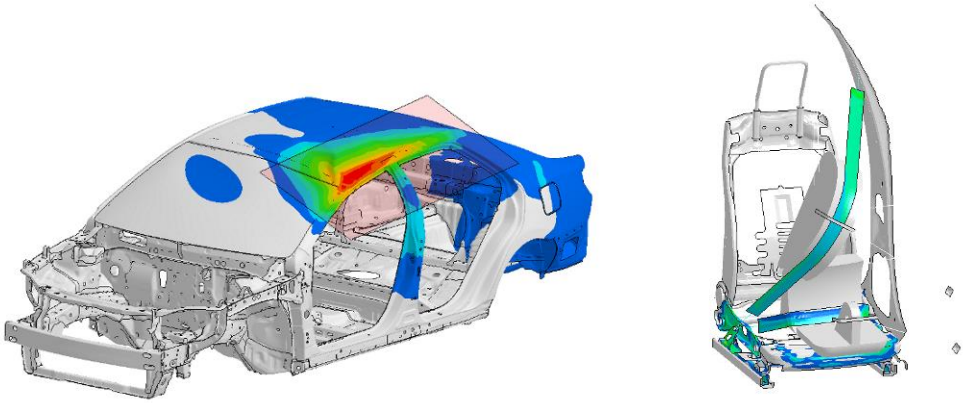
Quasi-Static Simulations using Implicit LS-DYNA®

Satish Pathy
LSTC, Livermore, CA

Thomas Borrvall
DYNAmore Nordic AB, Linköping, Sweden

Abstract

Quasi-static tests that are part of the federal regulations such as FMVSS 216[1] and FMVSS 207/210[1] is predominantly simulated using LS-DYNA's explicit solver. To be able to produce results in a reasonable time, mass-scaling and time-scaling are employed for these simulations, thereby the physics of the problem gets compromised. This paper demonstrates LS-DYNA's Implicit capabilities to solve quasi-static problems in a reasonable amount of time, without compromising physics. Recent development and improvements to the implicit solver enables us to solve problems such as this, accurately and with a run turn-around time that is inside the reactionary period of the design community. Two popular tests – Roof-Crush and Seat-Pull tests were chosen for this exercise.



Introduction

History

LS-DYNA [1] is a multi-physics solver that is historically renowned as an explicit code intended for the simulation of short term dynamic processes. This paper will present the work performed using the implicit solver in LS-DYNA, for brevity it is referred to as *LS-DYNA Implicit* throughout the paper. An implicit solver with limited features was introduced in version 950, primarily aimed at solving spring-back problems as a complement to standard explicit forming simulations. Since then LS-DYNA Implicit has evolved to a complete product, covering many different disciplines in linear and nonlinear simulation technology. During its lifetime, the nonlinear implicit solver has gone through many different phases, and robustness has sometimes been questioned. While this is partly due to the program itself it must also be said that many users, in particular those coming from an explicit background, lack experience that would allow them to tune implicit parameters for optimal performance. With gained maturity, a favorable price-performance index and an already strong foothold in the explicit market, an increased interest in LS-DYNA Implicit has been observed the last few years. To this end, a challenge facing many existing customers is to convert "explicit" input decks to run well with implicit settings. Due to rather different characteristics between explicit and implicit analysis in general, this does not necessarily mean to only add implicit cards and parameters, but may also require model adjustments to make it "implicit-ready". Part of the implicit development is devoted to facilitate, for users, in this respect by improved numerical algorithms, but much still relies on interventions by the user. However, it is the authors' opinion that with a methodical approach to these types of problems, LS-DYNA Implicit compares well to competitive software products, and a goal with this paper is to show how and why this is.

Development

As a starting point for discussing the basic philosophy behind implicit development, we refer to the simplistic overview of general numerical simulation in Figure 1. The goal with realistic simulations is to make predictions of a physical process, through the steps of *(i)* creating a mathematical model of the system in question, *(ii)* discretizing the model and *(iii)* solving the resulting numerical problem. Each of these three steps will introduce errors in our predictions, stemming from *(i)* model errors, *(ii)* discretization errors and *(iii)* solution errors. Examples of these are for instance *(i)* that we don't capture the actual response of a certain material, *(ii)* we use poor/inaccurate elements prone to locking or similar deficiencies or *(iii)* are unable to solve the numerical problem to a certain degree of accuracy. Obviously there is an interest to make the best possible predictions, i.e., minimize the accumulated error, achievable by for instance *(i)* using advanced material models, *(ii)* sophisticated elements (such as assumed strain or high order elements to alleviate locking tendencies) and *(iii)* tight tolerances. Unfortunately this is in direct conflict with another aim, namely to obtain results as quickly as possible; measures to increase accuracy will in general yield longer simulation times. It is therefore important to account for characteristics in the time discretization algorithm, explicit or implicit, when developing algorithms or model features for a reasonable trade-off between accuracy and efficiency.

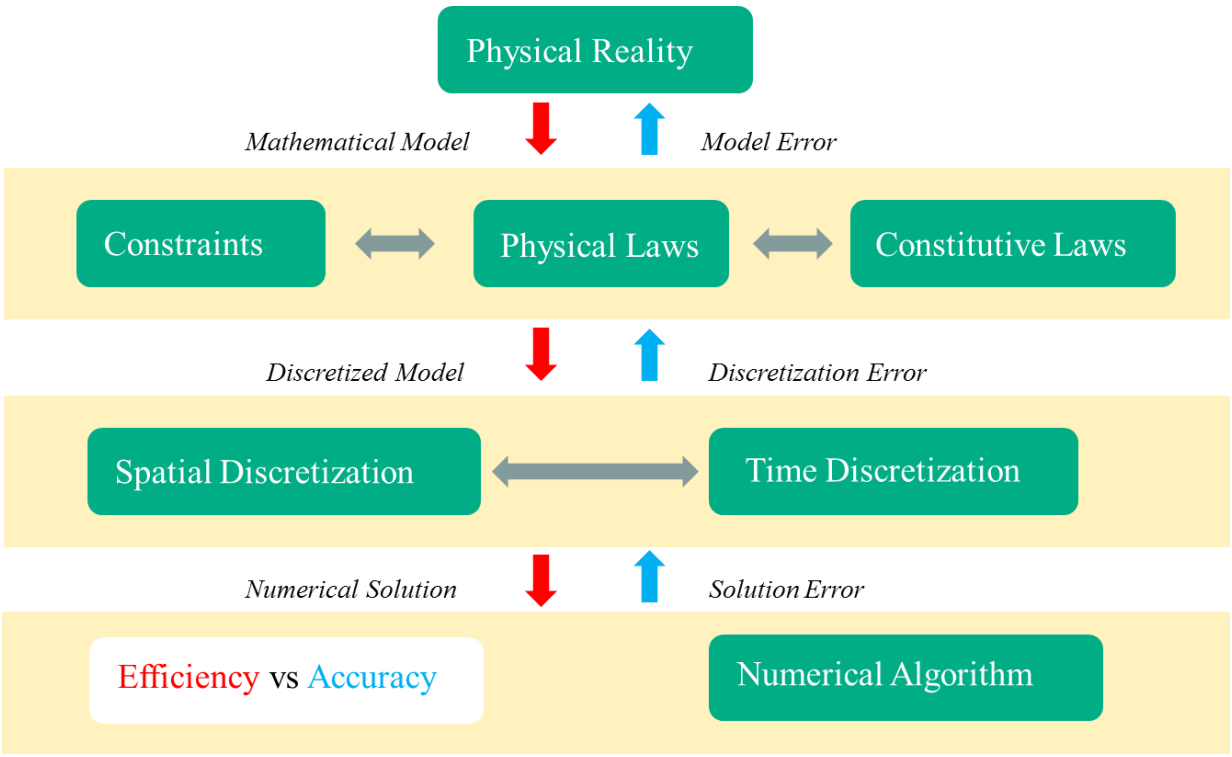


Figure 1 Numerical simulation chart

Explicit vs Implicit

Explicit analysis is characterized by many small time steps, and the major part of the simulation time is spent in elements, materials and contacts. The goal with explicit development can thus be summarized as to develop fast and *sufficiently* accurate algorithms for these three disciplines, to render a robust and accurate procedure.

For implicit analysis, the situation is in general quite different as the time step is significantly larger. A consequence of this is larger movements (deformation and/or rotation) within each time step, which in turn imposes stronger requirements on both accuracy and robustness of elements, materials and contacts. Furthermore, a great deal of the simulation time is inevitably spent in the numerical algorithm, since (large) systems of linear equations must be solved repeatedly to yield numerical iterations. The best recipe to shorten simulation times is therefore, next to having efficient linear solvers, to find ways to reduce the number of iterations in the numerical algorithm.

Accuracy vs Efficiency

In an implicit context, we believe in the correlation between *model accuracy* and *numerical efficiency*; accurate elements, materials and/or contacts, will improve convergence. The development of nonlinear implicit analysis is, contrary to that of explicit, in many respects focused on improving accuracy of model features because the extra time spent for that is believed to be made up by requiring less number of iterations. The notion that explicit and implicit analysis should always execute the same elements, materials and contacts is thus *not* adopted.

Finally, numerical errors may be significant in implicit analysis, which is partly remedied by having a good linearization of the model. To this end, adding implicit support of model features

by implementing representable tangent stiffness matrices is a continuous part of the implicit development. Accompanying this, from a user's perspective it is important that convergence tolerances are set to meet necessary requirements for equilibrium. While default values are a good starting point, adjustments may be needed if the simulation results reveal that this is called for, something that will be discussed in the presentation of the two numerical examples that constitutes the remainder of this paper. The intention is two-fold, first to present an approach to implicit simulations that is *repeatable*, meaning that it can be applied to other problems with little or no modifications, and second to show that LS-DYNA Implicit is able to solve quite complicated problems. For more information on implicit solution strategies, including tips and tricks, we refer to Appendix P in the LS-DYNA User's Keyword Manual [1].

Roof-Crush Simulation

The Model

The original FE model was developed by National Crash Analysis Center (NCAC) at George Washington University [3]. This public domain model was used for testing implicit solver's capabilities. Only the body-in-white (BIW) was used for the roof-crush simulation, the powertrain and closures were discarded. The public domain model had to be slightly modified for roof-crush analysis, mostly in a way of improving connections between parts, especially in the roof and windshield area. All of the model set-up and clean-up was performed in LS-PrePost [4].

FMVSS 216 test protocol dictates that the rigid loading device will load the roof structure at a rate of 13mm/second and the structure reaches the peak load of at least 1.5 x vehicle weight before a peak displacement of 5" is reached. A baseline was established by first running the model using explicit solver. The rate of loading had to be increased here, so the analysis could complete in a reasonable time. To get the model implicit-ready, several changes had to be made to the input deck. As discussed, it is not just that matter of adding implicit related control cards, and the steps and strategy used to get the roof-crush model implicit ready is discussed below. The model was simulated using implicit dynamics solver with loading applied at the same rate as the protocol dictated. The results from the both the approaches were compared.

Implicit Conversion

First and foremost, before running any implicit model, we have to make sure there aren't any parts that are 'unconnected', which might result in rigid body modes. So, to make sure of that, we first run the model to extract eigenmodes by inserting **CONTROL_IMPLICIT_EIGENVALUE*. One can choose any number of eigenmodes to be computed, the higher the number, larger the computational expense, for this model, we set *NEIG=50*. After fixing all the 'loose' parts, we run the model one more time to extract eigenmodes and if the first mode is a significant mode then we have completed the first step.

Next, we modified all penalty contacts to 'Mortar' contacts, mortar contact was originally developed for forming applications, but has since evolved into a go-to contact for implicit applications. Mortar contact [5**]** is a segment based contact with consistent coupling between non-matching discretization of two sliding surfaces. This consistency, together with a differentiable penalty function for penetrating and sliding segments assert the continuity and

smoothness in contact forces, that is very desirable for implicit convergence. Two Mortar contacts were used in this model (i) *CONTACT_AUTOMATIC_SINGLE_SURFACE_MORTAR and (ii) *CONTACT_AUTOMATIC_SURFACE_TO_SURFACE_MORTAR. The former was used to define the self-contact of the vehicle and the latter was used to define the contact between the rigid platen and the vehicle. The tied contact used for spotwelds was modified to use *CONTACT_TIED_SHELL_EDGE_TO_SURFACE_CONSTRAINED_OFFSET.

All the shell elements were modified to use element formulation 16 with implicit flag (IACC=1) activated in *CONTROL_ACCURACY, where the accuracy issues when subjected to large deformations/rotations over a single time step are improved. By activating the implicit accuracy flag in *CONTROL_ACCURACY, specific accuracy considerations in implicit analysis is honored. The values for flags in implicit control cards were chosen as follows.

- ✓ *CONTROL_IMPLICIT_GENERAL
  DT0 was set to 0.1, so to at least have 100 implicit steps
- ✓ *CONTROL_IMPLICIT_AUTO
  Allow time step size to be adjusted automatically, used a larger ITEWIN, so the time step is not adjusted unnecessarily. The number chosen here was a result of running the simulation more than once.
- ✓ *CONTROL_IMPLICIT_DYNAMICS
  Implicit dynamic analysis was activated. The reason to use this strategy was, as the roof deforms, severe buckling was observed and implicit-statics would struggle to achieve convergence. Numerical damping was invoked by setting GAMMA=0.6 and BETA=0.38.
- ✓ *CONTROL_IMPLICIT_SOLUTION
  NSOLVR=12, the new default non-linear solver along with Full Newton nonlinear solution method where the stiffness matrix is formed at every iteration was used. Nonlinear convergence norm type, NLNORM, was set to '4', were the sum of both translational and rotational degrees of freedom is considered. Finally a more robust and a brute force line search method, LSMTD=5 was used,. Convergence tolerances were set to use default values, except for ABSTOL. ABSTOL of '-3000' was found to be good for this problem, this cannot be chosen at the start of the project, you will need at least one complete run to arrive at a good ABSTOL number to use. This method to choose ABSTOL is explained in detail in Appendix P of the user's manual.

Results and Discussion

Both the simulations were run using 48cores, the explicit simulation took 2hrs 35mins for the full run with a termination time of 0.1second, time-step of 0.6µsec was used. The implicit simulation on the other hand took 19hrs to finish using a step-size of 0.1sec and termination time of 10seconds. The model has a total of 981k elements, with 959k deformable elements.

From the force-displacement plots, see Figure 2, it can be seen that the peak force of ~70kN is seen in both the runs, the peak load occurs sooner in the implicit run. Also, the deformation comparing the two runs look similar. Since we did not have a physical test to compare the results to, we are unsure which of these results are closer to reality. However, the goal and intent

of this study was to run a complicated, dynamic, and a highly non-linear model, successfully with robustness using LS-Dyna's implicit solver.

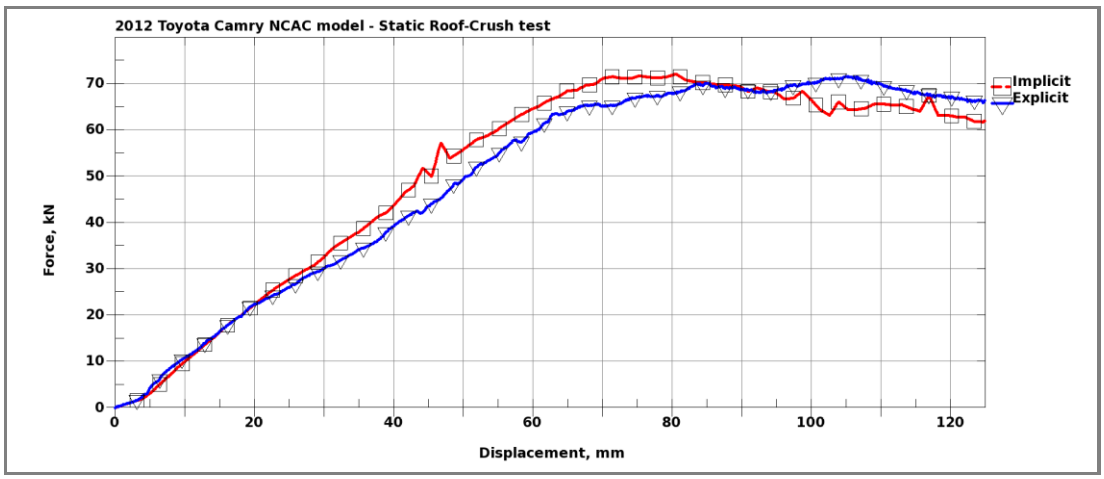


Figure 2: Force-Displacement plot

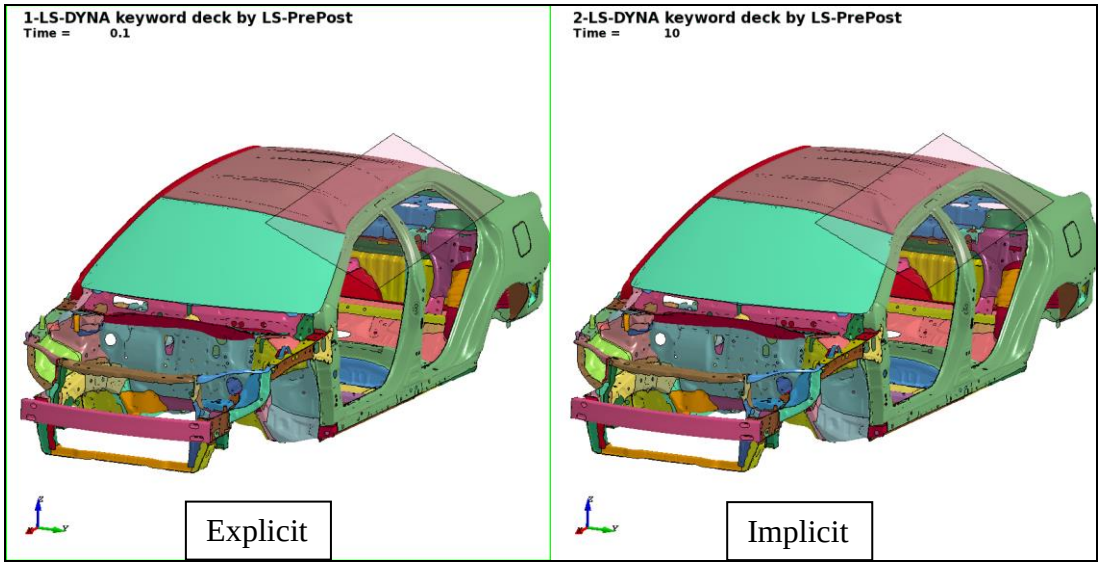


Figure 3 Final deformed state comparison

Seat-Pull Simulation

The Model

This seat from 2012 Toyota Camry FE model was also developed by NCAC at George Washington University. This public domain model was used to set up an implicit seat-pull example to study the capabilities of LS-Dyna Implicit Solver.

This was a simpler model compared to the vehicle model and set up followed the same strategy as stated for the previous example. Constraints were defined to rigidly fix the seat at the mounting locations. Seat foams were removed from the model as their contribution to seat integrity is negligible; also a severely deformed foam part can only mean trouble for any type of simulation. The shoulder blocks and lap blocks were positioned as the test protocol dictates.

To aid implicit convergence, a small amount of bending stiffness is required to be added to seatbelt elements. This can be done via two methods (i) *MAT_SEATBELT for 1D and 2D seatbelt elements (ii) *MAT_FABRIC if using thin shells (*ELEMENT_SHELL) to represent the seatbelts.

Results and Discussion

This simulation was run on a stand-alone workstation using 16-cores, the explicit run took 5hrs 45mins whereas the implicit run took 6hrs 10mins. The results between the two were comparable, in the sense there wasn't any failure seen in the seat structure. The localized buckling in the seat structure meant the problem was highly non-linear, so using Full Newton non-linear solver was necessary.

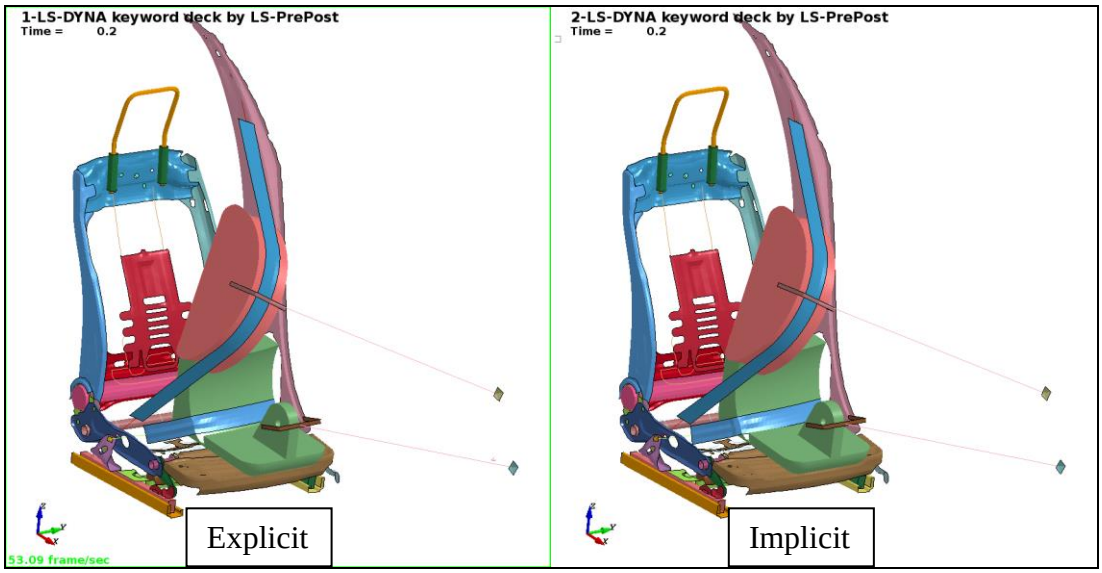


Figure 4 Final deformed state of seat structure

Summary

To get an explicit model to be implicit ready, the effort can be significant, although our developers are working to make this transition easier. Some of the effort we spent on roof-crush model was largely due to our inexperience in dealing with such complex problems for implicit. However, once the models were made suitable for implicit, they were robust enough to run to completion for every change and settings we experimented them with. Also, it is fair to expect that added complexity in an explicit model will mean slightly more effort to get it running comfortably in implicit. However, the solver can only get better if we get ample feedback from the users. LSTC and DYNAmore are continuously improving the implicit solver and are

confident that the solver will be able to handle larger and more complex problems as we make progress.

These and several other implicit examples can be downloaded from www.dynaexamples.com/implicit.

Acknowledgements

The authors would like to thank Singai Krishnamoorthy, Atieva Corporation for providing the initial explicit roof-crush model. We would also like to thank Rudolf Reichert, Center for Collision Safety and Analysis, George Mason University for providing the seat model.

References

[1] Federal Motor Vehicle Safety Standards and Regulations, U.S DOT, NHTSA

[2] LS-DYNA Keyword User's Manual, R9, Livermore Software Technology Corporation (LSTC), 2016.

[3] NCAC, http://www.ncac.gwu.edu/vml/models.html

[4] LS-PrePost, *www.lstc.com/lspp*

[5] Mortar Contact Algorithm for Implicit Stamping Analysis in LS-Dyna, Thomas Borrvall, 10th International LS-Dyna User's Conference, 2008