Processing of Equality Constraints for Implicit in LS-DYNA v. 970

C. Cleve Ashcraft
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, CA 94550
(925) 449-2500
cleve@lstc.com

Roger G. Grimes
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, CA 94550
(925) 449-2500
grimes@lstc.com

Bradley N. Maker
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, CA 94550
(925) 449-2500
maker@lstc.com

Keywords: Implicit, Constraints

ABSTRACT

LSTC is committed to building an implicit capability into LS-DYNA that is as capable as the flagship explicit capability. Over the years LSTC has added many different types of constraint handling capabilities in explicit that now have to be handled by implicit. The vast majority of these constraints are equality constraints imposed on the linear or nonlinear solution required at each time step. We will describe a new approach for handling equality constraints that has allowed us to robustly process them without placing unnecessary restrictions on how the user poses the constraints.

Our new approach effectively and efficiently processes the constraints for the linear, nonlinear, and eigenvalue problems that have to be solved by the users of Implicit LS-DYNA. We compute a transformation based on the Jacobian matrix of the constraint equations and apply that transformation to form a reduced stiffness and, if necessary, reduced mass matrices. The transformation is also used to transforms vectors from the unconstrained space to the constrained space and back again. The only restriction placed on the structure of the constraint matrix is that it is full rank.

We will also highlight the various constraints now supported for implicit solution in LS-DYNA v. 970 and demonstrate the solution of some problems illustrating these constraint features.

INTRODUCTION

Unlike explicit time integration, the computational costs of implicit time integration methods are dominated by solving the linear system

$$Ku = f$$
$$\textit{subject to} \;\; Cu = d$$

where *K* is the global stiffness matrix, *u* is the vector of incremental displacements, *f* is the vector of forces, *C* is the Jacobian of the constraint equations, and *d* is the right-hand-side of the constraint equations (representing motion from such constraints as prescribed motion and motors). This same linear system appears in static analyses. For eigenvalue problems arising in vibration analyses, the linear algebra problem is

$$K\Phi = M\Phi\Lambda$$
$$\textit{subject to} \quad Cu = 0$$

where Λ and Φ are the eigenvalues and eigenvectors of the system.

It is paramount that these linear algebra problems are solved efficiently and effectively.

THE STANDARD APPROACH TO CONSTRAINT APPLICATION

The key problems we were faced with at LSTC for implementing equality constraint equations for implicit were three-fold. First, we had to incorporate the vast number of equality constraint equations that are supported by the explicit side of the package into implicit. Second, we needed to efficiently apply the constraints to form the reduced stiffness and mass matrices required for the solution of the linear algebra problems. Third, we did not want to put undue restrictions on the users construction of their models.

One standard approach is to “constrain” the constraint equations to have the form

$$u_D + C_{D,I}u_I = d.$$

That is, for each constraint equation a dependent or slave degree of freedom (dof) is given as a function of independent or master dofs. The “constraint” imposed on the modeler is that the dependent dof cannot be used as an independent dof in another constraint equation.

This approach allows a simple and efficient substitution technique to apply the constraint equations to the global stiffness matrix *K* and, if necessary, the mass matrix *M*. This substitution is accomplished by first partitioning *K* into

$$\begin{bmatrix} K_{I,I} & K_{D,I}^T \\ K_{D,I} & K_{D,D} \end{bmatrix} \begin{bmatrix} u_I \\ u_D \end{bmatrix} = \begin{bmatrix} f_I \\ f_D \end{bmatrix}.$$

Substituting $d - C_{D,I}u_I$ for u_D yields

$$\hat{K}_{I,I}u_I = \hat{f}_I$$

where

$$\hat{K}_{I,I} = (K_{I,I} - K_{D,I}^T C_{D,I} - C_{D,I}^T K_{D,I} + C_{D,I}^T K_{D,D} C_{D,I})$$

and

$$\hat{f}_I = (f_I - C_{D,I}^T(f_D - K_{D,D}d) - K_{D,I}^T d).$$

This substitution can be efficiently performed during matrix assembly at either the elemental or global level.

An extension of this approach supported by some analysis codes allows the constraint equations to have the following form

$$U_{D,D}u_D + C_{D,I}u_I = d$$

where $U_{D,D}$ is a block upper triangular matrix and each diagonal block contains rows associated with dofs associated with a single nodal point. The upper triangularity of $U_{D,D}$ allows constraints to be chained together but no cycles of dependency can occur.

The difficulty of either of these two approaches is that they unduly place restrictions on the application of constraints to complicated models. In both cases, it is necessary for the user and/or the analysis package to explicitly specify the dependent dofs.

Some constraints have a natural set of dofs that can be specified as dependent, e.g., a rigid body has its independent dofs found at its center of mass, and all others are dependent. However, linkages of rigid bodies have proven to be problematic, for cases can arise where it is time to apply a constraint equation and all of the dofs at a center of mass have previously been defined as dependent. This is usually caused by an unfortunate choice of dependent dofs for a constraint at the beginning of a cycle of constraints, made without global knowledge of the constraints. Then when the cycle comes back to the start position there are no free dofs left. We have reached the point where the dependent dofs must be chosen *automatically* with a global view of the entire constraint system.

OUR NEW APPROACH

We wanted an approach that would be flexible and efficient to allow us to quickly implement the many equality constraint equations available to users of LS-DYNA. But that approach had to be computationally efficient.

To partition the constraint matrix (and so automatically choose the dof sets D and I) we use a a weighted bipartite graph matching algorithm from combinatorial optimization. This yields a constraint system

$$C_{D,D}u_D + C_{D,I}u_I = d$$

where $C_{D,D}$ is a well conditioned and sparse (frequently diagonal or block diagonal) matrix. Now we can efficiently and robustly compute $C_{D,D}^{-1}$ and apply is the above equation to yield

$$u_D + C_{D,D}^{-1}C_{D,I}u_I = C_{D,D}^{-1}d.$$

or

$$u_D + \hat{C}_{D,I}u_I = \hat{d}.$$

We can then apply this as the substitution in the standard approach to yield the reduced linear algebra problem.

EQUALITY CONSTRAINTS IMPLEMENTED FOR IMPLICIT

We have successfully implemented this new approach for processing equality constraints in v. 970 of LS-DYNA for implicit solution. The following is a list of all such constraints now supported. We cannot call this list complete because it continues to grow as we add additional features in LS-DYNA to support our user community.

*BOUNDARY_CYCLIC
*BOUNDARY_NON_REFLECTING
*BOUNDARY_NON_REFLECTING_2D
*BOUNDARY_PRESCRIBED_MOTION
(all options)
*BOUNDARY_SLIDING_PLANE
*BOUNDARY_SPC
(global and local)

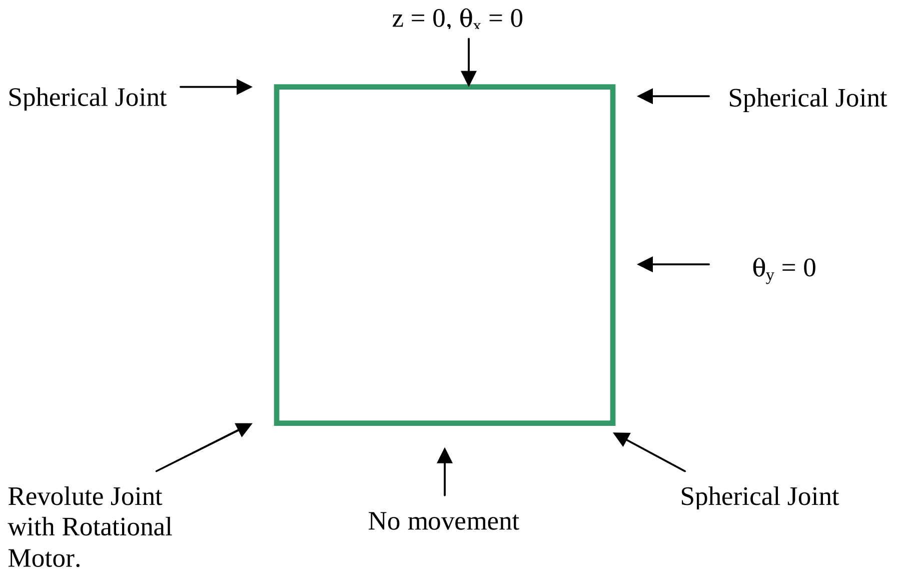
*BOUNDARY_SYMMETRY_FAILURE
*CONSTRAINED_ADAPTIVITY
*CONSTRAINED_EXTRA_NODES
*CONSTRAINED_GLOBAL
*CONSTRAINED_INTERPOLATION
*CONSTRAINED_JOINTS
*CONSTRAINED_LINEAR
*CONSTRAINED_NODAL_RIGID_BODY
(with and without local release conditions)

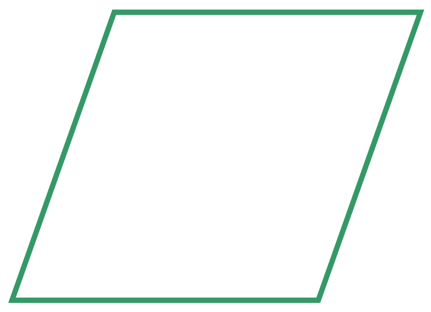
*CONSTRAINED_NODE_SET
*CONSTRAINED_POINTS
*CONSTRAINED_RIGID_BODIES
*CONSTRAINED_RIVET
*CONSTRAINED_SHELL_TO_SOLID
*CONSTRAINED_SPOTWELD
*CONSTRAINED_TIE-BREAK
*CONSTRAINED_TIED_NODES_FAILURE
*MAT_RIGID
(local or global SPC constraints)

EXAMPLE

To illustrate our new capability we created a simple example of 4 rigid bodies linked at the corners and initially in the shape of a square. In this case we want to drive the motion of the NE corner by rotations in the SW corner. Such a system is a simple model of mechanical systems such as part of an automobile suspension system or a drive mechanism for a robot arm. The bottom is fixed to have no movement. At the SW corner we have a revolute joint and a rotational motor on the rotation normal to the plane. We put spherical joints at the 3 corners.
To keep the system moving only in the x-y plane we put three additional constraints: no rotation around the y direction for the left edge; no rotation around the x direction for the top edge; and no translation motion in the z (out-of-plane) direction for the top edge. (The last constraint could have been put on the left edge.)

The resulting system moves as a parallelogram where the bottom left angle is specified via the rotational motor.





System with some rotation of the SW corner.

Figure 1. Schematic of Example

The interactions of the various constraints make this problem impossible to solve without factoring the $C_{D,D}$ matrix during the processing of the constraints. For each rigid body we used 2 endpoints and one midpoint. We also needed reference points added at the SW corner to both the bottom and left edges for the revolute joint. These 14 nodes led to 84 rigid body constraint equations. In addition we have a complicated set of 15 constraint equations describing the revolute joint, 3 spherical joints, and 1 motor constraints on the 4 center of masses. Nine other dofs belonging to 3 of the center of masses were suppressed with the single point (no motion) constraints. The cycle of 4 joints lead to a constraint matrix that can only be correctly processed with a global view of the constraint equations while labeling the dependent dofs and with a general approach to factorization of the constraint matrix.

The nonzero pattern of the constraint matrix is given in Figure 2. The first 84 rows correspond to the rigid body constraints. The last 15 correspond to the joints and motors.

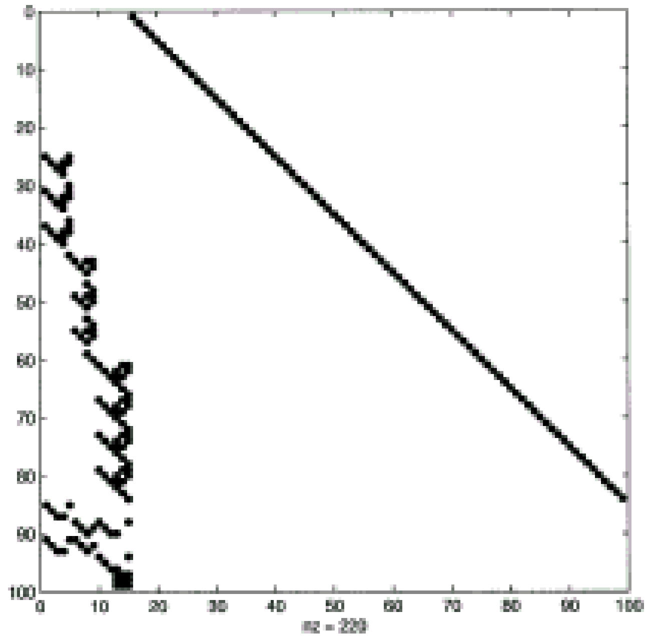


Figure 2. Pattern of Constraint Matrix

CONCLUSION

We have incorporated many of the equality constraint features used by explicit into implicit and linear analyses for v. 970 of LS-DYNA. We have also implemented these constraint features without any unnecessary restriction on the modeling by the user. The example demonstrates our ability to process many constraints that interact in such a way that would break the modeling assumptions made by other packages, giving LS-DYNA a unique and powerful capability. We expect that all of these new capabilities will assist our users in modeling complex physical systems using the implicit and linear analyses in LS-DYNA.