Enhancements to Implicit Mechanics

R. Grimes, C. Weisbecker, C. Ashcraft, E. Vecharynski, F.H. Rouet,, J. Anton, L. Li, and R. Lucas

Livermore Software Technology

1 Overview

LSTC is continually enhancing the capabilities, features, and performance of Implicit Mechanics. This article will focus on the new features that have been developed for the LSDYNA User Community. We have added the ability to compute the response to multiple loads for Implicit Linear. We have added the ability to compute Residual Vectors. We have added the automatic feature of identifying when it is possible to reuse the symbolic factorization from the previous matrix factorization to improve efficiency. We have migrated the storage management for the linear algebra from static memory to dynamic memory to make usage easier and improve efficiency of the management of memory. We will conclude with a quick overview of our ongoing activities in new development.

2 Implicit Linear Multiple Loads

A common analysis approach is to compute the response of a model to a given force. Many times the user is interested in the linear response to many forces. To accommodate this usage we have added the ability to do multiple implicit time steps in linear mode but reset the geometry to the initial state at the start of each time step. Previously multiple time steps in linear analysis would accumulate the deformation from the previous time steps. For this category of analysis this would pollute the results.

This feature is activated using NSOLVR=-1 on *CONTROL_IMPLICIT_SOLUTION. NSOLVR=+1 is the multistep linear analysis where the deformations are accumulated over the previous time steps. With NSOLVR=-1 the geometry is reset at the start of each time step. The other key to using this new feature is specifying the loads for each time step. You can use any combination of *LOAD keywords which are driven by load curves. The load curve for a particular load should be zero everywhere except the time step used for that particular load. An example for 3 loads applied in the x, y, and z direction at node 4101 is

```
*LOAD_NODE_POINT
$$ Each of the load curves have 2 points (0,0) and (1,1), but each curve
$$  has a different abscissa offset value.
$$ Thus for each step in the analysis, only one curve has a load value of 1.0 and the
$$   rest have a load value of 0.0.
$$
$#     nid      dof     lcid       sf      cid       m1       m2       m3
      4101        1    41011      1.0        0        0        0        0
      4101        2    41012      1.0        0        0        0        0
      4101        3    41013      1.0        0        0        0        0
*DEFINE_CURVE
$#    lcid     sidr      sfa      sfo     offa     offo   dattyp    lcint
     41011        0      0.0      0.0      0.0      0.0        0        0
$#              a1                 o1
               0.0                0.0
               1.0                1.0
*DEFINE_CURVE
$#    lcid     sidr      sfa      sfo     offa     offo   dattyp    lcint
     41012        0      0.0      0.0      1.0      0.0        0        0
$#              a1                 o1
               0.0                0.0
               1.0                1.0
*DEFINE_CURVE
$#    lcid     sidr      sfa      sfo     offa     offo   dattyp    lcint
     41013        0      0.0      0.0      2.0      0.0        0        0
$#              a1                 o1
```

```
            0.0             0.0
            1.0             1.0
```

The user will need to set the termination time to 3.0, the implicit time step to 1.0, and to insure that the auto time stepping is not active. Please contact LSTC Technical Support for a sample test case. This analysis only performs one matrix formation and matrix factorization. The *CASE statement approach is not as efficient because the assumption is the model can change from case to case. So each case requires a new matrix reformation and a new matrix factorization. Since matrix factorization is the expensive part of solving the system this new approach is more efficient.

Figures 1 shows a model, a vehicle frame, that has 4 points the user wants to apply loads in each of the 3 global directions. Figure 2 shows the Von Mises stresses for the load applied to the x direction at the 3rd node of interest.

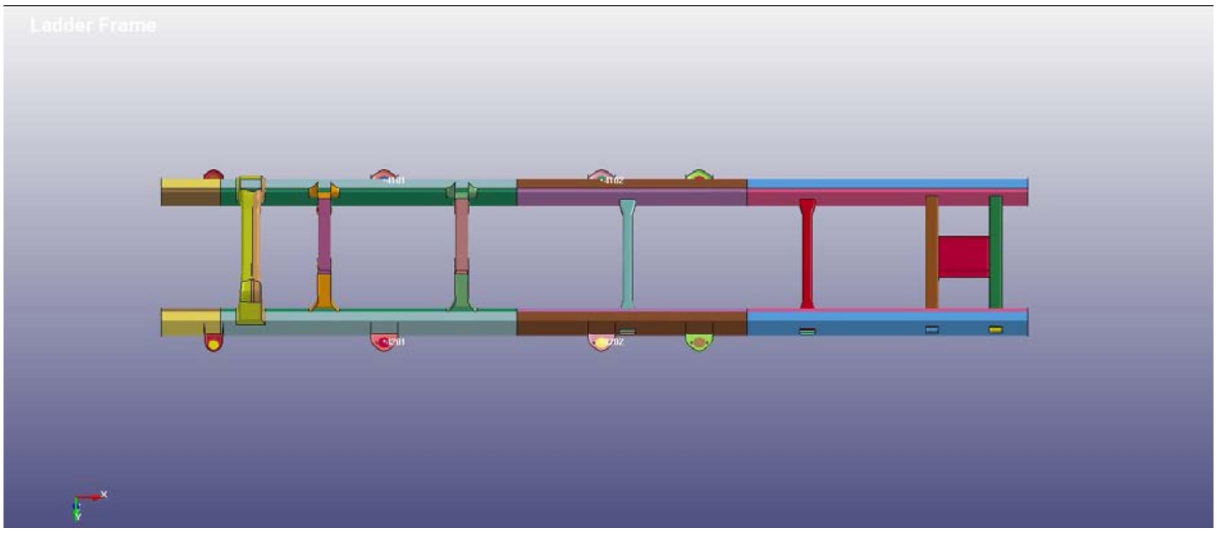

Fig.1: Implicit Linear Multistep model

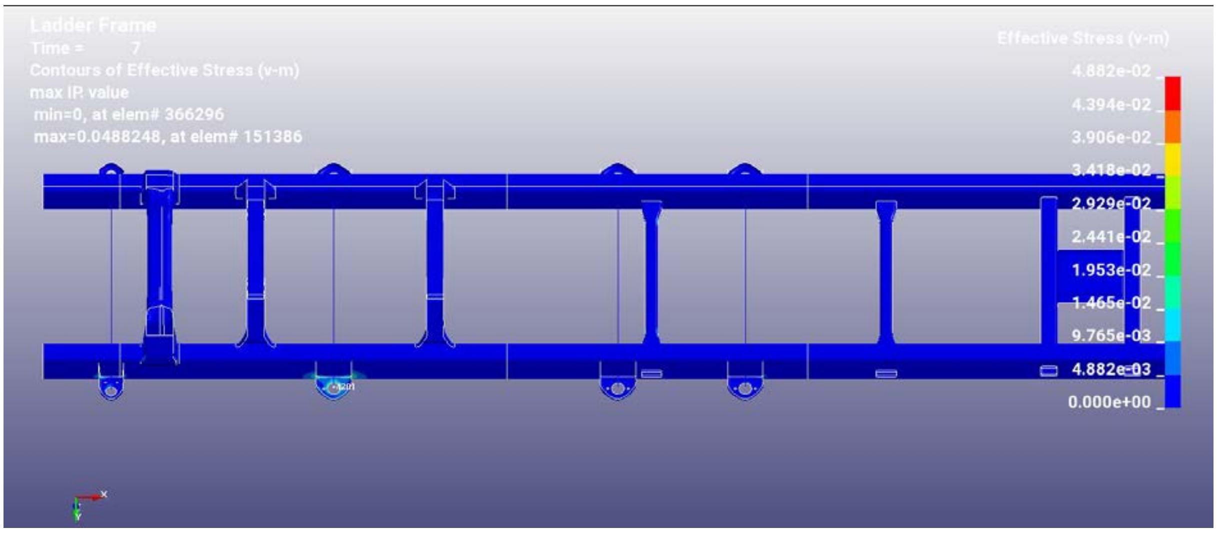

Fig.2: Stresses of a linear response in the x direction at the 3rd node of interest.

.

3 Residual Vectors

Residual vectors are used in Frequency Response Analysis to better capture the response to a given force that is not well represented by the basis of the linearized model. To compute the residual vector one computes the response to a given force and then orthogonalizes that response to the modes used in the linearization. All of the requests for this feature have all been for the force specification to be a unit force in a given set of directions for a given set of nodes. And the linearized models were based on eigenmodes. LSDYNA already has the feature of computing constraint, attachment, and eigen modes for a given model using *CONTROL_IMPLICIT_MODES. An attachment mode is the response of the model to a unit force for a specific direction at a specific node. To compute residual vectors all we needed to do was to add an option to orthogonalize the attachment modes to the eigenmodes. This new feature is implemented by putting a “1” in the first field of the (optional) 3rd line for *CONTROL_IMPLICIT_MODES. Then all attachment modes computed as specified by the attachment mode node set NSIDA (2nd field of the first line) will be orthogonalized to the eigenmodes computed using the specification of NEIG (3rd field of the first line).

The feature of computing residual vectors can be easily extended to allow more general load specifications and to use precomputed eigenmodes. This will be done if requested by users.

4 Reuse of Symbolic Factorization

LSDYNA uses direct solution for the linear systems that arise in Implicit Mechanics. Although we have some iterative methods they tend to work only for special problems. One of the costs for the direct solution is the symbolic factorization phase which computes the reordering of the rows and columns of the stiffness matrix to reduce the computational resources for computing the numerical factorization. For serial and SMP applications this cost is trivial. But for large models using MPP the numeric factorization can be faster than the symbolic factorization. LSTC implemented a feature to automatically determine if the symbolic factorization from the previous numeric factorization can be reused for the current factorization. If the structure of the matrix has not changed in can be reused. This is a common occurrence for many implicit applications.

5 Memory Management

LSTC is constantly modernizing LSDYNA to improve ease of use and efficiency. In addition to improvements in the direct solution technology, which is a separate presentation, we are migrating the memory management for the linear algebra from static memory specification to dynamic memory allocation. This improves ease of use and allows all parts of the linear algebra to share resources. Static memory is that memory allocated based on the memory= and memory2= specifications on the command line when starting an execution of LSDYNA. The matrix assembly package and default reordering software, because they are written in C, already were using dynamic memory. Converting the direct solution software to dynamic memory also efficient sharing of the memory resource. It also simplifies the user specifications for memory and memory2. With the development version of LSDYNA (post R10) the user needs to look at the d3hsp or messag or mes0000 file to find

```
 expanding   memory to    713015493 implicit matrix assembly allocation
```

In MPP one needs to find the maximum of this value for all mesxxxx files.

In SMP the user needs to specify memory on the command line to something slightly larger. Here I would use memory=715M because I like to round up. In MPP the user also needs to look for

```
 Memory required to process keyword :  1711m
```

For MPP I would then use memory=1711M and memory2=715M. The memory specification is for processor 0 to use for initial processing of the keyword input and performing the domain decomposition. After domain decomposition the memory for P0, as well as all of the other processes, Is reset to the memory2 specification.

All of the linear equation solvers, SMP and MPP, and eigensolvers have been converted to use dynamic memory in versions of LSDYNA after R10. At the time of this conference that means the development version.

6 What We Are Working On

Responding to either user requests or our forecast of requirements LSTC is working on a number of new developments. Of course, we are also evaluating new technology from other sources. For this presentation we will list those associated with eigenvalue computations. The enhancements for direct linear equation solution will be covered elsewhere. We have an ongoing development activity to implement the AMLS eigensolution algorithm for NVH applications. This activity is covered elsewhere in these proceedings. We have also begun looking at newly developed algorithms for eigencomputations such as iterative-based eigensolvers and spectrum slicing. We hope to identify when it is appropriate to use our default approach based on Block Shift and Invert Lanczos or switch to AMLS, or an approach based iterative based eigensolvers and spectrum slicing.

At the request of users whose models have a high degree of symmetry we have started a project to use Sectoral Symmetry for the EIgencomputation. This allows a smaller model to be used, perhaps many many times instead of the full model. Below is a simple example of a disk with a hole

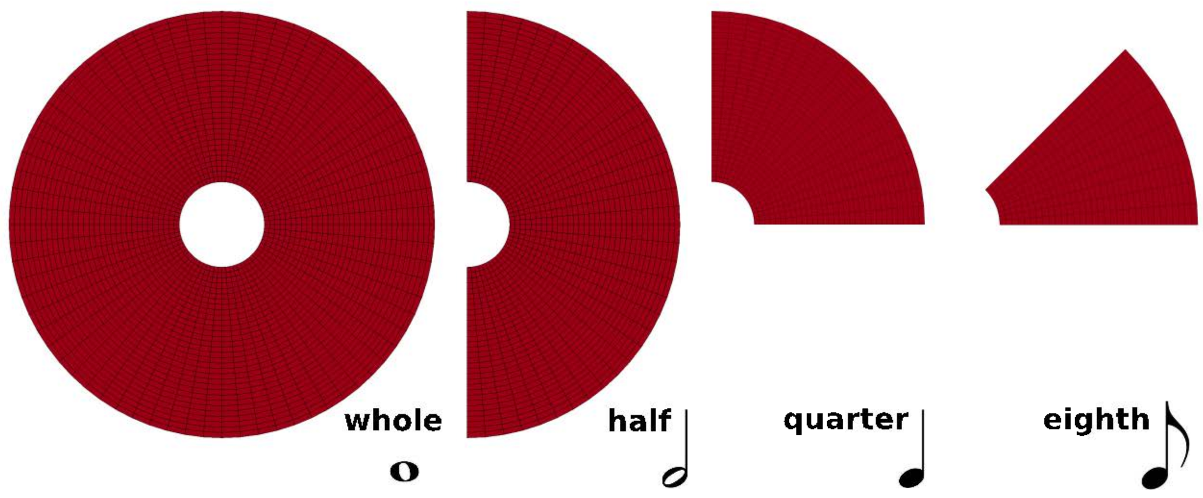

Fig.3: Disk with a hole for Sectoral Symmetry

The user can choose to provide a model of only a small section of the model and perform an eigenvalue computation using just that small section. This is important to users that may have up to 100 sectors whose union forms the entire model. Working on a small model allows them to have a very detailed model of that sector. Such detail for the full model would be intractable.

There is much user interested to compute eigenmodes of an implicit mechanics model that is interacting with a fluid. The fluid is exerting a pressure loading on the mechanics model that changes the vibratory response of the model. Examples are ships in water, dams with water behind them, automobile structures with air flow around them, and even the effect of water on an automobile structure, We have just started to investigate the methods to solve these eigenvalue problems.

7 Summary

This presentation has reviewed the ongoing enhancements to Implicit Mechanics in LSDYNA. We have added features such as Linear Analysis with multiple loads, the computation of residual vectors, and automatically identifying the reuse of the symbolic factorization. We highlighted our efforts to move to dynamic memory for the Linear Algebra and the benefits of ease of use and performance. We concluded with an overview of our ongoing activities in new development, especially in the area of eigencomputations.