On Setting up Multi-Materials 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 S-ALE multi-materials.

Multi-materials.

ALE Multi-materials, or ALE multi-material groups (AMMG), has been used throughout LS-DYNA user manuals and other documentations. This name, by itself, does not mean too much. It was just to follow the naming convention several decades old. The first generation ALE solver only supported one single material so it was called "single material" element formulation (ELEFORM=5 in *SECTION_SOLID). Then the concept of "volume fraction" was introduced and then a "single material with void" element formulation was implemented (ELEFORM=12). Based on the element volume fraction, we could use an interface reconstruction technique to regenerate material interface between that material and void.

Around the same time, the ALE developer found the same technique could be extended to deal with multiple materials flowing inside the ALE mesh. The idea is simple. In single material with void formulation, in each element we have a "volume fraction" which represents the ratio between the material volume and the total element volume. And later after advection, we use that information to reconstruct the material void interface in that element. Now we could simply allow for multiple materials; each has its own element volume fraction value; and each goes through the interface reconstruction process – to find the interface between this material and all other materials. We will cover that process a little bit more later as it is quite important to understand the process to set up the keyword right.

Then the multi-material element formulation (ELEFORM=11) was implemented. And these materials occupying the ALE mesh are referred as "ALE multi-materials". To the author, a more intuitive and shorted name "ALE fluids" might be more appropriate. In the following discussion, "ALE fluids" and "ALE multi-materials" are used interchangeably.

Multi-Material Setup in the Three step setup

S-ALE follows a straight-forward three step setup. First, mesh; secondly, material properties of fluids; thirdly, filling the mesh with fluids. For a detailed description of this 3-step setup, please refer to “On setting up a S-ALE model” published on FEA information Jan 2021 issue.

We set up the ALE fluids through two types of keywords: *MAT_+*EOS to provide material properties; and *ALE_STRUCTURED_MULTI_MATERIAL_GROUP to provide the list of ALE fluids and their order of interface reconstruction.

We will cover the material definitions in the next section and concentrate here on the card *ALE_STRUCTURED_MULTI_MATERIAL_GROUP. The card is of the following format.

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

“MID” and “EOSID” are the material ID and EOS ID, respectively. *MAT_+*EOS_ provides the complete material properties of 1 ALE fluid. Almost ALE fluids need *EOS properties.

“AMMGNM” is a name one gives to a AMMG (ALE Multi-Material Group), aka ALE fluid. It is a character string and case insensitive. The leading and trailing spaces are trimmed so alignment is not an issue. We suggest short descriptive names such as “soil”, “water”, “air”, “HE”; or sometimes “inAir”, “outAir”, “airBelow”, “airAbove”, etc. It will be cross-referenced in other keywords where AMMG needs to be specified. For example, *SET_MULTI-MATERIAL_GROUP_LIST.

“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 idea of reference pressure will be covered more in the next section.

A key point, not well known among even the most experienced ALE users, is that the order of AMMG definitions matters. It plays an important role in the interface reconstruction. And a wrongfully ordered *ALE_STRUCTURED_MULTI_MATERIAL_GROUP card could lead to unphysical behaviors.

The interface reconstruction algorithm is designed to construct one interface between two fluids. It generates a plane cut inside an ALE element; puts one fluid on one side of that plane, another on the other side. When the volumes of these two polygons match the volumes of two fluids, we are all settled there. This algorithm needs to be extended to deal with more than two fluids. For two fluids, we generate one interface. Say three fluids, two interfaces. How should we proceed?

“Onion skin” was adapted. We do one interface at a time. First, we generate the interface between fluid #1 and other fluids. Next, between fluid #1+ fluid #2 and other fluids. Then fluid #1+fluid#2+fluid#3 and others, so on and so forth. It is like peeling off the onion skin, first we peel #1 off, then #2, then #3. Only now we are constructing #1 first and then stack #2 up and then #3.

Apparently, in order for this process to make sense, an implied assumption needs to be made. That is, all these fluids listed next to each other must also be physically next to each other. Otherwise, the interface reconstruction will put fluids out of its real location.

For example, we have “air”, “water”, “soil” from up to bottom and we listed them in that order in *ALE_STRUCTURED_MULTI-MATERIAL_GROUP card. In the mixed cell contains all three fluids, we generate the “air” interface first. “Air” is now at the top. And then “air”+”water”, so “water” now is below “air”. And then “soil” is put at the bottom. Perfect, right? But how about if we list “air” on the first line, but switch the order between “water” and “soil”? Do you see the problem?

OK. First we do “air”. “Air” is at the top. Next “air”+”soil” as “soil” is on the second line. Without knowing better, our interface construction algorithm puts “soil” next to “air” at the center. Then “water” at the bottom. We would suddenly find that after advection, there are some “water” mysteriously moved to the bottom of the element. WHY? The order matters.

Not surprisingly the combination of lack of information and limitation in the interface reconstruction algorithm led to lots of strange ALE results. I apologize for the trouble it has brought to our users. As we did not do well on passing this important piece of information effectively to our users. To all readers of this article, I would greatly appreciate if you could forward it to our fellow ALE users.

Material Properties

Most our users are from solid Mechanics background. Same for me. What we care is to apply the fluid loadings onto our structure. The structural response is what we are after. ALE is never intended to solve “CFD” kind of problems. Rather it is to convey a pressure wave through certain fluid(s) and then load it onto a structure.

But in order to do that well, we still need to understand a little bit about fluids. Majorly, how it is different to solids. The first thing here is Equation-of-State. Why almost always we have EOS definition in ALE fluids?

Let us start with internal energy. What is internal energy? Ask a structural engineer, he would say it is the energy deposited into the material by external load. A block free-resting on the ground has zero internal energy. And internal energy starts to increase when load is applied. But is this also true for fluids? Uncompressed fluid has zero internal energy? Take air, can we say it has zero energy when not compressed?

What us structural engineers referred to as “internal energy” is, in fact, “internal energy difference” or “change in internal energy”. We pay no attention to the real internal energy of a block not loaded. What we have here is the external work = -internal energy change. The initial internal energy does not matter.

But in certain fluids, like air, we do have a “real” internal energy. And this internal energy contributes to the pressure value. And “EOS_” provides us the way to define the pressure dependence on the internal energy.

Typically in Solid Mechanics, *MAT_ itself is sufficient to describe the whole material behavior. A stress field is divided into dilatational and deviatoric. Dilatational linked to bulk modulus and deviatoric to shear modulus. And in case of plasticity, dilatational we have no pressure change as it is incompressible. There is no internal energy dependency as it does not play any role in solids behavior.

Fluids are different. First, compression always leads to pressure change. (Please note as ALE is a hydrocode we do not allow incompressibility.) Secondly, pressure depends on internal energy. So for fluids, pressure is a function of compressibility and internal energy. And here comes the *EOS_ keywords to let users prescribe that function.

Take water as an example. Water, when not heavily loaded, could be assumed to be of a linear material with no internal energy dependence. *EOS_LINEAER_POLYNOMIAL (*EOS_01) with bulk modulus defined is sufficient.

```
*MAT_NULL
$#     mid        ro        pc        mu     terod     cerod        ym        pr
         1     998.0       0.0       0.0       0.0       0.0       0.0       0.0
*EOS_LINEAR_POLYNOMIAL
$#   eosid        c0        c1        c2        c3        c4        c5        c6
         1       0.0     2.2e9       0.0       0.0       0.0       0.0       0.0
$#      e0        v0
       0.0       1.0
```

Deviatoric stress only comes from viscous shear for water. This viscosity coefficient (mu) could be defined in *MAT_NULL. For pressure field, we have $p = c_1({}^{\rho}/_{\rho_0} - 1)$. All other polynomial coefficients c_i are zero. e_0 is the internal energy density (per unit volume); and $v_0 = {}^{\rho_0}/_{\rho(t)}, t = 0$ is the compression ratio, at the initial state, respectively. Please note here ρ_0 is the uncompressed material density, not the initial material density. The above setup did two things. First, it defined material properties. Secondly, it also prescribed initial pressure to be 0. The later could be easily overlooked.

So what if we want to prescribe a non-zero initial pressure? For this water, it is simple. We back solve for the initial compression ratio. Say in an ALE model we have two fluids—air and water. To have a pressure equilibrium in the initial state, water needs to have a pressure of 1 bar=101,325 Pa, same as air. We let $v_0 = {}^{\rho_0}/_{\rho(t)} = \frac{1}{1+{}^{p}/_{c_1}} = \frac{1}{1+{}^{101325}/_{2.2e9}} = 0.9999539453$.

Now let us do air. Air is with idea gas law. We could either define it with *EOS_LINEAR_POLYNOMIAL (*EOS_01) or *EOS_IDEAL_GAS (*EOS_12). Let us use *EOS_01.

```
*MAT_NULL
$#     mid        ro        pc        mu     terod     cerod        ym        pr
         2      1.23       0.0       0.0       0.0       0.0       0.0       0.0
*EOS_LINEAR_POLYNOMIAL
$#   eosid        c0        c1        c2        c3        c4        c5        c6
         2       0.0       0.0       0.0       0.0       0.4       0.4       0.0
$#      e0        v0
  253312.5       1.0
```

Same we have no deviatoric stress and for pressure $p = [c_4 + c_5({}^{\rho}/_{\rho_0} - 1)] * E = 0.4 * {}^{\rho}/_{\rho_0} * E$. Ring a bell? Idea gas law has a $\gamma = 1.4$ and $p = (\gamma - 1) * {}^{\rho}/_{\rho_0} * E$. Its initial pressure of 101,325 Pa is prescribed by assigning internal energy density to $e_0 = \frac{p}{0.4} = 253312.5$ as ${}^{\rho}/_{\rho_0} = 1$.

Take another example, high explosive (HE). It is defined by *MAT_HE + *EOS_JWL (*EOS_02).

```
*MAT_HIGH_EXPLOSIVE_BURN
$#     mid        ro         d       pcj      beta         k         g      sigy
         2    1630.0    6930.02.10000E10       0.0       0.0       0.0       0.0
*EOS_JWL
$#   eosid         a         b        r1        r2      omeg        e0        vo
        23.71200E113.231000E9      4.15      0.95       0.37.000000E9       0.0
```

Again, we assume no deviatoric stress and pressure is a function of both compression ratio and internal energy. $p = A\left(1 - \frac{\omega}{R_1 V}\right)e^{-R_1 V} + B\left(1 - \frac{\omega}{R_2 V}\right)e^{-R_2 V} + \frac{\omega E}{V}$, where $V = {}^{\rho_0}/_{\rho}$. It contains two exponential decay terms and an idea gas term to model gaseous explosion process. *MAT_HE controls burn fraction. Basically the ratio of burning at current state. And the pressure is burn fraction times the *EOS_JWL pressure. If a piece of HE is fully burnt, the pressure is simply *EOS_JWL pressure. If not burnt at all, the pressure is zero. At the initial state, everywhere HE is not detonated so everywhere HE has a pressure of zero. From the design of *MAT_HE and *EOS_JWL, we could see that the pressure here is rather gauge pressure than the "real" pressure and they are differed by 1 bar.

Wait. Then how about if we have both air and HE inside an ALE model? Air has an initial pressure of 1 bar (it has to. otherwise how does it expand?). And HE has an initial pressure of … 0! There is no way to reach pressure equilibrium at the first state.

This pressure incompatibility is there for real and we have been living with it for quite some long time. To our defense, HE pressure is of magnitudes higher than air so this discrepancy is considered small, if not tiny. But now we have a way to resolve it. We will see that later in this section.

Reference Pressure

Here is another new to us structural engineers. Reference pressure. We go back to that free-sitting block on the ground. What is its boundary condition? It has a z-direction constraints on nodal motion at the bottom face, right? Is that all?

Right, and wrong. Indeed we only need to constrain z motion at the bottom face to make the simulation right. But one thing most people would overlook is that at all other 5 faces we have pressure boundary condition. Pressure at those 5 faces is … zero. This boundary condition is naturally satisfied. We are getting used to it so much that we never remember there is a boundary condition at those faces.

Now let us continue working on the air, water, HE model. In the previous section, we set up the material definitions and prescribed the initial pressure of each material. Take air, it has an initial pressure of 1 bar and occupies the upper portion of the box-shaped S-ALE mesh. Do we need to apply boundary conditions, say, at the top face?

We must. The air pressure is 1 bar. Without applying a 1 bar pressure boundary condition, the inside air would expand and fly out of the S-ALE mesh. After some time, all air flows out and the run crashes.

One way to apply this pressure boundary condition is to pick all surface ALE segments and apply pressure loadings on those segments through *LOAD_SEGMENT. This is straightforward and easy to understand. But it is a little bit costly. So we took another approach. Instead we subtract this 1 bar pressure from all elements' pressure when evaluating the internal force. As internal force is only affected by pressure difference, not the real pressure, the nodal force of internal nodes remains the same. To surfaces nodes, subtracting 1 bar off from element pressure is equivalent to applying a 1 bar pressure loading. This is a much faster and more efficient way computational cost wise.

So what we do is, instead of applying pressure boundary conditions, prescribing a reference pressure for each ALE fluid. This brings us a well preserved pressure equilibrium, between ALE fluids and outside.

In addition to that, the newly added *ALE_STRUCTURED_MULTI-MATERIAL_GROUP card had a design to reconcile pressures between different ALE fluids. In last section's example, water and air pressure both being 1 bar at the initial stage; but HE has a base pressure of 0 bar. Previously, the ALE setup only allows for one global reference pressure (PREF in *CONTROL_ALE). So in this case, in elements occupying by HE, pressure was wrongfully subtracted by 1 bar pressure in internal force calculation. *ALE_STRUCTURED_MULTI-MATERIAL_GROUP, to address this problem, allows for each ALE fluid to have its own reference pressure. We can easily reach pressure equilibrium by assigning reference pressure 1 bar to air and water, 0 to HE. Please refer to the example below to better understand this idea.

A Simple Example

We have three ALE fluids, water, air and HE. And a box-shaped S-ALE mesh. Air occupies the upper half; water the lower half; and HE is a small cylinder submerged in the water. Below is the ALE multi-material keywords setup.

Two points to note here.

1. Order matters. HE in the first line, water next, air the third. Of course, we could go the other way around, like air first, then water, HE last.
2. Each ALE fluid has its own reference pressure. Which is the pressure difference between itself and the outside world. For air and water, it is 1 bar. And for HE, it is 0.

```
*ALE_STRUCTURED_MULTI-MATERIAL_GROUP
$# mmgname       mid     eosid                                               pref
   HE              3         3                                                0.0
water              1         1                                           101325.0
    air            2         2                                           101325.0
*MAT_NULL
$#     mid        ro        pc        mu     terod     cerod        ym        pr
         1     998.0       0.0       0.0       0.0       0.0       0.0       0.0
*EOS_LINEAR_POLYNOMIAL
$#   eosid        c0        c1        c2        c3        c4        c5        c6
         1       0.0     2.2e9       0.0       0.0       0.0       0.0       0.0
$#      e0        v0
       0.0       1.0
*MAT_NULL
$#     mid        ro        pc        mu     terod     cerod        ym        pr
         2      1.23       0.0       0.0       0.0       0.0       0.0       0.0
*EOS_LINEAR_POLYNOMIAL
$#   eosid        c0        c1        c2        c3        c4        c5        c6
         2       0.0       0.0       0.0       0.0       0.4       0.4       0.0
$#      e0        v0
  253312.5       1.0
*MAT_HIGH_EXPLOSIVE_BURN
$#     mid        ro         d       pcj      beta         k         g      sigy
         2    1630.0    6930.02.10000E10       0.0       0.0       0.0       0.0
*EOS_JWL
$#   eosid         a         b        r1        r2      omeg        e0        vo
         23.71200E113.231000E9      4.15      0.95       0.37.000000E9       0.0
```

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.