On Setting up a Structured ALE Model

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 of the S-ALE model setup.

PART is the problem.

This section is to give a background information on the introduction of a new keyword *ALE_STRUCTURED_MULTI-MATERIAL_GROUP. For new users never used previous ALE/S-ALE setup, please skip this section as it is irrelevant and could be confusing without knowing some history of changes in ALE setups.

"Part" definition is conveniently used in Finite element models to link the material definition and mesh of a Lagrange structure. A typically "PART" definition contains three things: SECTION, which is the integration rule; MAT+EOS, the material property; and mesh, linked by the PARTID and listed under *ELEMENT keyword. In a Lagrange simulation, "PART" has dual meanings – it refers to both the mesh and the material.

In the world of ALE, it is a little more complicated. As we are dealing with fluids, our point of view is rather Eulerian, not Lagrangian. This means the mesh and materials are not; and should not be bundled together. Mesh is no longer a spatial representation of material; and its boundary surface is no long material interface. Rather, the mesh, in an ALE simulation, is simply a spatial domain, to provide room for the fluids to occupy and flow. They are multiple fluids inside this mesh and associate this mesh with any one material property does not make any sense.

The general ALE solver borrowed "PART" definition. This caused quite some confusion in our users, even the most experienced users. It is not surprising at all, as sometimes "PART" refers the mesh; other times the material; and on several occasions, it refers to both mesh and materials.

When constructing the Structured ALE keywords, the author tried to separate this dual meanings "PART" definition into two distinguished definitions – "Mesh Part" and "Material Part". It helped

to some extent, but still considerable confusion still exists. As S-ALE solver is gradually picking up more usage, it becomes mandate to address this issue once and for all. Streamlining the S-ALE setup would save a lot of users' effort, especially for our new users not familiar with LS-DYNA keyword setups.

A new keyword, *ALE_STRUCTURED_MULTI-MATERIAL_GROUP was introduced recently. It is available in the latest beta version executable and will be in the next R12.1 release. This keyword no longer uses "PART" to link the material properties and hence eliminates the concept of "Material Part". The author believes it would prevent most common user setup mistakes.

Three step setup

We follow a straight-forward three step setup. First, mesh; secondly, material properties of fluids; thirdly, filling the mesh with fluids. In this section, we describe the three keywords doing these three steps.

1. Mesh generation: *ALE_STRUCTURED_MESH; *ALE_STRUCTURED_MESH_CONTROL_POINTS

In S-ALE, mesh is always rectangular. Obviously an automatically generated mesh would get rid of lots of unnecessary hassles in the model building and execution. So that was the route we took. To determine the mesh layout in the space, we need the following information:

a. The mesh spacing along three axes (LCIDX,LCIDY,LCIDZ).
b. The origin (NID0), and local coordinate system (LCSID).

Other fields are for identification purpose only and are self-explanatory. MSHID stands for mesh ID; DPID Part ID; NBID and EBID are the IDs of first S-ALE mesh node and element, respectively. TDEATH is to the "death time" for S-ALE mesh. It is to turn off the S-ALE calculation once the most of fluid loading is applied; and keep the Lagrange model running as the structure deformation is not fully developed yet.

*ALE_STRUCTURED_MESH							
MSHID	DPID	NBID	EBID				TDEATH
CPIDX	CPIDY	CPIDZ	NID0	LCSID			

The LCIDX, LCIDY and LCIDZ are IDs of *ALE_STRUCTURED_MESH_CONTROL_POINTS cards. Each card specifies the mesh spacing along one axis.

*ALE_STRUCTURED_MESH_CONTROL_POINTS							
CPID							
N1		X1		RATIO1			
…		…		…			
Nn		Xn		RATIOn			

The idea of the "_CONTROL_POINTS" card is simple. Let us forget about the "RATIO" field which is for progressive mesh spacing. This card contains some number of (N,X) pairs. And each (N,X) pair means "the Nth node's coordinate is X". Between any two consecutive (N,X) pairs mesh is evenly distributed. Take the simplest case, say 10 elements between [0,1]. A simple two pairs setup of (1,0.) and (11,1.) is sufficient. To another extreme, say we really want some insane irregular mesh. We could make 11 pairs of (N,X) like (1,0.), (2,0.07), (3,0.13), (4,0.2), (5,0.26), (6,0.37), (7,0.47), (8,0.53), (9,0.79), (10,0.8), and finally (11,1.).

The progressive mesh spacing is something worth a separate article by itself and hence skipped in this introductory paper. It is powerful but conceptually not so user-friendly. Efforts are being continuously made to improve this part. Please stay tuned for the updates.

2. Material definitions: *ALE_STRUCTURED_MULTI-MATERIAL_GROUP

S-ALE mesh is simply a spatial domain in which fluids flow. In order to let the code know what and how many fluids there are, we need to provide material properties of each fluid and list them all under the card *ALE_STRUCTURED_MULTI-MATERIAL_GROUP. Please note, AMMG stands for ALE multi-material group, a rather alternative and maybe confusing name for "ALE fluid". In this paper we are going to use AMMG and fluid interchangeably.

*ALE_STRUCTURED_MULTI-MATERIAL_GROUP							
AMMGNM1	MID1	EOSID1					PREF1
…	...	…					…
AMMGNMn	MIDn	EOSIDn					PREFn

"AMMGNM" is a name one gives to a AMMG (ALE Multi-Material Group), aka ALE fluid. It is used in other cards, for example, *SET_MULTI-MATERIAL_GROUP_LIST to refer to that AMMG. "MID" and "EOSID" are the material ID and EOS ID, respectively.

"PREF" is to describe the reference pressure or "base pressure" of that fluid. This might be somewhat new to our typical users from solids background. Pressure of a solid material, if not preloaded, always starts from zero. In such case, its reference pressure or base pressure, is zero. But most fluids have non-zero reference pressure. For example, air has a base pressure of 101325 Pa (1 bar atmospheric pressure). Traditionally this reference pressure is prescribed using the field "PREF" in *CONTROL_ALE card. The new *ALE_STRUCTURED_MULTI-MATERIAL_GROUP has a design to allow each AMMG to have its own reference pressure. The author believes this added flexibility could be proven very useful in certain applications.

3. Volume Filling: *ALE_STRUCTURED_MESH_VOLUME_FILLING

Now we have a rectilinear S-ALE mesh, which is our calculation domain. And we have several fluids which resides in this domain. Before carrying out our simulation, there is still one critical information missing. That is: how are those fluids occupying the domain? This information is given as the form of volume fraction per element. Basically we need to assign the volume fraction

for each AMMG (ALE fluid) for each element. Say one element is fully occupied by AMMG "water", then the volume fraction of that element is (1, 0, 0), assuming the ALE materials are ("water", "air", "vacuum"). Similarly (0.6,0.3,0.1) if "water" 60%, air 30% and vacuum 10%. For each element, the code needs to know the volume fraction of all ALE fluids.

The first way is to explicitly list volume fractions for each element. This could be done through a keyword card called "*INITIAL_VOLUME_FRACTION". This approach is seldomly used as first, it is tedious and secondly, too much burden on users.

The much easier solution is to automatically generate the volume fraction information based on certain user supplied geometries. These geometries could be simple shapes like sphere, plane, box, cylinder. Or it could be user-defined complex shape like structure surfaces. With certain geometric shape defined, users could then fill the inside or outside of that shape with certain ALE fluid. The volume fractions are internally calculated, stored and then used in the subsequent simulation to reconstruct the fluid interface.

This is done by using the keyword "*ALE_STRUCTURED_MESH_VOLUME_FILLING". The "volume filling" process typically is done through multiple "tasks", each task by a separate keyword.

*ALE_STRUCTURED_ MESH_VOLUME_FILLING							
MSHID		AMMGTO		NSAMPLE			VID
GEOM	IN/OUT	E1	E2	E3	E4		

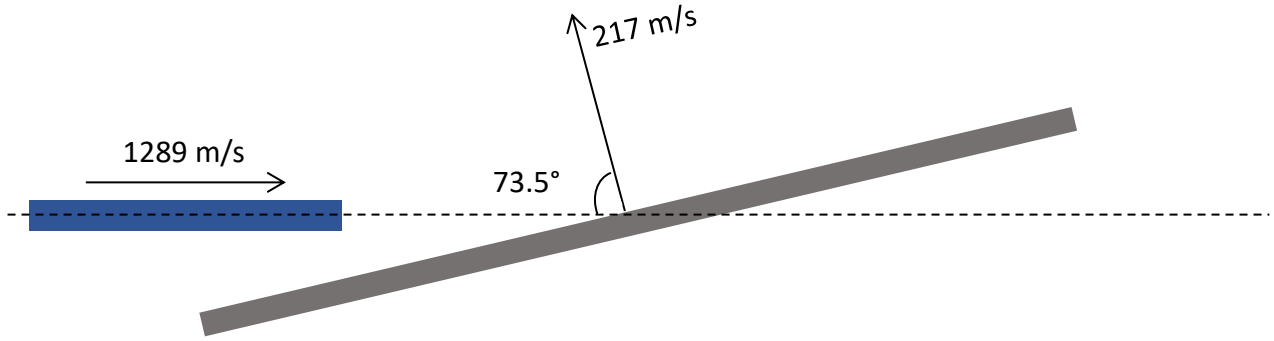
MSHID is the S-ALE mesh ID, defined in *ALE_STRUCTURED_MESH card. And AMMGTO is the name of ALE fluid to be filled, defined in *ALE_STRCUTURED_MULTI-MATERIAL_GROUP. VID is to prescribe the initial velocity of that fluid, if any. And NSAMPLE is default to 3 which means that one ALE cell is divided into 7x7x7 (7=2*3+1) sub-cells and each sub-cell is checked to see if it is inside/outside.

It supports 5 basic geometries: Plane, Cylinder, Ellipsoid, Box with indices and Box with coordinates. And E1-E4 are used to provide information of these geometries. For complicated geometries, we need users to provide us with a segment set (or something we could internally convert to a segment set). The assumption is that all segment normals are consistent; and those normals point to the "inside". For convenience, we provide a "IN/OUT" flag for an easier flip.

A Simple Example

Now let us use a simple example to illustrate this 3-step process. It is to model a long rod projectile impacting an oblique steel plate (Fugelso & Taylor 1978). The dimensions were from ARL-TR-2173 (Schraml & Kimsey 2000) and material properties from "Numerical Simulation of High-Velocity oblique Impacts of Yawed Long Rod Projectile Against Thin-Plate" (Yo-Han Yoo 2002).

Below is a sketch showing a projectile at 1289m/s and hitting a plate which is moving up at 217 m/s. Projectile has a length of 76.7mm and a diameter of 7.67mm. Plate has a thickness of 6.4 mm, a width of 60mm and a length of 150mm. The unit used is mm-g-s.



Step 1: To construct a mesh spans (-107.5,-30,-15) to (107.5,30,15) with 215x60x30 elements.

```
*ALE_STRUCTURED_MESH
$    mshid       pid      nbid      ebid
         1        11    100001    100001
$    cpidx     cpidy     cpidz      nid0     lcsid
      1001      1002      1003
*ALE_STRUCTURED_MESH_CONTROL_POINTS
$     cpid
      1001
$                  N                   X
                   1            -107.5000
                 216             107.5000
*ALE_STRUCTURED_MESH_CONTROL_POINTS
$     cpid
      1002
$                  N                   X
                   1             -30.000
                  61              30.000
*ALE_STRUCTURED_MESH_CONTROL_POINTS
$     cpid
      1003
$                  N                   X
                   1            -15.0000
                  31             15.0000
```

Step 2: Set up ALE multi-material Groups (AMMGs or ALE fluids). There are totally 3 AMMGs defined. First is “Rod” by *MAT_JOHNSON_COOK (MID=1) and *EOS_LINEAR_POLYNOMIAL (EOSID=1). Next is “Vacuum” by *MAT_VACUUM (MID=3). The third is “Plate” by *MAT_JOHNSON_COOK (MID=2) and *EOS_LINEAR_POLYNOMIAL (EOSID=2). Their materials properties are given as follows.

```
*MAT_JOHNSON_COOK
$      MID        RO         G         E        PR       DTF        VP    RATEOP
         1   18.6e-3  63.692e3   165.6e3       0.3
$        A         B         N         C         M        TM        TR      EPS0
```

```
   1.079e3    1.12e3      0.25    0.0070       1.0      1473       283   1.0e-3
$       CP        PC     SPALL        IT        D1        D2        D3        D4
     130.0
$       D5      C2/P      EROD     EFMIN    NUMINT

*EOS_LINEAR_POLYNOMIAL
$    eosid        c0        c1        c2        c3        c4        c5        c6
         1     0.000  138.00e3
$       e0        v0
     0.000     0.000
*MAT_JOHNSON_COOK
$      MID        RO         G         E        PR       DTF        VP    RATEOP
         2   7.87e-3  76.692e3  200.1e3       0.3
$        A         B         N         C         M        TM        TR      EPS0
   0.792e3   0.510e3      0.26    0.0140      1.03      1809       283    1.0e-3
$       CP        PC     SPALL        IT        D1        D2        D3        D4
     480.0
$       D5      C2/P      EROD     EFMIN    NUMINT

*EOS_LINEAR_POLYNOMIAL
$    eosid        c0        c1        c2        c3        c4        c5        c6
         2     0.000  166.75e3
$       e0        v0
     0.000     0.000
*MAT_VACUUM
$      MID
         3    1.0e-9
```

Now construct the ALE fluids by using *ALE_STRUCTURED_MULTI-MATERIAL_GROUP.

```
*ALE_STRUCTURED_MULTI-MATERIAL_GROUP
$     name       mid     eosid                                             pref
      rod          1         1
   vacuum          3
    plate          2         2
```

Step 3: Volume Filling the initial S-ALE mesh. By *ALE_STRUCTRUED_MESH_VOLUME_FILLING.

First, fill all with “vacuum”.

```
*ALE_STRUCTURED_MESH_VOLUME_FILLING
$#   mshid         -    ammgto         -   nsample         -         -     vid
         1              vacuum
$#    geom    in/out
ALL               0
```

Next, assign the inside of box (BOXID=1) to “plate”.

```
*ALE_STRUCTURED_MESH_VOLUME_FILLING
$#   mshid         -    ammgto         -   nsample         -         -       vid
         1              plate                                                 1
$#    geom    in/out     boxid
BOXCOR            0         1
*DEFINE_BOX_LOCAL
$#   boxid       xmn       xmx       ymn       ymx       zmn       zmx
```

```
         1       0.0     150.0       0.0       6.4       0.0        30
$#      xx        yx        zx        xv        yv        zv
   0.95882   0.28402       0.0  -0.28402   0.95882       0.0
$#      cx        cy        cz
  -71.0026  -24.3694     -15.0
```

Finally, assign the inside of a cylinder to “rod”.

```
*ALE_STRUCTURED_MESH_VOLUME_FILLING
$#   mshid         -    ammgto         -   nsample         -         -       vid
         1               rod                                                  2
$#    geom    in/out      nid1      nid2    radii1    radii2
CYLINDER            0         1         2     3.835     3.835
*NODE
$#   nid               x               y               z      tc      rc
       1          -103.0             0.0             0.0       0       0
       2          -26.33             0.0             0.0       0       0
```

And the initial velocities of “plate” and “rod” are prescribed by using the following cards.

```
*DEFINE_VECTOR
$#     vid        xt        yt        zt        xh        yh        zh       cid
         1   -61.631    208.06       0.0       0.0       0.0       0.0         0
         2    1289.0       0.0       0.0       0.0       0.0       0.0         0
```

That is all! But before we could make it run, we need to add in some more control card and database cards. We could use *CONTROL_ALE to make a choice of order of accuracy, 1st order (donor cell) or 2nd order (van Leer).

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

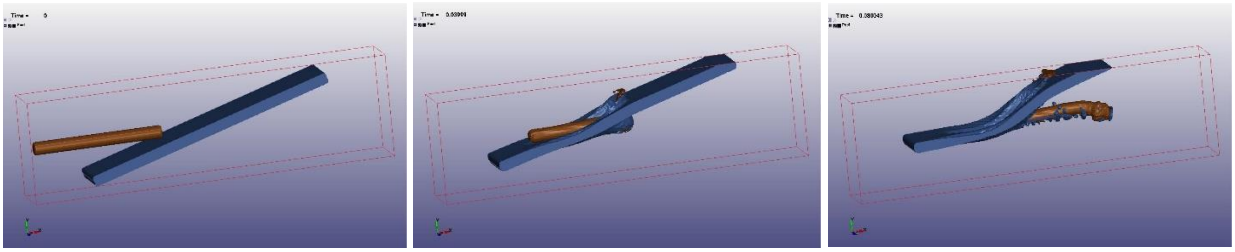
The rest of fields should be left as blank most of the time, if not all. They are used in generic ALE solvers but mostly ignored in S-ALE solvers with only a few exceptions.

And other cards:

```
*CONTROL_TERMINATION
$#  endtim    endcyc     dtmin    endeng    endmas
 0.0800000
*CONTROL_TIMESTEP
$#  dtinit    tssfac      isdo    tslimt     dt2ms      lctm     erode     ms1st
     0.000  0.600000
$#  dt2msf   dt2mslc     imscl
     0.000         0         0
*DATABASE_BINARY_D3PLOT
$#      dt      lcdt      beam     npltc    psetid
  0.001000
```

Now run it with 16 core MPP executable. It will only take 39 seconds. Please note as not all S-ALE keywords are finalized until recently (Jan 2021), we need to use the latest DEV version of LS-DYNA. R12.1 version should be available in the first half of year 2021.

Below is the projectile and plate at t=0., t=0.04ms and t=0.08ms. The input deck is available at https://ftp.lstc.com/anonymous/outgoing/hao/sale/models_R121/cthrod.k



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.