EOS constants
c          specen ---- energy/reference volume
c          df -------- volume ratio, v/v0 = rho0/rho
c          dvol ------ change in volume over a time step
c          v0 -------- reference volume
c          pc -------- pressure cut-off
c          dt -------- time step size
c          tt -------- current time
c          crv ------- curve array
c          nnpcrv ---- number of points in each curve
c          first ----- logical .true.  for tt,crv,first time step
c                      (for initialization of the history variables)
c
      logical first
c
      dimension hist(*),eosp(*),crv(lq1,2,*)
      integer nnpcrv(*)
c
      c  =eosp(1)
      s1 =eosp(2)
      s2 =eosp(3)
      s3 =eosp(4)
      g0 =eosp(5)
      sa =eosp(6)
      s11=s1-1.
      s22=2.*s2
      s33=3.*s3
      s32=2.*s3
      sad2=.5*sa
      g0d2=1.-.5*g0
      roc2=rho0*c**2
c
c***  calculate the bulk modulus for the EOS contribution to the sound speed
      if (iflag.eq.0) then
        xmu=1.0/df-1.
        dfmu=df*xmu
        facp=.5*(1.+sign(1.,xmu))
        facn=1.-facp
        xnum=1.+xmu*(+g0d2-sad2*xmu)
        xdem=1.-xmu*(s11+dfmu*(s2+s3*dfmu))
        tmp=facp/(xdem*xdem)
        a=roc2*xmu*(facn+tmp*xnum)
        b=g0+sa*xmu
        pnum=roc2*(facn+facp*(xnum+xmu*(g0d2-sa*xmu)))
        pden=2.*xdem*(-s11 +dfmu*(-s22+dfmu*(s2-s33+s32*dfmu)))
        cb=pnum*(facn+tmp)-tmp*a*pden+sa*specen+
     &        b*df**2*max(pc,(a+b*specen))
c
c***  update the pressure and internal energy
      else
```

```
        xmu=1.0/df-1.
        dfmu=df*xmu
        facp=.5*(1.+sign(1.,xmu))
        facn=1.-facp
        xnum=1.+xmu*(+g0d2-sad2*xmu)
        xdem=1.-xmu*(s11+dfmu*(s2+s3*dfmu))
        tmp=facp/(xdem*xdem)
        a=roc2*xmu*(facn+tmp*xnum)
        b=g0+sa*xmu
        dvov0=0.5*dvol/v0
        denom=1.+ b*dvov0
        pnew=(a+specen*b)/max(1.e-6,denom)
        pnew=max(pnew,pc)
        specen=specen-pnew*dvov0
      endif
c
      return
      end
```

Vectorized routine

```
      subroutine ueos21v(lft,llt,iflag,cb,pnew,hist,rho0,eosp,specen,
     &                   df,dvol,v0,pc,dt,tt,crv,nnpcrv,first)
      include 'nlqparm'
c
c***  example vectorized user implementation of the Gruneisen EOS
c
c***  variables
c          lft,llt --- tt,crv,first and last indices into arrays
c          iflag ----- =0 calculate bulk modulus
c                      =1 update pressure and energy
c          cb -------- bulk modulus
c          pnew ------ new pressure
c          hist ------ history variables
c          rho0 ------ reference density
c          eosp ------ EOS constants
c          specen ---- energy/reference volume
c          df -------- volume ratio, v/v0 = rho0/rho
c          dvol ------ change in volume over a time step
c          v0 -------- reference volume
c          pc -------- pressure cut-off
c          dt -------- time step size
c          tt -------- current time
c          crv ------- curve array
c          nnpcrv ---- number of points in each curve
c          first ----- logical .true.  for tt,crv,first time step
c                      (for initialization of the history variables)
c
      logical first
c
      dimension cb(*),pnew(*),hist(nlq,*),eosp(*),
     &          specen(*),df(*),dvol(*),pc(*),v0(*)
      dimension crv(lq1,2,*)
      integer nnpcrv(*)
c
      c  =eosp(1)
      s1 =eosp(2)
      s2 =eosp(3)
      s3 =eosp(4)
      g0 =eosp(5)
      sa =eosp(6)
      s11=s1-1.
      s22=2.*s2
      s33=3.*s3
      s32=2.*s3
```

```
      sad2=.5*sa
      g0d2=1.-.5*g0
      roc2=rho0*c**2
c
c***  calculate the bulk modulus for the EOS contribution to the sound speed
      if (iflag.eq.0) then
        do i=lft,llt
          xmu=1.0/df(i)-1.
          dfmu=df(i)*xmu
          facp=.5*(1.+sign(1.,xmu))
          facn=1.-facp
          xnum=1.+xmu*(+g0d2-sad2*xmu)
          xdem=1.-xmu*(s11+dfmu*(s2+s3*dfmu))
          tmp=facp/(xdem*xdem)
          a=roc2*xmu*(facn+tmp*xnum)
          b=g0+sa*xmu
          pnum=roc2*(facn+facp*(xnum+xmu*(g0d2-sa*xmu)))
          pden=2.*xdem*(-s11 +dfmu*(-s22+dfmu*(s2-s33+s32*dfmu)))
          cb(i)=pnum*(facn+tmp)-tmp*a*pden+sa*specen(i)+
     &          b*df(i)**2*max(pc(i),(a+b*specen(i)))
        enddo
c
c***  update the pressure and internal energy
      else
        do i=lft,llt
          xmu=1.0/df(i)-1.
          dfmu=df(i)*xmu
          facp=.5*(1.+sign(1.,xmu))
          facn=1.-facp
          xnum=1.+xmu*(+g0d2-sad2*xmu)
          xdem=1.-xmu*(s11+dfmu*(s2+s3*dfmu))
          tmp=facp/(xdem*xdem)
          a=roc2*xmu*(facn+tmp*xnum)
          b=g0+sa*xmu
          dvov0=0.5*dvol(i)/v0(i)
          denom=1.+b*dvov0
          pnew(i)=(a+specen(i)*b)/max(1.e-6,denom)
          pnew(i)=max(pnew(i),pc(i))
          specen(i)=specen(i)-pnew(i)*dvov0
        enddo
      endif
c
      return
      end
```

Implementation discussion

The Gruneisen EOS implemented in the example subroutines has the same form as *EOS_GRUNEISEN, EOS form 4. Its update of the pressure and the internal energy is typical for an EOS that is linear in the internal energy,

$$P = A(\rho) + B(\rho)E\ ,$$

where A and B correspond to the variables `a` and `b` in the example subroutines, and E is `specen`. Integrating the energy equation with the trapezoidal rule gives

$$E^{n+1} = E^n + \frac{1}{2}(\sigma'^n + \sigma'^{n+1})\Delta\varepsilon - \frac{1}{2}(P^n + q^n + P^{n+1} + q^{n+1})\frac{\Delta V}{V_0}$$

where the superscripts refer to the time step, ΔV is the change in the volume associated with the Gauss point and V_0 is the reference volume. Collecting all the energy

contributions on the right-hand side except for the contribution from the new pressure gives a simple linear relationship between the new internal energy and pressure,

$$E^{n+1} = \tilde{E} - \frac{P^{n+1}\Delta V}{2V_0}.$$

The value of `specen` passed to `ueos`*XX* for the pressure and energy update corresponds to $\tilde{E}$. Substituting this relation into the EOS and solving for the new pressure gives:

$$P^{n+1} = \frac{A\rho^{n+1} + B\rho^{n+1}\tilde{E}}{1 + \frac{B\Delta V}{2V_0}} .$$

The final update of the new energy is calculated using the updated pressure. For a more general EOS, the nonlinear equation for the new pressure,

$$P^{n+1} = P\left(\rho^{n+1}, \tilde{E} - \frac{P^{n+1}\Delta V}{2V_0}\right)$$

is solved iteratively using Newton iteration or successive substitution.

The pressure cut-off, `pc`, limits the amount of pressure that can be generated by tensile loading, `pnew=max(pnew,pc)`. Its value is usually specified in the *MAT input, such as *MAT_JOHNSON_COOK. It is not enforced outside the EOS subroutines. Thus, you determine whether to enforce the pressure cut-off in `ueos`*XX*. To impose the pressure cut-off, apply it before the final update to the internal energy; otherwise, the energy will be incorrect.

Calculating the bulk modulus is similar to updating the pressure and energy. Since the bulk modulus calculation always precedes the pressure update, the values may be saved in a common block during the bulk modulus calculation to reduce the cost of the pressure update. The arrays that store the values in the vectorized subroutines should be dimensioned by `nlq`.

A common pitfall is using the wrong internal energy when implementing an EOS from a paper or book. Three internal energies are commonly used: the energy per unit mass, e_M, the energy per unit current volume, e_V, and the energy per unit reference volume, E. LS-DYNA always uses the energy per unit reference volume. Some helpful relations for converting between the EOS in the literature and the variables in LS-DYNA are

$$e_V = E\frac{V_0}{V} = \frac{\text{specen}}{\text{df}}$$
$$e_M = E\frac{V_0}{M} = \frac{\text{specen}}{\text{rho0}}$$
$$\rho = \rho_0\frac{V_0}{V} = \frac{\text{rho0}}{\text{df}}$$