APPENDIX C: User-Defined Element Interface for Solids and Shells

This appendix describes the user-defined element interface for solids and shells. The interface can accommodate either an integrated or a resultant element. For the integrated element, you supply two matrices defining the kinematical properties of the element. You also determine whether to use standard LS-DYNA hourglass stabilization, a user-defined stabilization, or no stabilization in the absence of zero-energy modes (see IHGF on *SECTION_SHELL/SOLID). The number and locations of the integration points are arbitrary, meaning you define them. For the resultant/discrete element formulations, you implement the force and stiffness assembly. For both element formulations, history variables can be associated with the elements. In both cases, if desired, the element may have more than the conventional 3 (for bricks) and 6 (for shells) degrees-of-freedom per node (see Extra Degrees-Of-Freedom).

Algorithm Outline

We implemented user-defined, integrated elements in the same way as standard LS-DYNA elements aside from additionally calling user routines for setting up the matrices of interest. In the end, LS-DYNA constructs the gradient-displacement matrix, B_{ijkK}, with the property that

$$B_{ijkK}u_{kK} = \frac{\partial v_i}{\partial x_j} \ ,$$

where

$$u_{kK} = \text{ Vector of velocity nodal degrees-of-freedom}$$

$$\frac{\partial v_i}{\partial x_i} = \text{ Velocity gradient}$$

Moreover, the numerical integration requires the input of J, the determinant of the Jacobian matrix that determines the mapping from the isoparametric to the physical domain. Your subroutine determines the gradient-displacement matrix and either the Jacobian matrix or the determinant of the Jacobian matrix, depending on the definition of ITAJ. From these expressions, the symmetric part of the velocity gradient determines the strain, and the corresponding antisymmetric part determines the spin. LS-DYNA, then, evaluates the stresses with a specified constitutive model. From the stresses, it obtains the internal forces with

$$f_{kK} = \int \sigma_{ij} B_{ijkK} dV \ ,$$

where σ_{ij} are the stresses. LS-DYNA calculates the geometric and material tangent stiffnesses through

$$K_{iIjJ}^{\text{mat}} = \int C_{klmn} B_{kliI} B_{mnjJ} dV$$

and

$$K_{iIjJ}^{\text{geo}} = \int \sigma_{mn} B_{kmiI} B_{knjJ} dV$$

where C_{klmn} is the tangent modulus for the material. LS-DYNA evaluates the integrals using user-defined quadrature and J.

For user-defined hourglass control, you must provide the corresponding internal force and stiffness contribution in a separate user routine.

With resultant elements, you provide the force and stiffness matrix directly for the entire element.

Invoking User-Defined Elements

To invoke a user-defined element, you must do the following:

1. Write a user element subroutine that defines the kinematics or kinetics of the element.

2. Create a custom executable that includes these subroutines.

3. Invoke the element by specifying this subroutine on the corresponding *SECTION card with ELFORM (101 through 105).

We provide you with the dummy subroutines for the user-defined elements in a FORTRAN source file for you to modify, along with the necessary object files to compile a new executable. Contact Ansys or your local distributor for information about obtaining these files and what compiler/version to use for your specific platform. Up to five user elements can simultaneously be used for bricks and shells (i.e., ten). This text is an introductory guide to implementing such an element.

Integrated Elements
To activate a user-defined element, set ELFORM to a number between 101 and 105 and NIPP/NIP to a value greater than 0 on Card 5/3 of the *SECTION_SHELL/SOLID definition. By doing so, LS-DYNA calls the following user subroutine to determine the kinematics of the elements:

```
subroutine uXXX_bYYY(bmtrx,gmtrx,gjac,...
⋮
dimension bmtrx(nlq,3,3,*),gmtrx(nlq,3,3),gjac(*)
```

where `XXX` is substituted for `shl` for a shell section and `sld` for a solid section, and `YYY` is the value of ELFORM in *SECTION_SHELL/SOLID.

Your choice of ITAJ on *SECTION_SHELL/SOLID determines how to set up these matrices in your subroutine. If ITAJ = 0, set the isoparametric gradient-displacement matrix and the Jacobian matrix. The user subroutine represents these matrices with the arrays `bmtrx` and `gmtrx`, respectively. Here, the first index corresponds to the LS-DYNA block loop index, where `nlq` is the block size. For a more convenient notation in this discussion, we assign a correspondence between the arrays `gmtrx` and `bmtrx` in the subroutines to matrices/tensors as:

`gmtrx(*,i,j)` - g_{ij}
`bmtrx(*,i,j,k)` - b_{ijk}

Your subroutine should determine these matrices should so that at the current integration point:

$$g_{ij} = \frac{\partial x_i}{\partial \xi_j}$$

$$b_{ijk} u_k = \frac{\partial v_i}{\partial \xi_j} \Delta t$$

In the above, we assume summation over repeated indices. We use the following notation:

$x_i(\xi_1, \xi_2, \xi_3, t)$ = i^{th} component of the current position vector at the isoparametric coordinate (ξ_1, ξ_2, ξ_3) and time t.

$v_i(\xi_1, \xi_2, \xi_3, t)$ = i^{th} component of the velocity vector at the isoparametric coordinate (ξ_1, ξ_2, ξ_3) and time t.

Δt = current time step

u_k = k^{th} component of generalized local displacements

ξ_i = i^{th} component of the isoparametric coordinate ranging from -1.0 to 1.0

For shells, ILOC on *SECTION_SHELL specifies the coordinate system for the variables. ILOC = 0 sets the coordinate system to the LS-DYNA local coordinate system (`ILOC=0`), while ILOC = 1 sets it to the global coordinate system. LS-DYNA passes the coordinate system transformation matrix to the user routines, where the columns represent the local unit base vectors. The resulting strains must always be in the local coordinate system for the constitutive evaluations.

For no extra degrees-of-freedom (see Extra Degrees-Of-Freedom), the following formula gives the index k in the displacement expression:

$$k = n(m-1) + d \ ,$$

where $n = 3$ if only translational degrees-of-freedom are present (typical for solids) and $n = 6$ if rotational degrees-of-freedom are present (typical for shells), m is the local node

number ($m = 1,2,\ldots$), and d is the degree of freedom. The translational degrees-of-freedom correspond to $d \leq 3$ and the rotational degrees-of-freedom to $4 \leq d \leq 6$.

If ITAJ = 1, your subroutine sets up the physical gradient-displacement matrix and the Jacobian determinant. The user subroutines represent these by the arrays `bmtrx` and `gjac`, respectively. Again, we assign a correspondence between the arrays `gjac` and `bmtrx` in the subroutines to matrices/tensors as follows

`gjac(*)`	-	J
`bmtrx(*,i,j,k)`	-	b_{ijk}

Your subroutine should determine these matrices should so that at the current integration point:

$$J = \det \frac{\partial x_i}{\partial \xi_j}$$

$$b_{ijk} u_k = \frac{\partial v_i}{\partial x_j} \Delta t$$

To set up these matrices, LS-DYNA passes a set of additional auxiliary variables to the user-defined element subroutines. These variables include the isoparametric coordinates, the element thickness, the shape function values, and the derivatives of the shape functions. Again, for shells, LS-DYNA expresses these variables in either the local or global coordinate system depending on your choice of ILOC. For more information on these variables, see the comments in the subroutines.

The integrated elements can use up to 100 integration points (in the plane for shells) at arbitrary locations. You must specify these integration points in terms of isoparametric coordinates and weights (see Card 5.1 for *SECTION_SHELL and Card 4 for *SECTION_SHELL). The isoparametric coordinates should range from –1 to 1, and the weights should sum up to 4 for shells and 8 for solids.

You may need to incorporate hourglass stabilization to suppress zero energy modes. To do this, set IHGF > 0 in *SECTION_SHELL/SOLID. IHGF = 1 sets automatically using the LS-DYNA hourglass routines. For IHGF = 2 or 3, you must provide hourglass force and stiffness in a specific user-defined routine. If IHGF = 3, physical stabilization becomes available since LS-DYNA passes the resultant material tangent moduli to the hourglass routine to provide the material's current membrane, bending, and coupled membrane-bending stiffness. With C_{ij} denoting the material tangent modulus in matrix form, we express the resultant tangent moduli as

$$\bar{C}^0_{ij} = \int C_{ij}\, dV \qquad \text{(membrane)}$$

$$\bar{C}^1_{ij} = \int z^1 C_{ij}\, dV \qquad \text{(membrane} - \text{bending)}$$

$$\bar{C}^2_{ij} = \int z^2 C_{ij}\, dV \qquad \text{(bending)}$$

where z is the thickness coordinate for shells. For solids, LS-DYNA only passes the first resultant modulus. In this case, the array has 21 entries corresponding to the subdiagonal terms of the 6 by 6 resultant matrix. For the matrix, index (i,j) in the material tangent modulus matrix, where $i \geq j$, the following gives the index I of the array passed to the routine:

$$I = i(i-1)/2 + j$$

In other words, the subdiagonal terms are stored row-wise in the array. For shells, LS-DYNA passes all three moduli in the local coordinate system, where each array has 15 entries corresponding to the subdiagonal terms of the 5 by 5 resultant matrices. We eliminated the through-thickness direction by assuming plane stress. The formula for the array indices transformation above holds.

The subroutine for user-defined hourglass control is called

```
subroutine uXXX_eYYY(force,stiff,ndtot,...
⋮
dimension force(nlq,*),stiff(nlq,ndtot,*)
```

where again `XXX` is substituted for `shl` for a shell section and `sld` for a soli section, and `YYY` is the value of ELFORM in *SECTION_SHELL/SOLID. The variables in the subroutine correspond to the force and stiffness as

`force(*,i)` - f_i

`stiff(*,i,j)` - K_{ij}

The indices correspond to node and degree-of-freedom numbers in the same way as for the displacements. For shells, ILOC on *SECTION_SHELL specifies the coordinate system for the force and stiffness. The variable `ndtot` gives the total number of degrees-of-freedom for the element ($4 \times (6 + n_{\text{xdof}})$ for shells and $8 \times (3 + n_{\text{xdof}})$ for solids). This subroutine has the property parameters and history variables associated with the element as input. You specify the property parameters on *SECTION_SHELL/SOLID. Each user-defined element can have no more than 40 property parameters and 100 history variables. You must update the history variables in this routine.

Resultant/Discrete Elements

By setting NIPP/NIP equal to 0 in *SECTION_SHELL/SOLID, LS-DYNA assumes a resultant/discrete element formulation. For this option, you provide force and stiffness in the same user-defined routine as for the user-defined hourglass control, that is, subroutine `uXXX_eYYY` where `XXX` is substituted for `shl` for a shell-section and `sld` for a solid-section, and `YYY` is the value of ELFORM in *SECTION_SHELL/SOLID. Thus, IHGF > 0 on *SECTION_SHELL/SOLID (hourglass stabilization) is incompatible with this option. Because you provide the force and stiffness, LS-DYNA does not call a material subroutine to update stresses and history variables. Instead, this user-defined element subroutine updates the stresses and history variables.

Nevertheless, you need to define *MAT_ELASTIC as the material for the corresponding part with suitable values of Young's modulus and Poisson's ratio. LS-DYNA calculates

the time step and determines the contact stiffness with these material properties. Again, LS-DYNA passes the property parameters and history variables to the routine. It also passes the thickness of the elements for shells. For the shell thickness update option (ISTUPD > 0 on *CONTROL_SHELL), you need to update the thicknesses in this routine.

The stiffness matrix assembled in the element routines can be input as nonsymmetric if and only if LCPACK = 3 on *CONTROL_IMPLICIT_SOLVER, that is if LS-DYNA uses the nonsymmetric solver to update the Newton iterates.

Additional Features of User-Defined Elements

Here we will provide a short description of the additional features associated with user-defined elements.

Nodal Fiber Vectors
If a user-defined shell element formulation requires the nodal fiber vectors, set IUNF to 1 on the *SECTION_SHELL card. With this option, the element routines process the nodal fiber vectors, which can be used as input to the user-defined routines for determining the b_{ijk}, g_{ij}/J, f_i and K_{ij} tensors/matrices. If IUNF ≠ 1, LS-DYNA assumes that the fiber direction is always normal to the plane of the shell. LS-DYNA expresses these nodal fiber vectors in either the local or global system, depending on your choice of ILOC on *SECTION_SHELL. See comments in the user subroutines for more information.

Extra Degrees-Of-Freedom
Exotic element formulations may require extra degrees-of-freedom per node beside the translational (and rotational) degrees-of-freedom. You can specify up to 15 extra degrees-of-freedom per node for user-defined elements. Note that if your element requires more than three extra degrees-of-freedom per node, you must increase NXDOFUE in the file nhisparm.inc to the corresponding value. This parameter is, by default, set to 3. We chose this value to limit the default sizes of common blocks and heaps. Note that NXDOFUE does not set the number of extra degrees-of-freedom per node for your element. It just sets the limit for allocation. To specify the number, you set NXDOF on *SECTION_SHELL/SOLID. We recommend setting NXDOFUE equal to NXDOF if possible.

We implemented the extra degrees-of-freedom per node so that LS-DYNA creates *a scalar node for every three extra degrees-of-freedom* associated with an actual node. Scalar nodes are dummy nodes linked to a node on an element. In other words, LS-DYNA automatically creates ceil(NXDOF/3) scalar nodes. With *NODE_SCALAR_VALUE and *ELEMENT_SOLID/SHELL_DOF, you can associate some of the extra degrees-of-freedom with a scalar node ID of your choice. This association is optional as LS-DYNA will create the scalar nodes regardless of using these keywords. It helps with setting boundary conditions and can be used to specify initial conditions, which will be discussed in more depth below. With *NODE_SCALAR_VALUE, you select the node ID associated with up to three degrees-of-freedom and then associate this scalar node with the real node

using *ELEMENT_SOLID/SHELL_DOF. However, using this method, only one related is associated with each real node. Thus, if you have more than three extra degrees-of-freedom, you cannot associate a scalar node for all the degrees-of-freedom.

We have implemented two methods for setting initial conditions. With the first method, you set the initial conditions in the first step of the user subroutine, while with the second method, you use *NODE_SCALAR_VALUE and *ELEMENT_SOLID/SHELL_DOF. We generally recommend the first method. The second method, however, is helpful if you want a node number associated with the extra degrees-of-freedom and want to apply boundary conditions to a degree-of-freedom. With the second method, you can set the initial values of up to three degrees-of-freedom per scalar node. Note that this method can only initialize up to three extra degrees-of-freedom per node because you can only associate one scalar node to each actual node. As an example:

```
*NODE_SCALAR_VALUE
$     NID               V1              V2              V3      NDF
       11              1.0                                       1
       12              1.0                                       1
       13              1.0                                       1
       14              1.0                                       1
*ELEMENT_SHELL_DOF
$     EID      PID       N1      N2      N3      N4
        1        1        1       2       3       4
$                      NS1     NS2     NS3     NS4
                        11      12      13      14
```

defines an element with one extra degree-of-freedom per node. The initial value of the corresponding degree-of-freedom is 1.0, and it is unconstrained. Recall that NXDOF on *SECTION_SHELL specifies the actual number of extra degrees-of-freedom. Thus, if you set NXDOF to 4 and use *NODE_SCALAR_VALUE as in the previous example, you would still need to initialize three more degrees-of-freedom in the user subroutine.

The array `xdof` contains the current values of these extra variables. LS-DYNA passes `xdof` to the user routines for setting up the correct kinematical properties. See comments in the routines for more information. The formula for the displacement index changes to

$$k = (n + n_{\text{xdof}})(m - 1) + d$$

where n_{xdof} is the number of extra degrees-of-freedom, n is the number of normal degrees-of-freedom, and m is an index for the node. The extra degrees-of-freedom for each node corresponds to $n + 1 \leq d \leq n + n_{\text{xdof}}$.

For dynamic simulations, the mass corresponding to these extra degrees-of-freedom may be defined by either:

- *ELEMENT_INERTIA or *ELEMENT_MASS, or
- the subroutine `uXXX_mYYY` for `XXX` equal to `sld` or `shl` and `YYY` between 101 and 105.

For solid elements with the subroutine method for specifying mass, XNOD on *SECTION_SOLID indicates the interpretation of the input mass. By default (XNOD = 0), the

mass is interpreted as per extra degree-of-freedom. By setting XNOD to 1, the mass is interpreted as nodal, which means you can specify the mass for the scalar node. The mass for the scalar node is then divided equally among the degrees-of-freedom associated with that scalar node.

Section Keywords

The following is a list of keywords that apply to the user-defined elements.

The *SECTION_SHELL Card
Add Card 5 with the accompanying optional cards, Cards 5.1 and 5.2, of *SECTION_-SHELL to the keyword input deck if you invoke the user-defined shell element option.

Additional Card for ELFORM = 101,102,103,104 or 105

Card 5	1	2	3	4	5	6	7	8
Variable	NIPP	NXDOF	IUNF	IHGF	ITAJ	LMC	NHSV	ILOC
Type	I	I	I	I	I	I	I	I
Default	0	0	0	0	0	0	0	0

Include NIPP cards according to the following format.

Card 5.1	1	2	3	4	5	6	7	8
Variable	XI	ETA	WGT					
Type	F	F	F					

Define LMC property parameters using 8 parameters per card.

Card 5.2	1	2	3	4	5	6	7	8
Variable	P1	P2	P3	P4	P5	P6	P7	P8
Type	F	F	F	F	F	F	F	F

VARIABLE	DESCRIPTION
ELFORM	GT.100.AND.LT.106: User-defined shell
NIPP	Number of in-plane integration points for user-defined shell (0 if resultant element). Up to 100 allowed.
NXDOF	Number of extra degrees-of-freedom per node for user-defined shell
IUNF	Flag for using nodal fiber vectors in user-defined shell EQ.0: Nodal fiber vectors are not used. EQ.1: Nodal fiber vectors are used
IHGF	Flag for using hourglass stabilization (NIPP is greater than 0): EQ.0: Hourglass stabilization is not used. EQ.1: LS-DYNA hourglass stabilization is used. EQ.2: User-defined hourglass stabilization is used. EQ.3: Same as 2, but the resultant material tangent moduli are passed.
ITAJ	Flag for setting up finite element matrices (NIPP is greater than 0) EQ.0: Set up matrices with respect to the isoparametric domain EQ.1: Set up matrices with respect to the physical domain
LMC	Number of property parameters
NHSV	Number of history variables
ILOC	Coordinate system option: EQ.0: Pass all variables in the LS-DYNA local coordinate system EQ.1: Pass all variables in the global coordinate system
XI	First isoparametric coordinate
ETA	Second isoparametric coordinate
WGT	Isoparametric weight
P*i*	i^{th} property parameter

For more information on the variables, consult the previous sections in this appendix.

The *SECTION_SOLID Card

Add Card 3 with the accompanying optional cards, Cards 4 and 5, of *SECTION_SOLID to the input deck if you invoke the user-defined solid element option.

Additional card for ELFORM = 101,102,103,104 or 105

Card 3	1	2	3	4	5	6	7	8
Variable	NIP	NXDOF	IHGF	ITAJ	LMC	NHSV	XNOD	
Type	I	I	I	I	I	I	I	
Default	0	0	0	0	0	0	0	

Include NIP cards according to the following format.

Card 4	1	2	3	4	5	6	7	8
Variable	XI	ETA	ZETA	WGT				
Type	F	F	F	F				

Define LMC property parameters using 8 parameters per card.

Card 5	1	2	3	4	5	6	7	8
Variable	P1	P2	P3	P4	P5	P6	P7	P8
Type	F	F	F	F	F	F	F	F

VARIABLE	DESCRIPTION
ELFORM	GT.100.AND.LT.106: User-defined solid
NIP	Number of integration points for user-defined solid (0 if resultant element)
NXDOF	Number of extra degrees-of-freedom per node for user-defined solid

VARIABLE	DESCRIPTION
IHGF	Flag for using hourglass stabilization (NIP > 0): EQ.0: Hourglass stabilization is not used. EQ.1: LS-DYNA hourglass stabilization is used. EQ.2: User-defined hourglass stabilization is used. EQ.3: Same as 2, but the resultant material tangent moduli are passed.
ITAJ	Flag for setting up finite element matrices (NIP > 0): EQ.0: Set up matrices with respect to the isoparametric domain. EQ.1: Set up matrices with respect to the physical domain.
LMC	Number of property parameters
NHSV	Number of history variables
XNOD	Controls how the mass of extra degrees-of-freedom is interpreted when defined using user subroutine. EQ.0: The mass is defined per degree of freedom. EQ.1: The mass is defined as a nodal mass, where each scalar node has three extra degrees-of-freedom. See Extra Degrees-Of-Freedom.
XI	First isoparametric coordinate
ETA	Second isoparametric coordinate
ZETA	Third isoparametric coordinate
WGT	Isoparametric weight
P*i*	i^{th} property parameter

For more information on the variables, consult the previous sections in this appendix.

Sample User Shell Element 101 (Belytschko-Tsay shell)

We can write the geometry of the Belytschko-Tsay element in local coordinates as:

$$x_i = (x_{iI} + \frac{t}{2}\xi_3\delta_{i3})N_I(\xi_1,\xi_2)$$
$$v_i = (v_{iI} + \frac{t}{2}\xi_3 e_{ij3}\omega_{jI})N_I(\xi_1,\xi_2)$$

where

$x_{iI} =$ i^{th} component of the coordinate of node I

$v_{iI} =$ i^{th} component of the translational velocity of node I

$\omega_{jI} =$ j^{th} component of the rotational velocity of node I

$t =$ thickness of the element

$e_{ijk} =$ permutation tensor

$N_I =$ shape function localized at node I

$\delta_{i3} =$ Kronecker delta

Taking the derivative of these expressions with respect to the isoparametric coordinate yields:

$$\frac{\partial x_i}{\partial \xi_1} = (x_{iI} + \frac{t}{2}\xi_3\delta_{i3})\frac{\partial N_I}{\partial \xi_1}$$
$$\frac{\partial x_i}{\partial \xi_2} = (x_{iI} + \frac{t}{2}\xi_3\delta_{i3})\frac{\partial N_I}{\partial \xi_2}$$
$$\frac{\partial x_i}{\partial \xi_3} = \frac{t}{2}\delta_{i3}$$

and

$$\frac{\partial v_i}{\partial \xi_1} = (v_{iI} + \frac{t}{2}\xi_3 e_{ij3}\omega_{jI})\frac{\partial N_I}{\partial \xi_1}$$
$$\frac{\partial v_i}{\partial \xi_2} = (v_{iI} + \frac{t}{2}\xi_3 e_{ij3}\omega_{jI})\frac{\partial N_I}{\partial \xi_2}$$
$$\frac{\partial v_i}{\partial \xi_3} = \frac{t}{2}e_{ij3}\omega_{jI}N_I$$

respectively. Using these expressions, we can implement the element as a user-defined shell as follows.

```
      subroutine ushl_b101(bmtrx,gmtrx,gjac,
     1     xi,eta,zeta,
     2     n1,n2,n3,n4,
     3     dn1dxi,dn2dxi,dn3dxi,dn4dxi,
     4     dn1deta,dn2deta,dn3deta,dn4deta,
     5     x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,
     6     xdof,
     7     thick,thck1,thck2,thck3,thck4,
     8     fx1,fx2,fx3,fx4,
     9     fy1,fy2,fy3,fy4,
     .     fz1,fz2,fz3,fz4,
     .     gl11,gl21,gl31,gl12,gl22,gl32,gl13,gl23,gl33,
     .     lft,llt)
```

```
      include 'nlqparm'
c
c     Compute b and g matrix for user-defined shell 101
c
      dimension bmtrx(nlq,3,3,*),gmtrx(nlq,3,3),gjac(nlq)
      REAL n1,n2,n3,n4
      dimension x1(nlq),x2(nlq),x3(nlq),x4(nlq)
      dimension y1(nlq),y2(nlq),y3(nlq),y4(nlq)
      dimension z1(nlq),z2(nlq),z3(nlq),z4(nlq)
      dimension thick(nlq)
      dimension thck1(nlq),thck2(nlq),thck3(nlq),thck4(nlq)
      dimension xdof(nlq,8,3)
      dimension fx1(nlq),fx2(nlq),fx3(nlq),fx4(nlq)
      dimension fy1(nlq),fy2(nlq),fy3(nlq),fy4(nlq)
      dimension fz1(nlq),fz2(nlq),fz3(nlq),fz4(nlq)
      dimension gl11(nlq),gl21(nlq),gl31(nlq),
      gl12(nlq),gl22(nlq),gl32(nlq),
      gl13(nlq),gl23(nlq),gl33(nlq)
c
      do i=lft,llt
c
         gmtrx(i,1,1)=
     1        x1(i)*dn1dxi+x2(i)*dn2dxi+
     2        x3(i)*dn3dxi+x4(i)*dn4dxi
         gmtrx(i,2,1)=
     1        y1(i)*dn1dxi+y2(i)*dn2dxi+
     2        y3(i)*dn3dxi+y4(i)*dn4dxi
         gmtrx(i,3,1)=
     1        0.
         gmtrx(i,1,2)=
     1        x1(i)*dn1deta+x2(i)*dn2deta+
     2        x3(i)*dn3deta+x4(i)*dn4deta
         gmtrx(i,2,2)=
     1        y1(i)*dn1deta+y2(i)*dn2deta+
     2        y3(i)*dn3deta+y4(i)*dn4deta
         gmtrx(i,3,2)=
     1        0.
         gmtrx(i,1,3)=
     1        0.
         gmtrx(i,2,3)=
     1        0.
         gmtrx(i,3,3)=
     1        .5*thick(i)
c
         coef=.5*thick(i)*zeta
c
         bmtrx(i,1,1,1) =dn1dxi
         bmtrx(i,1,1,7) =dn2dxi
         bmtrx(i,1,1,13)=dn3dxi
         bmtrx(i,1,1,19)=dn4dxi
c
         bmtrx(i,1,1,5) =coef*dn1dxi
         bmtrx(i,1,1,11)=coef*dn2dxi
         bmtrx(i,1,1,17)=coef*dn3dxi
         bmtrx(i,1,1,23)=coef*dn4dxi
c
         bmtrx(i,1,2,1) =dn1deta
         bmtrx(i,1,2,7) =dn2deta
         bmtrx(i,1,2,13)=dn3deta
         bmtrx(i,1,2,19)=dn4deta
c
         bmtrx(i,1,2,5) =coef*dn1deta
         bmtrx(i,1,2,11)=coef*dn2deta
         bmtrx(i,1,2,17)=coef*dn3deta
         bmtrx(i,1,2,23)=coef*dn4deta
c
```

```
         bmtrx(i,2,1,2) =dn1dxi
         bmtrx(i,2,1,8) =dn2dxi
         bmtrx(i,2,1,14)=dn3dxi
         bmtrx(i,2,1,20)=dn4dxi
c
         bmtrx(i,2,1,4) =-coef*dn1dxi
         bmtrx(i,2,1,10)=-coef*dn2dxi
         bmtrx(i,2,1,16)=-coef*dn3dxi
         bmtrx(i,2,1,22)=-coef*dn4dxi
c
         bmtrx(i,1,3,5) =.5*thick(i)*n1
         bmtrx(i,1,3,11)=.5*thick(i)*n2
         bmtrx(i,1,3,17)=.5*thick(i)*n3
         bmtrx(i,1,3,23)=.5*thick(i)*n4
c
         bmtrx(i,3,1,3) =dn1dxi
         bmtrx(i,3,1,9) =dn2dxi
         bmtrx(i,3,1,15)=dn3dxi
         bmtrx(i,3,1,21)=dn4dxi
c
         bmtrx(i,2,2,2) =dn1deta
         bmtrx(i,2,2,8) =dn2deta
         bmtrx(i,2,2,14)=dn3deta
         bmtrx(i,2,2,20)=dn4deta
c
         bmtrx(i,2,2,4) =-coef*dn1deta
         bmtrx(i,2,2,10)=-coef*dn2deta
         bmtrx(i,2,2,16)=-coef*dn3deta
         bmtrx(i,2,2,22)=-coef*dn4deta
c
         bmtrx(i,2,3,4) =-.5*thick(i)*n1
         bmtrx(i,2,3,10)=-.5*thick(i)*n2
         bmtrx(i,2,3,16)=-.5*thick(i)*n3
         bmtrx(i,2,3,22)=-.5*thick(i)*n4
c
         bmtrx(i,3,2,3) =dn1deta
         bmtrx(i,3,2,9) =dn2deta
         bmtrx(i,3,2,15)=dn3deta
         bmtrx(i,3,2,21)=dn4deta
c
      enddo
c
      return
      end
```

To use the element for a part, we add the following section input to the keyword deck:

```
*SECTION_SHELL
$    SECID    ELFORM
         1       101
$       T1         T2         T3         T4

$     NIPP     NXDOF      IUNF      IHGF
         1         0         0         1
$       XI       ETA       WGT
        0.        0.        4.
```

Sample User Solid Element 101 (constant stress solid)

We can express the geometry for the constant stress solid as:

$$x_i = x_{iI} N_I(\xi_1, \xi_2, \xi_3)$$
$$v_i = v_{iI} N_I(\xi_1, \xi_2, \xi_3)$$

where

$x_{iI} =$ i^{th} component of the coordinate of node I

$v_{iI} =$ i^{th} component of the translational velocity of node I

$N_I =$ shape function localized at node I

Taking the derivative of these expressions with respect to the isoparametric coordinate yields:

$$\frac{\partial x_i}{\partial \xi_1} = x_{iI} \frac{\partial N_I}{\partial \xi_1}$$
$$\frac{\partial x_i}{\partial \xi_2} = x_{iI} \frac{\partial N_I}{\partial \xi_2}$$
$$\frac{\partial x_i}{\partial \xi_3} = x_{iI} \frac{\partial N_I}{\partial \xi_3}$$

and

$$\frac{\partial v_i}{\partial \xi_1} = v_{iI} \frac{\partial N_I}{\partial \xi_1}$$
$$\frac{\partial v_i}{\partial \xi_2} = v_{iI} \frac{\partial N_I}{\partial \xi_2}$$
$$\frac{\partial v_i}{\partial \xi_3} = v_{iI} \frac{\partial N_I}{\partial \xi_3}$$

respectively. Using these expressions, we can implement the element as a user-defined solid as follows.

```
      subroutine usld_b101(bmtrx,gmtrx,gjac,
     1     xi,eta,zeta,
     2     n1,n2,n3,n4,n5,n6,n7,n8,
     3     dn1dxi,dn2dxi,dn3dxi,dn4dxi,
     4     dn5dxi,dn6dxi,dn7dxi,dn8dxi,
     5     dn1deta,dn2deta,dn3deta,dn4deta,
     6     dn5deta,dn6deta,dn7deta,dn8deta,
     7     dn1dzeta,dn2dzeta,dn3dzeta,dn4dzeta,
     8     dn5dzeta,dn6dzeta,dn7dzeta,dn8dzeta,
     9     x1,x2,x3,x4,x5,x6,x7,x8,
     .     y1,y2,y3,y4,y5,y6,y7,y8,
     .     z1,z2,z3,z4,z5,z6,z7,z8,
     .     xdof,
     .     lft,llt)
      include 'nlqparm'
c
c     Compute b and g matrix for user-defined solid 101
c
      dimension bmtrx(nlq,3,3,*),gmtrx(nlq,3,3),gjac(nlq)
      REAL n1,n2,n3,n4,n5,n6,n7,n8
      dimension x1(nlq),x2(nlq),x3(nlq),x4(nlq)
      dimension x5(nlq),x6(nlq),x7(nlq),x8(nlq)
      dimension y1(nlq),y2(nlq),y3(nlq),y4(nlq)
      dimension y5(nlq),y6(nlq),y7(nlq),y8(nlq)
      dimension z1(nlq),z2(nlq),z3(nlq),z4(nlq)
      dimension z5(nlq),z6(nlq),z7(nlq),z8(nlq)
```

APPENDIX C

```
      dimension xdof(nlq,8,3)
c
      do i=lft,llt
c
         gmtrx(i,1,1)=x1(i)*dn1dxi+x2(i)*dn2dxi+
     1        x3(i)*dn3dxi+x4(i)*dn4dxi+
     2        x5(i)*dn5dxi+x6(i)*dn6dxi+
     3        x7(i)*dn7dxi+x8(i)*dn8dxi
         gmtrx(i,2,1)=y1(i)*dn1dxi+y2(i)*dn2dxi+
     1        y3(i)*dn3dxi+y4(i)*dn4dxi+
     2        y5(i)*dn5dxi+y6(i)*dn6dxi+
     3        y7(i)*dn7dxi+y8(i)*dn8dxi
         gmtrx(i,3,1)=z1(i)*dn1dxi+z2(i)*dn2dxi+
     1        z3(i)*dn3dxi+z4(i)*dn4dxi+
     2        z5(i)*dn5dxi+z6(i)*dn6dxi+
     3        z7(i)*dn7dxi+z8(i)*dn8dxi
         gmtrx(i,1,2)=x1(i)*dn1deta+x2(i)*dn2deta+
     1        x3(i)*dn3deta+x4(i)*dn4deta+
     2        x5(i)*dn5deta+x6(i)*dn6deta+
     3        x7(i)*dn7deta+x8(i)*dn8deta
         gmtrx(i,2,2)=y1(i)*dn1deta+y2(i)*dn2deta+
     1        y3(i)*dn3deta+y4(i)*dn4deta+
     2        y5(i)*dn5deta+y6(i)*dn6deta+
     3        y7(i)*dn7deta+y8(i)*dn8deta
         gmtrx(i,3,2)=z1(i)*dn1deta+z2(i)*dn2deta+
     1        z3(i)*dn3deta+z4(i)*dn4deta+
     2        z5(i)*dn5deta+z6(i)*dn6deta+
     3        z7(i)*dn7deta+z8(i)*dn8deta
         gmtrx(i,1,3)=x1(i)*dn1dzeta+x2(i)*dn2dzeta+
     1        x3(i)*dn3dzeta+x4(i)*dn4dzeta+
     2        x5(i)*dn5dzeta+x6(i)*dn6dzeta+
     3        x7(i)*dn7dzeta+x8(i)*dn8dzeta
         gmtrx(i,2,3)=y1(i)*dn1dzeta+y2(i)*dn2dzeta+
     1        y3(i)*dn3dzeta+y4(i)*dn4dzeta+
     2        y5(i)*dn5dzeta+y6(i)*dn6dzeta+
     3        y7(i)*dn7dzeta+y8(i)*dn8dzeta
         gmtrx(i,3,3)=z1(i)*dn1dzeta+z2(i)*dn2dzeta+
     1        z3(i)*dn3dzeta+z4(i)*dn4dzeta+
     2        z5(i)*dn5dzeta+z6(i)*dn6dzeta+
     3        z7(i)*dn7dzeta+z8(i)*dn8dzeta
c
         bmtrx(i,1,1,1) =dn1dxi
         bmtrx(i,1,1,4) =dn2dxi
         bmtrx(i,1,1,7) =dn3dxi
         bmtrx(i,1,1,10)=dn4dxi
         bmtrx(i,1,1,13)=dn5dxi
         bmtrx(i,1,1,16)=dn6dxi
         bmtrx(i,1,1,19)=dn7dxi
         bmtrx(i,1,1,22)=dn8dxi
c
         bmtrx(i,2,1,2) =dn1dxi
         bmtrx(i,2,1,5) =dn2dxi
         bmtrx(i,2,1,8) =dn3dxi
         bmtrx(i,2,1,11)=dn4dxi
         bmtrx(i,2,1,14)=dn5dxi
         bmtrx(i,2,1,17)=dn6dxi
         bmtrx(i,2,1,20)=dn7dxi
         bmtrx(i,2,1,23)=dn8dxi
c
         bmtrx(i,3,1,3) =dn1dxi
         bmtrx(i,3,1,6) =dn2dxi
         bmtrx(i,3,1,9) =dn3dxi
         bmtrx(i,3,1,12)=dn4dxi
         bmtrx(i,3,1,15)=dn5dxi
         bmtrx(i,3,1,18)=dn6dxi
         bmtrx(i,3,1,21)=dn7dxi
```

```
          bmtrx(i,3,1,24)=dn8dxi
c
          bmtrx(i,1,2,1) =dn1deta
          bmtrx(i,1,2,4) =dn2deta
          bmtrx(i,1,2,7) =dn3deta
          bmtrx(i,1,2,10)=dn4deta
          bmtrx(i,1,2,13)=dn5deta
          bmtrx(i,1,2,16)=dn6deta
          bmtrx(i,1,2,19)=dn7deta
          bmtrx(i,1,2,22)=dn8deta
c
          bmtrx(i,2,2,2) =dn1deta
          bmtrx(i,2,2,5) =dn2deta
          bmtrx(i,2,2,8) =dn3deta
          bmtrx(i,2,2,11)=dn4deta
          bmtrx(i,2,2,14)=dn5deta
          bmtrx(i,2,2,17)=dn6deta
          bmtrx(i,2,2,20)=dn7deta
          bmtrx(i,2,2,23)=dn8deta
c
          bmtrx(i,3,2,3) =dn1deta
          bmtrx(i,3,2,6) =dn2deta
          bmtrx(i,3,2,9) =dn3deta
          bmtrx(i,3,2,12)=dn4deta
          bmtrx(i,3,2,15)=dn5deta
          bmtrx(i,3,2,18)=dn6deta
          bmtrx(i,3,2,21)=dn7deta
          bmtrx(i,3,2,24)=dn8deta
c
          bmtrx(i,1,3,1) =dn1dzeta
          bmtrx(i,1,3,4) =dn2dzeta
          bmtrx(i,1,3,7) =dn3dzeta
          bmtrx(i,1,3,10)=dn4dzeta
          bmtrx(i,1,3,13)=dn5dzeta
          bmtrx(i,1,3,16)=dn6dzeta
          bmtrx(i,1,3,19)=dn7dzeta
          bmtrx(i,1,3,22)=dn8dzeta
c
          bmtrx(i,2,3,2) =dn1dzeta
          bmtrx(i,2,3,5) =dn2dzeta
          bmtrx(i,2,3,8) =dn3dzeta
          bmtrx(i,2,3,11)=dn4dzeta
          bmtrx(i,2,3,14)=dn5dzeta
          bmtrx(i,2,3,17)=dn6dzeta
          bmtrx(i,2,3,20)=dn7dzeta
          bmtrx(i,2,3,23)=dn8dzeta
c
          bmtrx(i,3,3,3) =dn1dzeta
          bmtrx(i,3,3,6) =dn2dzeta
          bmtrx(i,3,3,9) =dn3dzeta
          bmtrx(i,3,3,12)=dn4dzeta
          bmtrx(i,3,3,15)=dn5dzeta
          bmtrx(i,3,3,18)=dn6dzeta
          bmtrx(i,3,3,21)=dn7dzeta
          bmtrx(i,3,3,24)=dn8dzeta
c
      enddo
c
      return
      end
```

To use the element for a part, we add the following section input to the keyword deck

```
*SECTION_SOLID
$    SECID    ELFORM
```

```
         1       101
$       NIP     NXDOF      IHGF
         1         0         1
$        XI       ETA      ZETA       WGT
         0.        0.        0.       8.0
```

Examples

We present three test examples. Our first example is a simple tension-compression test of a solid cylinder. Figure 53-1 shows the geometry. We used the sample implementations of user elements and compared the results and performance with standard LS-DYNA elements. As for the computational efficiency, we note that the performance is worse. We expected this decrease in performance because we sought a user-friendly interface that came at the expense of performance. The implicit performance compares well with the other elements in LS-DYNA.

The second example is a combined bending and stretching simulation with the geometry shown in Figure 53-2. We ran the problem with the user element implementations and compared the results and performance with standard LS-DYNA elements. We saw the same tendencies as for the solid elements.

The third example is an impact between a solid bar and a shell beam. Both parts are modeled with user-defined elements. The results were similar to the ones obtained by substituting the sections for standard LS-DYNA sections, but the simulation time was about 3-4 times longer.

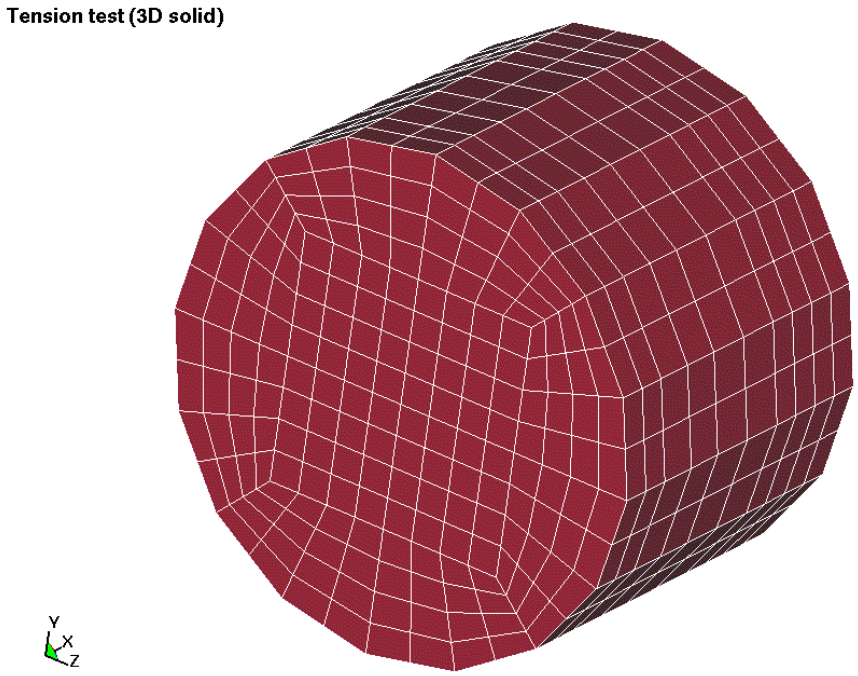

Figure 53-1. Solid mesh for user element test.

Figure 53-2. Shell mesh for the user element test.

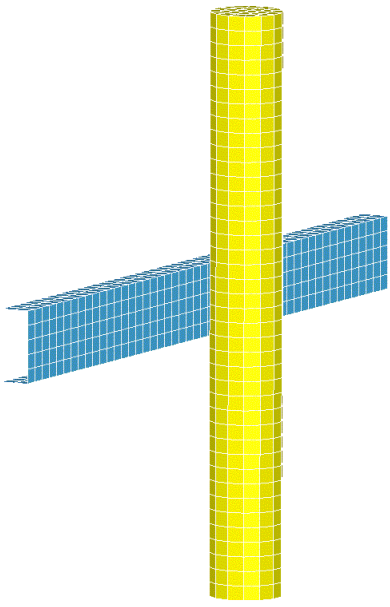

Figure 53-3. Impact between a user-defined shell and user-defined solid part.