An Introduction to *CONSTRAINED_BEAM_IN_SOLID

Hao Chen

Livermore Software Technology Corp

Background

Rebar reinforced concrete is commonly used in construction industries. Its mechanical properties are of interest to people working in various engineering fields. While experimental, theoretical studies provided us essential guidelines to utilize this material efficiently and effectively, numerical simulations also showed their usefulness in predicting the overall structure behavior.

There are different techniques to simulate rebar reinforced concrete. One is to construct an inhomogeneous material model in which the concrete and the rebars inside were treated as a whole. This way, there is not explicit modeling of rebars. Instead they are assumed to be aligned along some specific directions inside the concrete solid elements.

Another way is to discretize rebars as beams and concrete as solids and make them share the same sets of nodes. Of course this requires extra efforts in mesh generations. It is not always doable if not cumbersome enough.

So an alternative technique becomes appealing to our users. It is to apply constraints between two set of nodes. One is for beams and another for solids. This way we avoid the meshing difficulties in "shared nodes" technique. Also, we don't need to construct complicated material models with the "composite material" approach.

Motivation

The rebar-concrete constraint coupling was done through a legacy keyword called *CONSTRAINED_LAGRANGE_IN_SOLID. This keyword is shared by two totally different applications. CTYPE=2 is used to model rebar coupling while CTYPE=4/5 is used to perform ALE fluid structure interactions. We will refer its rebar coupling function as "CLIS CTYPE 2" in the discussion below.

The CLIS CTYPE 2 had been widely used and proved being quite helpful in solving our users' problems. However, there are several flaws and shortcomings found by both our users and the author. Efforts were made to fix and enhance this function. But later the author found it was not possible to solve the fundamental error without overhauling it coding structure. He also found its implementation made it is very hard to add in new features requested by users.

In early 2015, the author started to develop *CONSTRAINED_BEAM_IN_SOLID to perform rebar constraint coupling. A new keyword was introduced for two reasons. First, the fix for CLIS CTYPE 2 was designed for beam only. However the "slave" in the legacy CLIS could also be other Lagrange entities such as node set, segment set and parts other than beams. The second reason is to be user-friendly. Too many times, the author witnessed users' confusion caused by the dual functionalities of the CLIS card. Also it has become a heavy, lengthy one with way too many flags so that even the most experienced user would frequently make input mistakes. So the author thought it would be the best to separate these two functionalities by giving rebar coupling a new, dedicated keyword contains minimum input fields. This way, the author could also secure the input fields needed for new features.

Constraint coupling

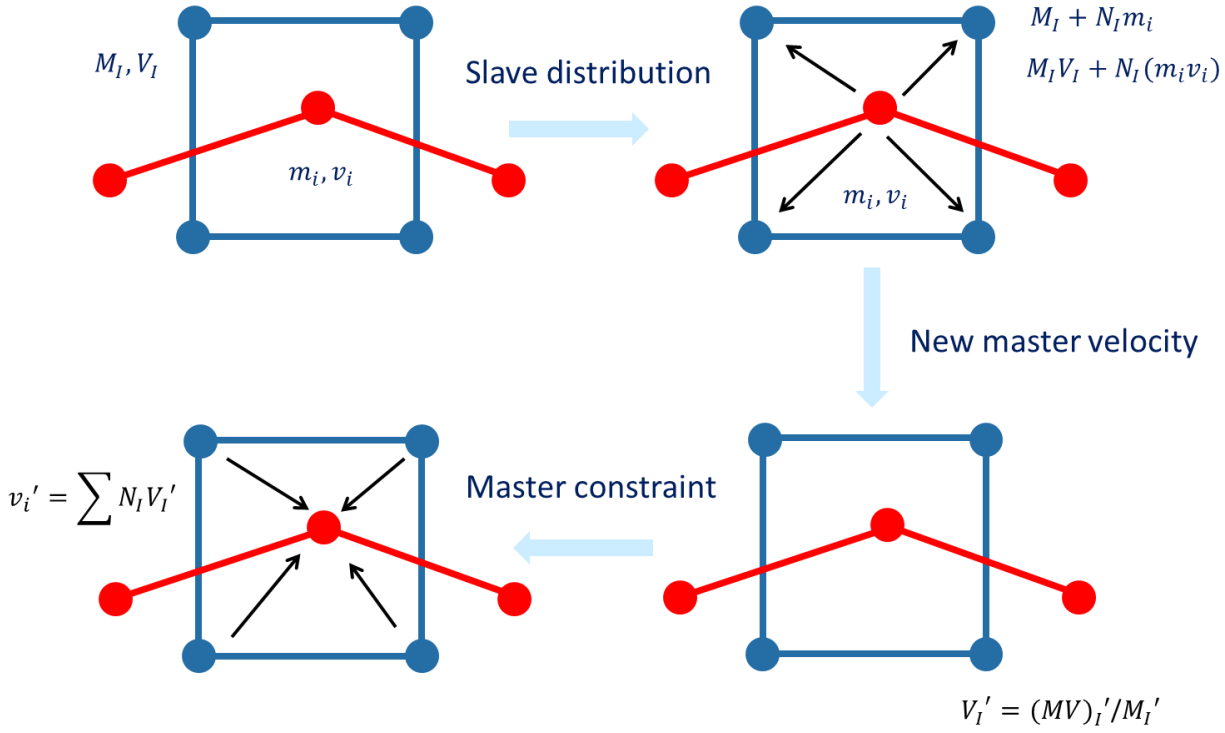
Rebars and concrete are modeled by beams and solids, respectively. Beam mesh is submerged in solid mesh. Each of them has its own independent motion. Without some kind of coupling algorithm, they will move freely as if the other one doesn't exist at all. The way we couple these two is called "constraint method". There are always two parties involved in a constraint coupling. One is "master" and the other "slave". The slave contributes to the master and the master constraints the salve.

Typically both velocity and acceleration need to be constraint. The first is to ensure momentum conservation and the second force balance. The algorithm is exactly the same for both velocity and acceleration. For simplicity, we will limit the discussion below to velocity only.

We start with an incompatible velocity field. Beam nodes are slave and denoted by lower case characters; solid nodes are master and upper case ones.

1. The first step is for slave beam nodes to distribute their nodal mass and momentum to master solid nodes.
2. Next we update the master nodal velocity by dividing the new momentum by new mass.
3. Finally we assign the interpolated velocity back to slave nodes.

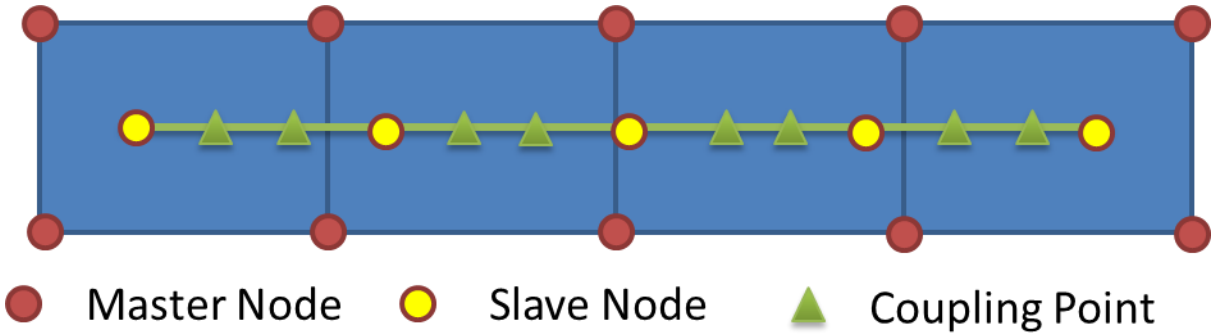
Now we have slave nodes moving exactly the same way as master nodes. The process is shown in the figure below.



Problems

The theory is straightforward. However, in real cases, the beam nodes are not always placed that well so that all solid elements contain at least one beam node. If a beam crossed certain solid element but its nodes did not fall in that solid, this solid won't get any distribution from this beam and the algorithm simply would fail.

So in both CLIS CTYPE 2 and CBIS we have an option to put extra "coupling points" in between the two end nodes of a beam element. This way, solid nodes get distributions either from beam nodes or these coupling points. This field is referred as "NQUAD" in CLIS or "NCOUP" in CBIS.

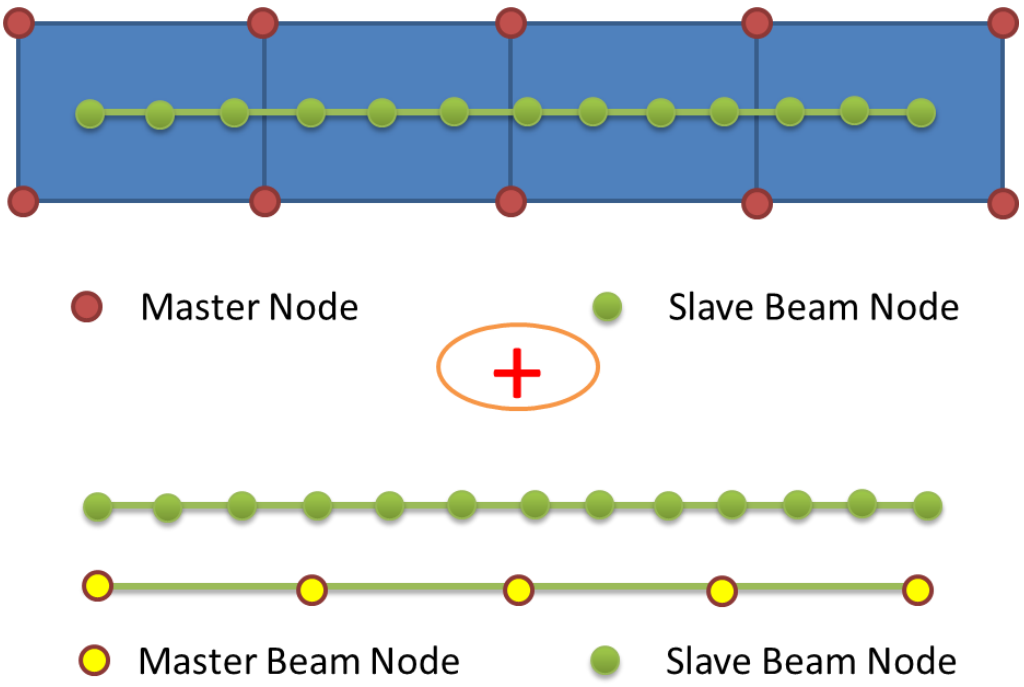


Now comes the puzzle, we all know each beam node has its nodal mass. This mass comes out naturally from discretization. Also it has its nodal velocity. These two entities are "physical". But for these artificially generated coupling points, there are no such properties. For velocity, it is pretty straightforward. We simply assume the velocity at a coupling point should be interpolated from the beam end nodes. How about mass?

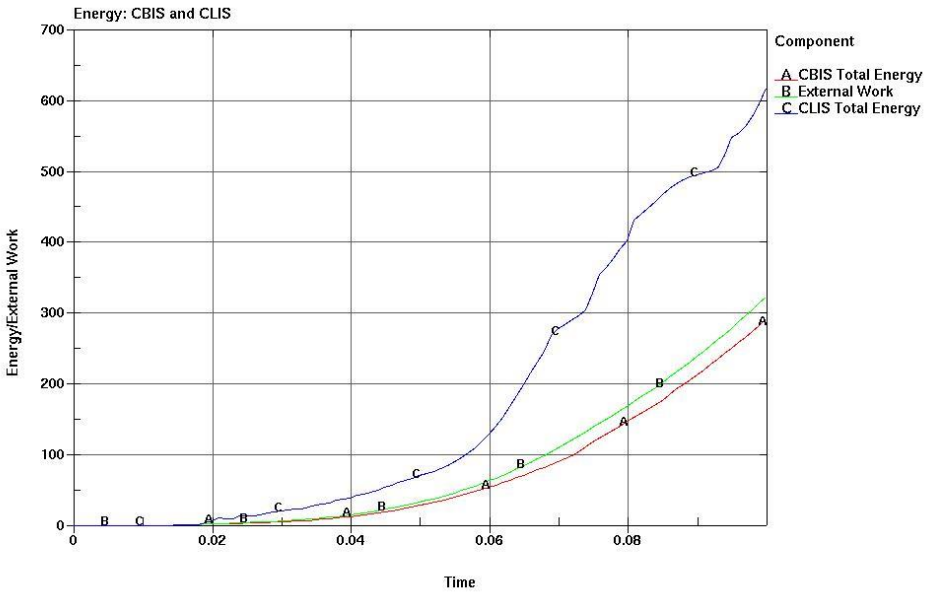
Unfortunately CLIS CTYPE 2 did not do it right. It moved half the beam element mass from nodes to these coupling points. This approach is rather arbitrary and lacks of theoretical basis. Another mistake it made was in the constraint process. The velocity was mapped only from master nodes to beam nodes, not to coupling points. So the overall process is not complete. These two errors won't reveal themselves if we only look at the structure motion. But when we checked the energy plot, we saw a spurious large internal energy increase.

Bridging Coupling

So how should we address this problem? The author came up with an idea which he called "bridging coupling". As the beam mesh is too coarse to be directly coupled to the solid elements, a "slave beam" is constructed in between to couple to both "master beam" and "master solid". Now we have two couplings. The first is between the "slave beam" and the "master beam"; the second between the "slave beam" and the solid mesh. "Slave beam" serves as a "bridge" connecting the real beam and solid elements. The concept is shown in the figure below.



We could see from the following figure that with the new CBIS implementation, the previously shown spurious energy increase disappeared. This mysterious spurious energy increase had puzzled both the author and our users for quite some time.



The mass at coupling point now simply takes the value of the "slave beam" nodal mass. It has a clear physical meaning and is theoretically correct. The coupling point has its velocity constrained by solid nodes during the "mapped-back" stage. And it then distributes the corresponding momentum to the "master beam" nodes.

The author does not intend to bother the readers with too many details. The idea of bridging coupling is conceptually simple and straightforward. However its implementation has been through some difficulties. It underwent several trial and error loops. Our users tested it extensively and provided valuable feedbacks.

Bucket sorting and searching

There were also some other improvements in CBIS. One deserves some explanation here. CBIS has an enhanced, independent bucket sorting and searching routine. That is in contrast to CLIS CTYPE 2, which shares these routines with ALE FSI coupling.

There are two advantages for CBIS to have its own sorting and searching routines. First, only solid elements belong to the "master" concrete are included in the bucket sorting. ALE elements won't be included. This brings memory reduction and a more efficient execution.

Secondly, as most ALE elements are hexahedron, searching subroutine, for efficiency, doesn't contain a separate treatment for tetrahedron and pentahedron. Rather they are treated as degenerated hexahedron. While in ALE FSI case it won't have too much difference, it is not acceptable for unstructured mesh used in rebar coupling. The new CBIS subroutine contains the enhanced algorithm to treat these three different solid elements separately.

Conclusion

By introducing the new *CONSTRAINED_BEAM_IN_SOLID keyword, we successfully fixed problems in the legacy *CONSTRAINED_LAGRANGE_IN_SOLID CTYPE 2. By separating this functionality from ALE FSI, we achieved both a clean keyword card and a clean code base for future development.

The author wants to express his gratitude towards our users for their effort testing this new keyword and valuable feedbacks. And hopes it could be of help to our users in solving their challenging problems.