On Structured ALE Mesh Trimming

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 the mesh trimming feature.

Mesh regularity.

S-ALE solver expects a regular, box-shaped rectilinear mesh. This regularity enables an automated mesh generation by the solver, instead of pre-processor. And more importantly, this mesh regularity enables a simpler algorithm leads to a reduced simulation time and memory usage. It comes with a price though.

In the generic ALE solver, ALE mesh is generated by the user and could be of arbitrary shape. Users have the flexibility to mesh the ALE domain to better fit their needs. For example, to model a spherical blast one could generate a spherical ALE mesh domain. Another example is to model fuel sloshing in a rigid body container. One could choose to make the ALE outer surface conform to the Lagrange container. And then constrain the normal directional flow by using *ALE_ESSENTIAL_BOUNDARY.

The above two examples are without any fluid structure interactions. Now let us go to cases with FSI as these are what S-ALE targets to solve. Still on fuel sloshing problem, what we care is to see how the fuel tank deforms under a sudden fluid impact so container has to be flexible and FSI is needed. The shape of the container, for example, is somewhat very irregular. Let us say its height on the left side is much less than the right side, only ¼. To reduce the number of ALE elements, one might want to mesh the left side with less ALE elements. Just to make sure it covers the Lagrange container with several extra cushion layers of elements. As ALE runs are comparatively much more expensive, stinginess becomes a virtue.

From the above example, one could understand the motivation behind the development of mesh trimming feature. Its major objective is to keep the element cost low, by trimming the mesh to better fit the domain of interest. In most cases, to better fit the geometry of Lagrange structures

coupled to ALE fluids. Simply put, mesh trimming is to remove the elements far away from the structure or some geometric entities.

A first glance.

Let us see how it is done. Below is the format and description of the keyword *ALE_STRUCTURED_MESH_MESH_TRIM. There could be multiple trimming cards which are executed in the order of their appearances.

*ALE_STRUCTURED_MESH_MESH_TRIM							
MSHID	OPTION	OPER	OUT/IN	E1	E2	E3	E4

MSHID is the Mesh ID of the S-ALE mesh, defined in *ALE_STRUCTURED_MESH card. OPTION is to provide geometries to trim the S-ALE mesh and could be chosen from the following six: PARTSET, SEGSET, PLANE, CYLINDER, BOXCOR, BOXCPT and SPHERE. OPER has two choices: 0 to trim or 1 to keep. IN/OUT is to prescribe inside or outside; 0 – outside, 1 – inside. E1,E2,E3,E4 are to provide the geometric data and have different meanings for each different OPTION.

A picture is better than thousand words. The figure below shown a S-ALE mesh trimmed by using OPTION=”SEGSET”, OPER=”0” (trim) and OUT/IN=”0” (outside) with an offset of “10”. The segment set is the tube-like surface of red color. S-ALE mesh is on top and around the red tube, transparent and of a color of light blue.

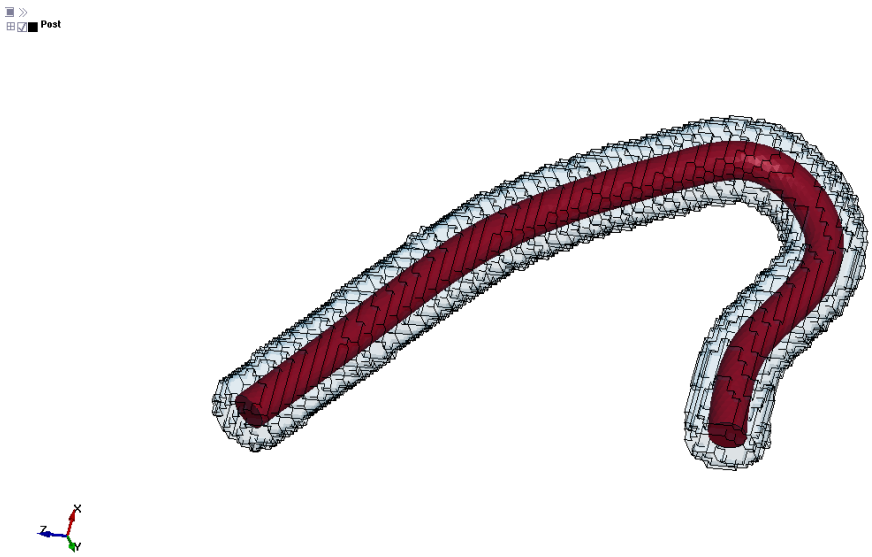


Figure 1. S-ALE mesh trimmed around “glue” (the red tube)

This model is done by Mukul Atri, Ansys ACE India, mukul.atri@ansys.com. The input deck could be found at https://ftp.lstc.com/anonymous/outgoing/hao/sale/models_R121/trim/tube_trim.k

Let us go through its setup to illustrate the usage of mesh trimming card. The initial mesh is a box mesh of 75x63x68 elements spans from (-150,-125,-135) to (150,125,135). There are two ALE multi-materials in this mesh – “glue” and “vacuum”. The segment set #2 is created from the

outer surface of a Lagrange dummy part; And used later to do the volume filling for ALE fluid "glue". We want to study how "glue" flows under compression (not included in this model for simplicity).

```
$=========================================================================
*ALE_STRUCTURED_MESH
$   meshid      dpid      nbid      ebid
         1         2
$    cpidx     cpidy     cpidz      nid0     lcsid
         1         2         3
*ALE_STRUCTURED_MESH_CONTROL_POINTS
$     cipd
         1
$        n                   x
         1                -150
        76                 150
*ALE_STRUCTURED_MESH_CONTROL_POINTS
$     cipd
         2
$        n                   x
         1                -125
        64                 125
*ALE_STRUCTURED_MESH_CONTROL_POINTS
$     cipd
         3
$        n                   x
         1                -135
        69                 135
$=========================================================================
*ALE_STRUCTURED_MULTI-MATERIAL_GROUP
$# mmgname       mid     eosid                                          pref
  glue          1001      1001
  vacuum        1002
*MAT_ALE_VISCOUS
      1001  1.000E-5   -1.0E10    2.0E-6       8.0   2.16E-2              0.201
*EOS_LINEAR_POLYNOMIAL
      1001       0.0    1000.0       0.0       0.0       0.0       0.0       0.0
       0.0       0.0
*MAT_VACUUM
      1002   1.0E-08
$=========================================================================
```

After setting up the above cards, we have a box mesh of 321,300 S-ALE elements, as shown below. There is quite some waste. All space not taken by "glue" (which occupies the same space as the red Lagrange tube) is going to be filled with "vacuum". Most of the space, "glue" will never flow into it. There is no point to keep all those elements far away from "glue". So using a box mesh mostly filled with vacuum is really something we strongly dislike.

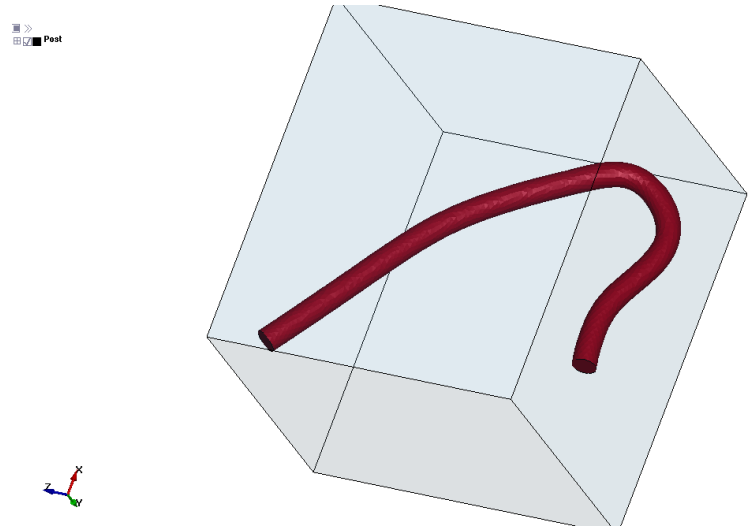


Figure 2. S-ALE box mesh untrimmed

To eliminate this waste, it is easy and straightforward. Simply adding the card below.

```
*ALE_STRUCTURED_MESH_TRIM
$     mshid     option       oper     out/in      segid     offset
          1     SEGSET          0          0          2       10.0
```

What it does is to remove S-ALE elements far away (10.0) from the outer surface of the red tube. The number of S-ALE elements is reduced to 14,173 from 321,300. Impressive, right? We would not expect the cost saving is always this significant, but still for special cases it could be quite surprisingly good.

*ALE_STRUCTURED_MESH_TRIM

Let us describe the design of *ALE_STRUCTURED_TRIM to get a better understanding of its usage.

*ALE_STRUCTURED_MESH_MESH_TRIM							
MSHID	OPTION	OPER	OUT/IN	E1	E2	E3	E4

1. Multiple entries allowed. Processed in the order of their appearances.
2. Five basic geometries: PLANE, CYLINDER, BOXCOR, BOXCPT and SPHERE.
3. Complicated geometries provided using PARTSET or SEGSET.
4. Operation be either trim (0) or keep (1) (or call it untrim, bring the trimmed elements back from previous trims)
5. OUT/IN: To operate on the elements outside (0) or inside (1) of the geometry For PARTSET and SEGSET options, “outside” is defined as the region to which the segment normal points.

Most of the above points could be easily understand by our users. Maybe with one exception: One might wonder what the purpose of operation “keep” is. Let us illustrate its usage by using

an airbag model. The full input deck could be found at the following directory. https://ftp.lstc.com/anonymous/outgoing/hao/sale/models/meshmotion/airbag1/

The S-ALE mesh is shown in the figure below. The mesh is composed of two boxes. The upper one has 7 layers of elements along z direction from 0.0315 to 0.2415; each layer contains 16x16 elements and spans from (-0.36, -0.36) to (0.36, 0.36). The bottom one has 3 layers from -0.0885 to 0.0315; each layer contains 6x6 elements and spans from (-0.135, -0.135) to (0.135, 0.135).

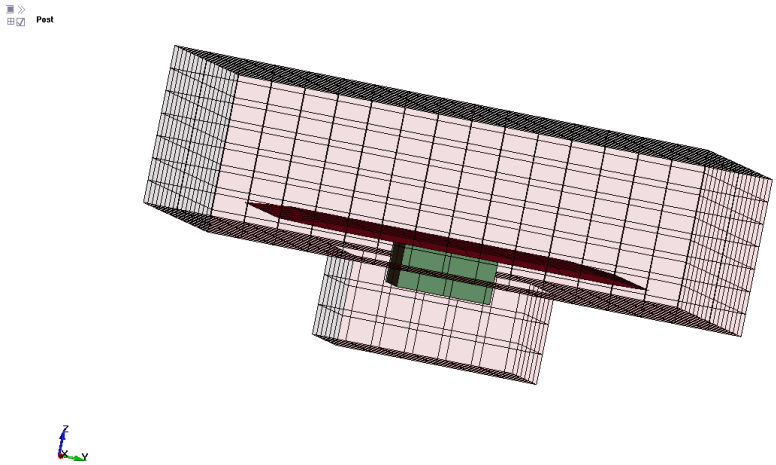


Figure 3. S-ALE mesh trimmed to have less elements at the bottom

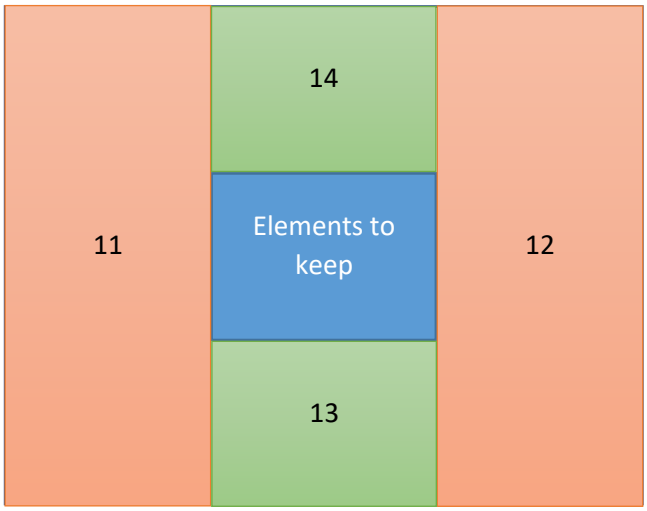
As always, let us define the ALE mesh first.

```
*ALE_STRUCTURED_MESH
$     mshid        pid       nbid       ebid
          1         11     100001     100001
$      nptx       npty       nptz       nid0      lcsid
       1001       1001       1003        272       1001
*DEFINE_COORDINATE_NODES
       1001        272        254        233          1
*ALE_STRUCTURED_MESH_CONTROL_POINTS
       1001
$                   x1                    x2
                     1                 -0.36
                    17                  0.36
*ALE_STRUCTURED_MESH_CONTROL_POINTS
       1003
$                   x1                    x2
                     1              -0.0885
                     4               0.0315
                    11               0.2415
```

Now we need to trim the mesh at the bottom away from the center. The most suitable choice is to use BOXCPT option to define the geometry. For now, let us forget about the “keep” operation and do the trim with “trim” operation only. The bottom mesh starts from node [6, 6] to node [12, 12] in the xy plane. So this means we need to trim 4 times, each time with a box of elements. One way to do it is as follows.

```
*ALE_STRUCTURED_MESH_TRIM
$     mshid    command      oper     out/in      boxid
          1     BOXCPT                             11
          1     BOXCPT                             12
          1     BOXCPT                             13
          1     BOXCPT                             14
*DEFINE_BOX
$     boxid       xmin       xmax       ymin       ymax       zmin       zmax
         11          1          6          1         17          1          4
         12         12         17          1         17          1          4
         13          6         12          1          6          1          4
         14          6         12         12         17          1          4
```

These four boxes are shown in the figure below.



The result fits our expectations and the process seems OK. But somehow you have a gut feeling that there must be a better way to do it, right?

*ALE_STRUCTURED_MESH_TRIM is designed to do addition or subtraction between geometries. That is where “keep” operation comes into play. The above card could be simplified to:

```
*ALE_STRUCTURED_MESH_TRIM
$     mshid    command      oper     out/in      boxid
          1     BOXCPT                              1
          1     BOXCPT         1          1         2
*DEFINE_BOX
$     boxid       xmin       xmax       ymin       ymax       zmin       zmax
          1          1         17          1         17          4         11
          2          6         12          6         12          1          4
```

It contains two operations. The first is to trim any element outside (OUT/IN=0) of the box 1 at the top. After this, all elements at the bottom 3 layers are deleted. That is not what we wanted – we still need the center 6x6 elements at the bottom. So let us reverse the trim operation, but only at those 6x6 elements. That is what the second card does here. It is to “keep” (untrim) the elements inside (OUT/IN=1) the box 2 which spans from node [6, 6] to [12,12] in the bottom 3 layers.

One can easily imagine the added flexibility it achieved by adding this “keep/untrim” operation. Most cases a complicated geometry could be described using addition/subtraction between 2 or at most 3 simple geometries. For a more detailed description of other options/flags in this keyword, please refer to LS-DYNA user’s manual.

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.