LS-DYNA® Structured ALE (S-ALE) Solver

Hao Chen

Livermore Software Technology Corporation

Abstract

A new Structured ALE solver is recently added into LS-DYNA. It targets to solve ALE problems with structured mesh. It supports all parallel versions of LS-DYNA, i.e., SMP, MPP and MPP Hybrid.

The new solver generates structured ALE mesh automatically. The elements do not have to be of equal size. Instead, a user can define element spacing according to needs of the specific engineering problems. Listing all the elements and nodes used for ALE mesh in the input deck becomes unnecessary. Rather, we can simply define the mesh geometry by a number of control point pairs and let the automated mesh generator do the work. For larger ALE problems, the input deck size will be greatly reduced. The execution time on reading in keyword and writing out structured input decks are minimized. Also, changing mesh geometry becomes much simpler.

*The new S-ALE solver is easy to use, especially for users acquainted to the old generic ALE solver. The solver is automatically invoked with the generated structured mesh. This process is transparent to users. Most ALE keywords remain the same with exception of three new keywords. They are *ALE_STRUCTURED_MESH, *ALE_STRUCUTURED_MESH_CONTROL_POINTS and *ALE_STRUCTURED_MESH_REFINE.*

Introduction

The LS-DYNA R9.0 release contains a new Structured ALE solver. The solver is dedicated to solve ALE problems which use structured mesh. Structured mesh, sometimes also referred as IJK mesh, has a shape of rectangular box and contains a grid of cuboids. The elements are not necessarily evenly distributed even though quite often the grid is evenly spaced along one direction.

The new solver generates ALE mesh based on user-provided geometric information at the initial stage of the execution. Users no longer need to generate mesh and save it to keyword file. And LS-DYNA no longer needs to read the ALE mesh keyword file back to start the run. While this might seem trivial for smaller models, for larger problems, say ones with more than 10 million elements, it could be a both time and resource consuming process.

Once the structured ALE mesh is generated, the S-ALE solver will be automatically invoked. The new S-ALE solver was implemented based on the exact same theory which the general ALE solver used. It employs the same advection process to remap the element and nodal quantities from deformed mesh to ALE mesh in each time step. It uses the same interface construction technique to determine multi-material interfaces in a mixed element.

But the S-ALE solver was implemented separately from the general ALE solver. Structured mesh has its merits. Its geometric information could be represented in a much smaller database. The element connectivity is patterned so data dependency is much more predictable. Sorting and searching become trivial arithmetic operations. To fully take advantage of these characteristics, the algorithms re-implemented with new coding to achieve optimal performance.

The structured ALE solver supports MPP. ALE calculations were re-organized and re-grouped to better accommodate the needs for MPP communications to achieve an enhanced performance. The structured ALE solver also supports SMP, thanks to the reduced complexity in data dependency brought by structured mesh. Consequently it also has the MPP hybrid capability. It means in a run, each MPP process could contain multiple SMP threads. We expect that MPP hybrid would be proven quite useful in the coming years accompanied by the hardware improvement and growth in model size.

This paper is organized as follows. First we introduce the new keywords and their usage. Next we describe the S-ALE setup process by using a simple small model. Then we discuss S-ALE's performance with numerical results from several test runs. Lastly we end this paper with some concluding remarks.

Keywords

There are three new keywords added. They are: * ALE_STRUCTURED_MESH, *ALE_STRUCTURED_MESH_CONTROL_POINTS and * *ALE_STRUCTURED_REFINE.

***ALE_STRUCTURED_MESH_CONTROL_POINTS**

The whole purpose of * ALE_STRUCTURED_MESH _CONTROL_POINTS card is to provide mesh geometry for *ALE_STRUCTURED_MESH card at three local directions. Its format is as follows.

* ALE_STRUCTURED_MESH_CONTROL_POINTS					
CPID			SFO		OFFO
A1		O1			
A2		O2			
…		…			

CPID is the ID of the CONTROL_POINTS card; each (A1, O1) defines a (node ID, coordinate) pair; SFO and OFFO are the scale factor and the offset to the ordinate values, respectively.

For evenly spaced mesh along one direction, the CONTROL_POINTS card is sufficient with just two pairs of (node ID, coordinate). For example, (1, 0.0) and (11, 1.0) would place the first node at coordinate 0.0 and the 11th node at coordinate 1.0. Later the *ALE_STRUCTURED_MESH card will use that information to generate 10 evenly spaced elements along that direction.

For extreme cases where the nodes are randomly distributed, N+1 pairs of (node ID, coordinate) are needed to specify the mesh spacing for N elements along one direction.

*ALE_STRUCTURED_MESH card needs element spacing information along all three local coordinate axes. If none of the geometry definitions is the same, we need three different CONTROL_POINTS cards. Otherwise we can simply reuse the existing CONTROL_POINTS card id.

***ALE_STRUCTURED_MESH**

* ALE_STRUCTURED_MESH					
MSHID	PID	NBID	EBID		
CPIDX	CPIDY	CPIDZ	NID0	LCSID	

In the above, MSHID is the ID of the generated structured mesh. PID is the “mesh part” ID for generated elements. NBID and EBID are the starting IDs of the structured mesh nodes and elements. To avoid conflict, we request NBID and EBID to be larger than the existing largest node ID and element ID, respectively.

CPIDX, CPIDY and CPIDZ are the *ALE_STRUCTURED_MESH_CONTROL_POINTS card IDs along the local x, y and z direction. These CPIDX, CPIDY, CPIDZ values can be the same if the mesh geometry is the same along those directions.

NID0 and LCSID are used to define the mesh origin and local coordinate system. NID0 is the node ID of the mesh origin. We can describe the translational motion for the structured mesh by applying prescribed motion to NID0 through *BOUNDARY_PRESCRIBED_MOTION. LCSID is the ID of *DEFINE_COORDINATE card. It is to specify the local coordinate system the structured mesh is built on.

We can have a rotating S-ALE mesh by using *DEFINE_COORDINATE_NODES with IFLAG=1 option. *DEFINE_COORDINATE_NODES card defines a local coordinate system through 3 user-defined nodes. With prescribed motion applied on those three nodes, the local coordinate then rotates. So does the structured mesh.

***ALE_STRUCTURED_MESH_REFINE**

*ALE_STRUCTURED_MESH_REFINE					
MSHID	NX	NY	NZ		

MSHID is the structured mesh ID. NX, NY and NZ are integers. With the presence of this card, the mesh generated by *ALE_STRUCTURED_MESH card will be refined in such a way that each original element now becomes a NX x NY x NZ evenly spaced elements.

Say a mesh with MSHID 1 was generated by *ALE_STRUCTURED_MESH card that contains 20x10x5 elements. The following card will divide each element in the base model to 5x4x3 elements and build a refined model with 100x40x15 elements.

*ALE_STRUCTURED_MESH_REFINE					
MSHID	NX	NY	NZ		
1	5	4	3		

The purpose of this card is to provide users with a convenient way of model buildup and testing. A user can start to build a smaller model with coarser mesh. Then test the validity of the model by performing several quick runs to check the results. After having enough confidence on the model quality, he can submit the real run with a much finer mesh by simply adding this one keyword card. We believe this coarser to finer mesh approach can save a lot of frustrations in modeling large ALE problems.

This keyword card automatically modifies node/segment/solid sets defined in the base model. These node/segment/solid sets need to be defined by SALEFAC and SALECPT options in *SET_NODE/SEGMENT/SOLID_GENERAL which are described below. Users do not need to regenerate those entities for the refined mesh.

NEW options in *SET_NODE/SEGMENT/SOLID_GENERAL

There are two new options added in *SET_?_GENERAL cards for easier node/segment/solid set generations. As the mesh is generated during the input phase of program execution, we are unable to list the user IDs of these entities explicitly. We provided the following two options for users to easily construct the node/segment/solid sets.

SALEFAC stands for S-ALE face. This option is to select all the nodes/segments/solid elements at one or more structured ALE mesh faces. Setting one or more (-X,+X,-Y,+Y,-Z,+Z) fields to 1 means to include all entities at that face/ those faces.

SALEFAC	MSHID	-X	+X	-Y	+Y	-Z	+Z

Say we have a mesh with MSHID=101 and we want to set a node set including all the surface nodes and later to apply SPC on it. We can do that by adding the following entry under *SET_NODE_GENERAL keyword.

SALEFAC	101	1	1	1	1	1	1

The following assigns all elements at the –X face to a solid set in *SET_SOLID_GENERAL.

SALEFAC	101	1	0	0	0	0	0

SALECPT stands for S-ALE control points. This option is to select all the nodes/segments/solid elements in a box inside the structured ALE mesh. The box is denoted by the node IDs used in *ALE_STRUCTURED_MESH_CONTROL_POINTS card.

SALECPT	MSHID	XMIN	XMAX	YMIN	YMAX	ZMIN	ZMAX

Say we have a mesh with MSHID=101 containing 20x10x5 elements. It has 21x11x6 nodes. To include all these nodes in a node set, the following card is used.

SALECPT	101	1	21	1	11	1	6

To pick the right half of elements and add them into a solid set:

SALECPT	101	11	21	1	11	1	6

We can have multiple SALEFAC and SALECPT cards. They can also be used in a mixed manner. But we do not allow these two options to be used with other options in *SET_?_GENERAL cards.

It is worth to mention that the node IDs are of the "base" model when used with *ALE_STRUCTURED_MESH_REFINE card. In the above example, if later we decide to apply a 2x2x2 refinement on the base model containing 20x10x5 mesh, there is no need to modify the SALECPT card. For example, the following card will always pick all the nodes at –X face in both the base model and the refined model.

SALECPT	101	1	1	1	11	1	6

Model Setup

While S-ALE inherits most of the ALE keywords, the model setup is not quite the same. There are two major differences.

First, part definition in general ALE solver can have multiple roles. Sometimes it represents mesh; sometimes material; sometimes both mesh and material. This caused a certain amount of confusion before.

We try to avoid this by categorizing the part definition in S-ALE into two -- "mesh part" and "material part". "Part" can either mean mesh, or material, but not both. "Mesh part" represents a collection of elements and nodes and contains no material definition. "Material part" contains material definition which is a set of *MAT_ + *EOS_ + *HOURGLASS but has not elements.

Please note, as "mesh part" has no material information, there is no need to define a *PART card for it. Rather, we simply put its ID at PID field in the *ALE_STRUCTURED_MESH card. And then all the elements generated, subsequently, would bear a PID of that "mesh part".

For each "material part", on the other hand, we need to define a separate *PART card to hold its material information and assign it with a unique PID. Later this PID will be listed in *ALE_MULTI-MATERIAL card. "Material part" serves as a wrapper to hold MAT ID+ EOS ID + HG ID. Its whole purpose is to provide material information to *ALE_MULTI-MATERIAL card. Any other appearance of the "material part" PID in the input deck is erroneous.

Secondly, we strictly enforce a one-to-one relationship between the so-called "material parts" and ALE multi-material groups (AMMGs). Each "material part" corresponds to 1 AMMG and vice versa.

Now conceptually the model setup is more intuitive and straightforward. First we construct a mesh and assign a PID to it. This is the "mesh part" and we denote this PID as MSHPID. Next, for each AMMG, we define a "material part" and assign it with a MATPID. Then we list those MATPID, one entry for each AMMG, in the *ALE_MULTI_MATERIAL card. Now we have an ALE mesh and the material definitions of those fluids flowing in it. The final step would be to fill the fluids into the mesh. It is done by constructing the initial volume fractions by using *INITIAL_VOLUME_FRACTION_GEOMETRY card.

In rest of this section, we will use a simplified underwater explosion model to show the construction of a S-ALE input deck following these three steps. We only build the S-ALE keywords below, assuming that all other general keywords are already in place.

Step 1: S-ALE Mesh Generation

1. Set up the following three * ALE_STRUCTURED_MESH_CONTROL_POINTS cards to provide spacing info along the three local axes.

* ALE_STRUCTURED_MESH_CONTROL_POINTS					
3001					
1		-12.0			
25		12.0			

There are 25 nodes along x-direction from -12.0 to 12.0; 24 elements evenly spaced.

* ALE_STRUCTURED_MESH_CONTROL_POINTS					
3002					
1		0.0			
13		12.0			

There are 13 nodes along y-direction from 0.0 to 12.0; 12 elements evenly spaced.

* ALE_STRUCTURED_MESH_CONTROL_POINTS					
3003					
1		0.0			
17		16.0			

There are 17 nodes along z-direction from 0.0 to 16.0; 16 elements evenly spaced.

2. Generate Mesh.

* ALE_STRUCTURED_MESH					
MSHID	PID	NBID	EBID		
1	9	200001	200001		
CPIDX	CPIDY	CPIDZ	NID0	LCSID	
3001	3002	3003	199997	890	

The PID is 9; starting NODE ID and ELEMENT ID are both 200001; CONTROL_POINTS ID along three axes are 3001, 3002 and 3003; the mesh origin is at node 199997; location coordinate system ID is 890 and defined as follows.

* DEFINE_COORDINATE_NODES					
LCSID	NID1	NID2	NID3	IFLAG	
890	199998	199999	200000		

Here, IFLAG=0 means the local coordinate system is set up at the initial time step and won't be updated during the run.

Step 2: Define S-ALE multi-materials

1. Define Material Parts: We have the following three materials in S-ALE mesh, high explosive, water below structure and water above structure.

* PART					
Water below plate					
PID	SECID	MID	EOSID	HGID	
10	10	10	10	10	

*SECTION_SOLID					
SECID	FORM				
10	11				

*MAT_NULL					
MID	RO	PC	MU		
10	1.00				

*EOS_GRUNISEN					
EOSID	C	S1	S2	S3	GAMA0
10	0.148	1.75	0.0	0.0	0.28

*HOURGLASS					
HGID	FORM	FACTOR			
10	1	1.0E-6			

The other two "material parts" are defined below with *MAT and *EOS cards skipped.

* PART					
High explosive					
PID	SECID	MID	EOSID	HGID	
11	10	11	11	10	

* PART
Water above plate

PID	SECID	MID	EOSID	HGID	
12	10	12	12	10	

2. List all S-ALE “material parts” in *ALE_MULTI-MATERIAL_GROUP card

* ALE_MULTI-MATERIAL_GROUP					
PID	TYPE				
10	1				
11	1				
12	1				

In the above card, PART 10 represents material “water below”, marked as AMMG #1; PART 11 “high explosive”, AMMG #2 and PART 12 “water above”, AMMG #3. Note here all those PARTs are only to provide material definitions; no mesh info at all.

Step 3: Define the initial configuration of all ALE multi-materials

In theory, any given element inside the S-ALE mesh could be fully or partially occupied by any ALE multi-material(s) defined. Naturally comes the question how we record such information. We introduce an additional element history variable called “volume fraction”. We calculate volume fractions for each AMMG in each ALE element and record them accordingly. Later we use these volume fractions to reconstruct the material interfaces inside each element.
So, it is required for users to provide the initial values of those volume fractions; same as users need to provide values for initial stresses. We use the following command to generate those values.

*INITIAL_VOLUME_FRACTION_GEOMETRY					
SID	IDTYP	BAMMG	NTRACE		
9	1	1			
TYPE	FILLOPT	FAMMG			
6		2			
X0	Y0	Z0	R		
0.0	0.0	0.0	2.0		
TYPE	FILLOPT	FAMMG			
3		3			
X0	Y0	Z0	XCOS	YCOS	ZCOS
0.0	0.0	12.0	0.0	0.0	15.0

Here we first fill the MESH PART 9 completely with AMMG 1 (water below); then we switch the water to AMMG 2 (High explosive) inside a sphere with radius 2.0 and centered at (0.0, 0.0, 0.0); at last we switch AMMG 1 (water under) to AMMG 3 (water above) above a plate located at (0.0, 0.0, 12.0).

Miscellaneous

1. *CONTROL_ALE and *INITIAL_DETONATION

* CONTROL_ALE					
DCT	NADV	METH	AFAC		
		1	-1		
START	END	AAFAC	PRIT		

The only option users need to care is METHOD while using the S-ALE solver. Currently, S-ALE supports two advection methods: Donor Cell (1st order) and Van Leer (2nd order). AFAC=-1 is mandatory for using multi-material ALE formulations. It means mesh smoothing is turned off.

* INITIAL_DETONATION					
PID	X	Y	Z	LT	
9	0.0	0.0	0.0		

*INITIAL_DETONATION is to set up the detonation center. Please note the PID here needs to refer to the "mesh part" PART. As we know that "material parts" do not contain any elements.

2. Boundary Conditions

As for the boundary condition, it is pretty much the same as in a typical Lagrange or ALE problem. *BOUNDARY_SPC to apply fixed or symmetric boundaries; *BOUNDARY_PRESCRIBED_MOTION to apply velocity boundaries; *LOAD_SEGMENT to apply pressure boundaries.

First we define a node set consisting of all the nodes at –Y face by using the SALEFAC option in *SET_NODE_GENERAL.

*SET_NODE_GENERAL							
SID							
1							
OPTION	MSHID	-X	+X	-Y	+Y	-Z	+Z
SALEFAC	1			1			

Next we lock Y direction at all nodes in NODE SET 1. Now we have the intended symmetric condition.

* BOUNDARY_SPC_SET					
ID	LCSID	X	Y	Z	
1		0	1	0	

3. Fluid Structure Interaction (FSI)

S-ALE uses exactly the same FSI algorithm and keyword ALE solver uses, with only one tiny modification to the MCOUP field. Now MCOUP can be a PART SET which includes all the AMMGs coupling to the structure. Here PARTSET 1011 contains "material part" 10 (water below, AMMG1) and 11(high explosive, AMMG 2). This way, we can avoid the unnecessary *SET_MULTI-MATERIAL_GROUP_LIST card.

* CONSTRAINED_LAGRANGE_IN_SOLID							
SLAVE	MASTER	SSTYP	MSTYP	NQUAD	CTYPE	DIREC	MCOUP
1	9	0	1	2	4	1	1011
START	END	PFAC			NORM		

Now we have finished the model setup. The example input deck setup here can be downloaded from http://ftp.lstc.com/anonymous/outgoing/hao/sale/models/sale.tar.gz.

Numerical Results

In this section, we will illustrate the S-ALE solver's performance by using several numerical examples. All the numerical examples discussed here can be downloaded from the http://ftp.lstc.com/anonymous/hao/sale/models/ directory. So we won't be listing all details but rather giving a brief description of the model.

First, we have a small model to simulate a simplified airbag inflated by pressurized ideal gas. It has a structured ALE mesh of 9261 elements with a grid of 21x21x21. Airbag is modeled by 294 FORM 16 shell elements.

SMP performance

To study the SMP scalability of the S-ALE solver, we ran the airbag problem with both "Donor Cell" and "Van Leer" advection scheme. The former one is a first order algorithm while the latter second order. Please note that S-ALE supports only SMP with consistency flag on. This means all SMP results would be digit-to-digit consistent. When execution, NCPU= -N command line option needs to be used. The following two tables show the raw data of running time spent in S-ALE solver with respect to the number of SMP threads used.

Donor Cell:

NCPU	-1	-2	-4	-8
Time (s)	298	168	103	70
(Speed up)		1.77	2.87	4.25

Van Leer:

NCPU	-1	-2	-4	-8
Time (s)	531	292	179	111
(Speed up)		1.82	2.97	4.78

We believe the 4.78 times of speedup achieved with 8 SMP threads is satisfactory for a first version SMP implementation. We will keep exploring ways of further improvements.

MPP Hybird performance

We also tested the efficiency of MPP hybrid runs. We ran both Donor cell and Van Leer cases with four different setups, 24x-1, 12x-2, 8x-3 and 6x-4. 24x-1 means we ran 24 MPP processes and each MPP process has 1 SMP thread. 12x-2 means 12 MPP processes and each MPP has 2 SMP threads. Again, the running time spent in S-ALE solver is given in the below two tables.

Donor Cell:

NCPU	24x-1	12x-2	8x-3	6x-4
Time (s)	55	57	60	60
(Speed up)	9.72	9.32	8.8	8.8

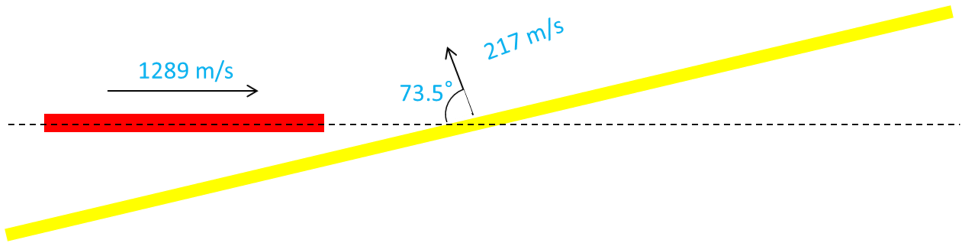
Van Leer:

NCPU	24x-1	12x-2	8x-3	6x-4
Time (s)	90	94	99	98
(Speed up)	11.05	11.68	10.73	10.16

Generally, MPP performs much better than SMP. So consequently by increasing the number of SMP threads per MPP process we should see a deteriorating performance. As we observed only a little longer running time from the above, it looks like that S-ALE still achieved a pretty decent MPP hybrid performance.

MPP performance

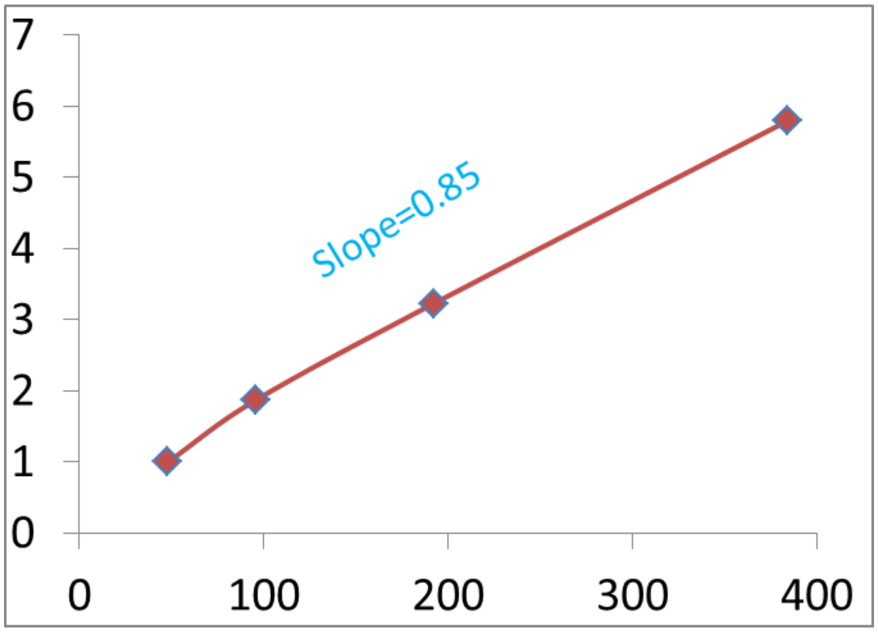
Next we are going to use a much larger model to show the new solver's MPP performance. It is a long rod projectile impacting an oblique steel plate (Fugelso & Taylor 1978) depicted in the following sketch. This model is a pure ALE model and has 3 multi-materials. They are rod, plate and vacuum which filled all the empty space. It contains no Lagrange parts.



We first constructed a base model consisting of 215x60x30 ALE elements to test the validity of the model setup. It only took 14 minutes to complete the run using a single thread SMP executable. Once we were sure the model was set up correctly we added the *ALE_STRUCTURED_MESH_REFINE card to refine the model 5 times along each direction to get a 50 million elements model.

The maximum statically allocated memory required to run this model was 1.4G words, right at the beginning of the MPP decomposition phase. So we were able to run this 50-million model using a single precision executable. We ran the same model with different number of cores from 48 to 384 to test the MPP scalability. Runtime varied from 9.7 hours with 48 cores to 1.67 hours with 384 cores. The results are shown in the following table and scalability graph.

NCPU	48	96	192	384
Total Time (s)	34966	18624	10837	6032
(speedup)		1.88	3.23	5.80
S-ALE time (s)	17323	9228	5320	2616
(speedup)		1.88	3.26	6.62



In the above graph, x axis is number of cores used and y axis is the speedup achieved. The scalability curve remains linear until 384 cores and the slope is around 0.85. This indicates an excellent MPP efficiency for the S-ALE solver.

Conclusion

This paper presents the new Structured ALE solver added in LS-DYNA R9 release. It targets to solve large ALE problems using structured mesh. The S-ALE solver uses the same theory as in the general ALE solver. But it is with a new implementation taking advantages of structured mesh. It supports SMP, MPP and MPP hybrid with an excellent scalability. The memory usage was kept at minimum to accommodate large problems. The solver is still at early stage of its life cycle and new features will be added in the coming years. We hope it can assist our users to solve the ALE problems faster, more efficiently and smoothly.