On Setting up Boundary and Initial Conditions in S-ALE Models

Hao Chen, Ansys

LS-DYNA ALE has been widely used to simulating moving fluids interacting with structures. Unlike CFD, the focus is rather on the structure response under dynamic loading from fluids, than the fluids' motion. Fluids are agitated by a high pressure gradient; and then hit the structure, carrying a large momentum. The key in successfully capturing the physics lies in the fluid-structure interaction algorithm. It needs to accurately predict the peak of pressure loading during the impact, which is characterized as a momentum transfer process. This request could only be fulfilled by a transient analysis with a penalty-based coupling between fluids and structure.

In 2015, LSTC introduced a new structured ALE (S-ALE) solver option dedicated to solve the subset of ALE problems where a structured mesh is appropriate. As expected, recognizing the logical regularity of the mesh brought a reduced simulation time for the case of identical structured and unstructured mesh definitions. It also comes with a cleaner, conceptually simpler way of model setup. This article gives a brief description on setting up the boundary and initial conditions in S-ALE models.

What is Boundary?

In a Lagrangian model, mesh conforms to the material interface. The boundary is a collection of surface segments and nodes enclosing the material (*PART). And we could choose a set of segments or nodes to apply boundary conditions on them. For example, one uses *LOAD_SEGMENT to apply pressure on a set of segments; *BOUNDARY_SPC to apply nodal constraints on a set of nodes, etc.

However, in an ALE model, things are totally different. Mesh is no longer the spatial representation of the material. Rather, it is simply a domain in which we study the flow of that (and other) material. In Lagrangian, what really happened was that we applied boundary conditions on the material surface. The equivalent thing to do in an ALE model is to apply some force or constraints at the material interface, NOT at the mesh boundary. As the mesh boundary is just the limit of our working field, nothing more.

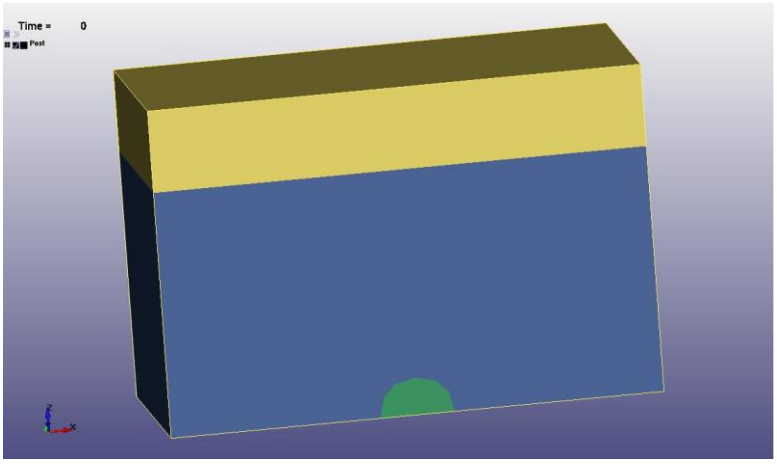
As the material interface is evolving in ALE models, plus it has no explicit description, generally it is not possible to apply "boundary conditions" directly at the material interface. Most of times, boundary conditions are applied through fluid-structure interaction (FSI). For example, we want to create a wave by pushing water sideways from left to right. If modeled as Lagrangian water, all we need to do is to find its boundary nodes and apply a prescribed motion on them. In ALE, it is not possible as these nodes do not move with water. Remember? Mesh does not move; fluids flow inside it. So what we do is to create a Lagrange plate and set up a FSI between water and the plate. Then we apply this prescribed motion onto the plate. Then FSI is going to enforce the water to move together with the plate.

Boundary Conditions

The above being said, on one condition we could still use boundary conditions in an ALE model. As you might have guessed, that is when the mesh boundary conforms to the material interface. Say we have a half symmetry model and -x mesh face is the symmetry plane. What we need to do is to apply nodal constraints along x direction on all nodes at that face. Another example is a flat bottom container. If we choose to align the bottom mesh face with that simple geometry container, we could apply a SPC along z direction at -z face nodes.

Another exception is transmitting boundary condition, i.e. *BOUNDARY_NON_REFLECTING. The transmitting boundary is to apply an impedance to minimize the pressure wave reflection at the mesh boundary when we use a finite mesh to model the infinite domain. It is designed for Lagrangian but used often in ALE models to help reducing the model size.

In both cases, we first define a set of nodes (SPC) or segments (NON_REFLECTING). And then we setup the SPC cards or NON_REFECTING cards using those definitions. Let us use a case of underwater explosion to show the process. We have three multi-materials (fluids) in the model - a half sphere high explosive (HE), water above and around it, and then air on the top, as shown in the figure below. We construct a half symmetry model with symmetric plane aligning with -y face.



https://ftp.lstc.com/anonymous/outgoing/hao/sale/models_R121/underwater/

We follow the three-step setup. First mesh. A 24x12x16 box spans from (-12,0,0) to (12,0,16).

```
*ALE_STRUCTURED_MESH
$    mshid       pid       nbid       ebid      ityp    nparts
         1         9     200001     200001         0         0
$     nptx      npty       nptz       nid0     lcsid
      3001      3002       3003
*ALE_STRUCTURED_MESH_CONTROL_POINTS
      3001
                    1               -12.00
                   25                12.00
*ALE_STRUCTURED_MESH_CONTROL_POINTS
```

```
      3002
                    1                0.00
                   13               12.00
*ALE_STRUCTURED_MESH_CONTROL_POINTS
      3003
                    1                0.00
                   17               16.00
```

Next, multi-materials. Water, HE and Air. Please note, air has a reference pressure of 1.0e-5; water and HE have their reference pressure set to 0.

```
*ALE_STRUCTURED_MULTI-MATERIAL_GROUP
$# mmgname       mid     eosid                                          pref
  water           10        10
  He              11        11
  air             12        12                                        1.0e-5
*MAT_NULL
$#     mid        ro        pc        mu     terod     cerod        ym      pr
        10  1.000000
*EOS_GRUNEISEN
$#   eosid         c        s1        s2        s3     gamao         a      e0
        10  0.148000  1.750000     0.000     0.000  0.280000
$#      v0
  1.000000
*MAT_HIGH_EXPLOSIVE_BURN
$#     mid        ro         d       pcj      beta         k         g    sigy
        11  1.630000  0.784000  0.260000
*EOS_JWL
$#   eosid         a         b        r1        r2      omeg        e0      vo
        11  3.710000  0.032300  4.150000  0.950000  0.300000  0.043000 1.00000
*MAT_NULL
$#     mid        ro        pc        mu     terod     cerod        ym      pr
        12  0.001280
*EOS_LINEAR_POLYNOMIAL
$#   eosid        c0        c1        c2        c3        c4        c5
c6
        12     0.000       0.0     0.000     0.000  0.400000  0.400000
$#      e0        v0
   2.25e-5     0.000
```

And then, volume filling. First, all elements filled with water. Next, inside a sphere to HE. Then above a plane to air.

```
*ALE_STRUCTURED_MESH_VOLUME_FILLING
$    mshid                 to
         1              water
$ geometry
       ALL
*ALE_STRUCTURED_MESH_VOLUME_FILLING
$    mshid                 to
         1                 HE
$ geometry    in/out      NID1         r
 ELLIPSOID              199997       2.0
*ALE_STRUCTURED_MESH_VOLUME_FILLING
$    mshid                 to
         1               air
```

```
$ geometry    in/out      NID1      NID2
    PLANE               199998    199999
*NODE
  199997   0.0000000e+00   0.0000000e+00   0.0000000e+00
  199998   0.0000000e+00   0.0000000e+00  12.0000000e+00
  199999   0.0000000e+00   0.0000000e+00  15.0000000e+00
```

And let us be a Minimalist on *CONTROL_ALE.

```
*CONTROL_ALE
$#     dct      nadv      meth      afac      bfac      cfac      dfac     efac
                   1         1 -1.000000
$#   start       end     aafac     vfact      prit       ebc      pref   idebc
```

Now to apply boundary conditions on the six faces of the S-ALE mesh. They could be categorized into three types:

1. Symmetric plane: -y face. This translates to a *BOUNDARY_SPC on all nodes at that face along the local y direction.
2. No flow in and out: -z face. This face aligns perfectly to the seabed. Water and HE could move freely inside the plane. But nothing could flow in/out of the plane. This translates to a *BOUNDARY_SPC on all nodes at that face along the local z direction.
3. Transmitting boundary: all other 4 faces. At those four faces, we want to allow the pressure wave travel freely into the unmeshed infinite domain. We do that by adding a *BOUNDARY_NON_REFLECTING on all segments on those faces.

For a more user-friendly S-ALE setup, we added the following “macro-like” keyword to apply SPC constraints. It will be available in the next release in R12 (R12.1).

*BOUNDARY_SALE_MESH_FACE							
OPTION	MSHID	-X	+X	-Y	+Y	-Z	+Z
SYM	1			1			
NOFLOW	1					1	
NONREFL	1	1	1		1		1

Internally it is translated into the following keywords:

```
*BOUNDARY_SPC_SET
         1                    0         1         0
*BOUNDARY_SPC_SET
         2                    0         0         1
*SET_NODE_GENERAL
$       SID
         1
$    OPTION      MSHID        XMN        XMX        YMN        YMX        ZMN      ZMX
    SALEFAC          1                                1
*SET_NODE_GENERAL
$       SID
         2
```

```
$    OPTION      MSHID        XMN        XMX        YMN        YMX        ZMN      ZMX
   SALEFAC          1                                                       1
*BOUNDARY_NON_REFLECTING
$       SID
         11
*SET_SEGMENT_GENERAL
$       SID
         11
$    OPTION      MSHID        XMN        XMX        YMN        YMX        ZMN      ZMX
   SALEFAC          1          1          1                     1                   1
```

It is totally users' choice to use *BOUNDARY_SALE_MESH_FACE or not. Other than being concise, another advantage is that it is less error prone. Especially when the mesh is tilted, i.e. there is a local coordinate system. In this case, if we use *BOUNDARY_SPC, we need to remember to set up a local coordinate system, and make sure that we constraint the correct component. *BOUNNDARY_SALE_MESH_FACE, on the other hand, handles all those silently and automatically and takes off those unnecessary burdens from users.

One thing to note is that the two options SYM and NOFLOW, are doing the same thing. That is to constrain the flow perpendicular to the plane. We have both options available, simply to provide a one-to-one match between the options and real physical scenarios.

Another option, as you might have guessed, is FIXED. It is to fix all nodal motions at that face.

SALECPT and SALEFAC in *SET_?_GENERAL

In the model example above, when using the traditional *BOUDNARY_SPC and *BOUNDARY_NON_REFLECTING setup, one can notice that *SET_NODE_GENERAL and *SET_SEGMENT_GENERAL are used to generate node and segment sets, respectively. Unlike traditional ALE, S-ALE relies solely on *SET_?_GENERAL to create node/segment/element sets. This is because S-ALE mesh is generated internally so it is difficult, if not possible, to explicitly list out IDs of nodes/segments/elements. So what we do is to generate those sets by using SALECPT and SALEFAC options in *SET_?_GENERAL.

SALEFAC will pick up all nodes/segments/solids at certain S-ALE mesh face(s). The one below added all nodes at -y face of S-ALE mesh #1 into node set #1.

```
*SET_NODE_GENERAL
$       SID
          1
$    OPTION      MSHID        XMN        XMX        YMN        YMX        ZMN      ZMX
   SALEFAC          1                                 1
```

SALECPT provides a more flexible way. It picks up all nodes/segments/elements inside a box indexed by S-ALE control points. For example, this one below added the bottom half of nodes in the S-ALE mesh #1 into node set #101. (Remember our mesh is a box of 24x12x16 elements?)

```
*SET_NODE_GENERAL
$       SID
        101
```

```
$    OPTION      MSHID        XMN        XMX        YMN        YMX        ZMN      ZMX
   SALECPT          1          1         25          1         13          1        9
```

One thing we need to stress here. No other options in *SET_?_GENERAL could be applied onto S-ALE mesh. A common mistake is BOX option. The reason we did not support it was to avoid user mistakes. Most of the time Lagrange mesh and S-ALE mesh are on top of each other, i.e., overlaps in space. And more likely users tend to pick nodes inside a box for the Lagrange mesh; or the S-ALE mesh. Not both. Supporting BOX option might accidentally include S-ALE nodes into an intended Lagrange node set, without users even realizing that.

Initial Conditions

Initial conditions are relatively simple. Typically it is only to assign some initial velocity to some nodes. But still, there is some fine difference between LAG and ALE models. Again, we must emphasize that ALE material interface is not necessarily at the boundary. And it makes sense to apply initial velocities on certain nodes only if those nodes are the real spatial representation of a material.

Most commonly, initial velocities are applied simultaneously with volume filling process, through the *ALE_STRUCTURED_MESH_VOLUME_FILLING card. Like the following:

```
*ALE_STRUCTURED_MESH_VOLUME_FILLING
$#   mshid         -    ammgto         -   nsample         -         -       vid
         1               plate                                                 1
$#    geom    in/out     boxid
BOXCOR              0         1
```

And the initial velocity (vx=-61.631,vy=208.06) is prescribed by using the *DEFINE_VECTOR card.

```
*DEFINE_VECTOR
$#     vid        xt        yt        zt        xh        yh        zh       cid
         1   -61.631    208.06       0.0       0.0       0.0       0.0         0
```

The volume filling process will first fill the box with the multi-material named “plate” and then, find all nodes belongs to “plate” and assign an initial velocity to them.

Ending Remarks

LS-DYNA ALE module has been known for its steep learning curve. Partially it was because setting up Eulerian models are intrinsically different from Lagrange models. But the design of ALE keyword cards, for sure, has caused quite a lot of confusions among our users, new and experienced.

To prompt LS-DYNA ALE usages, Structured ALE solver introduced a new, user-friendly, streamlined three-step setup. We hope this effort could help users, new or old, to perform their work more efficiently and smoothly.