APPENDIX P: Implicit Solver

Introduction

The terms implicit and explicit refer to time integration algorithms. In the explicit approach, internal and external forces are summed at each node point, and a nodal acceleration is computed by dividing by nodal mass. The solution is advanced by integrating this acceleration in time. The maximum time step size is limited by the Courant condition, producing an algorithm which typically requires many relatively inexpensive time steps.

While explicit analysis is well suited to dynamic simulations such as impact and crash, it can become prohibitively expensive to conduct long duration or static analyses. Static problems, such as sheet metal springback after forming, are one application area for implicit methods. In the implicit method, a global stiffness matrix is computed, inverted, and applied to the nodal out-of-balance force to obtain a displacement increment. The advantage of this approach is that time step size may be selected by the user. The disadvantage is the large numerical effort required to form, store, and factorize the stiffness matrix. Implicit simulations therefore typically involve a relatively small number of expensive time steps.

In a dynamic implicit simulation these steps are termed *time steps*, and in a static simulation they are *load steps*. Multiple steps are used to divide the nonlinear behavior into manageable pieces, to obtain results at intermediate stages during the simulation, or perhaps to resolve a particular frequency of motion in dynamic simulations. In each step, an equilibrium geometry is sought which balances dynamic, internal and external forces in the model. The *nonlinear equation solver* performs an iterative search using one of several Newton based methods. *Convergence* of this iterative process is obtained when norms of displacement and/or energy fall below user-prescribed tolerances. Within each implicit iteration there is a *line search* performed for enhancing robustness of the procedure.

The implicit analysis capability was first released in Version 950. Initially designed for metal forming springback simulation, this new capability allowed static stress analysis. Version 970 added many additional implicit features, including new element formulations for linear and modal analysis. A milestone in implicit was the advent of Version 971 in which implicit analysis was carried over to MPP and thus allowed much larger problems to be solved, and from there it has evolved extensively to presently contend well among competitive softwares. Still, it can be a gruesome task to set up an implicit input problem that runs all the way to completion with acceptable results, especially when contacts are involved. The purpose of this text is to explain keywords of interest and suggest values of important parameters in order to give users tools for developing own solution strategies to these kinds of problems.

APPENDIX P

A prerequisite for running implicit is to use a double precision version of LS-DYNA, and a machine with a significant amount of memory. A theoretical overview of implicit can be found in the LS-DYNA Theory Manual.

Nonlinear Implicit Analysis

Activating Implicit Analysis

The keyword *CONTROL_IMPLICIT_GENERAL is used to activate the implicit method, and in principle it is sufficient to set

IMFLAG = 1
DT0 = some reasonable initial time step.

The initial time step should be chosen small enough to resolve the frequency spectrum of interest and/or provide decent convergence properties, but large enough to benefit from an implicit analysis. With no other implicit options, this converts an explicit input deck to a static implicit input deck with a constant time step throughout the simulation and the problem will terminate upon convergence failure. Time stepping strategies to prevent this are discussed below.

Singularities and Eigenvalue Analysis

The first concern in implicit analysis is to prevent singularities in the stiffness matrix, otherwise the chance of succeeding is close to none. The major cause of a singular stiffness matrix is the presence of rigid body modes in a static problem which can be revealed in an eigenvalue analysis. This is done by using *CONTROL_IMPLICIT_EIGENVALUE and setting

NEIG = number of modes to extract.

First, run an eigenvalue analysis and extract (if practically possible) enough modes to see all the rigid body modes just to get an impression of the properties of the model. The frequencies are in the output text file eigout and the mode shapes can be animated in the binary output file d3eigv using LS-PrePost. Some rigid body modes may come as a surprise and should be constrained. A typical example of this is a beam that is free to rotate around its own axis. Other rigid body modes are a consequence of the nature of the problem and cannot be constrained without destroying the connection to reality. An example of this could be components that are to be constrained with contacts but are initially free to move. There are several strategies to deal with these latter rigid body modes; some are discussed in the context of contacts but here dynamics is used.

Dynamics and Intermittent Eigenvalue Analysis

Dynamics alleviates singular rigid body modes and is activated on *CONTROL_IMPLICIT_DYNAMICS by setting

IMASS = 1 or a negative number.

If the purpose is to solve a dynamic problem in the first place, then just use IMASS = 1 and don't bother about the other parameters; otherwise some other strategy is necessary. The idea is start with a dynamic analysis until the rigid modes have been constrained by contacts at which point dynamics is turned off. The simplest way of doing this is to set

TDYBIR = 0
TDYDTH = time when contacts have been established and rigid body modes are constrained
TDYBUR = time after TDYDTH for fading out dynamics between TDYDTH and TDYBUR.

If the dynamic results are of no interest but just a way to proceed to the static solution, then it is recommended to use numerical damping to prevent unnecessary oscillations, that is, set

GAMMA = 0.6
BETA = 0.38.

This should be enough to start up most problems of interest. A restriction with this is that when dynamics has been turned off it cannot be turned on again. If this is necessary for some reason, then a negative number of IMASS should be used to control dynamics through a load curve.

If it is hard to deduce how to choose the time when to turn off dynamics, the use of intermittent eigenvalues may be of great help. By setting

NEIG = a negative number

on *CONTROL_IMPLICIT_EIGENVALUE the user may extract eigenvalues at given time points during a nonlinear simulation and deduce from that if all rigid body modes have been eliminated.

The Geometric Stiffness Contribution

Stiffness singularities may also occur during a simulation due to a complex global phenomenon involving the stress state and geometry. For instance, slender components with compressive stresses could give rise to zero eigenvalues, commonly known as buckling. The mathematical explanation for these kinds of singularities is that the material and geometric stiffness contributions cancel out, and for this reason the geometric contribution to the stiffness matrix in LS-DYNA is optional. It is activated by putting

IGS = 1

on *CONTROL_IMPLICIT_GENERAL. Most often singularities due to this contribution have a negative effect on convergence although it sometimes helps. It is recommended to leave this as default and turn it on if other strategies fail. A controlled way of getting past singular points of this type is to use so called arc length methods for which the geometric stiffness should be turned on. This way of dealing with the problem is discussed in the Theory Manual.

BFGS or Full Newton

The nonlinear solver parameters are set on *CONTROL_IMPLICIT_SOLUTION where the solver type is specified as

NSOLVR = nonlinear or linear implicit analysis option.

By default, a nonlinear BFGS solution strategy is used when the stiffness matrix is reformed every 11th iteration and a maximum of 15 reformations is allowed (for a linear solution set NSOLVR = 1). These parameters are set by

ILIMIT = iterations between reforming stiffness matrix
MAXREF = maximum number of stiffness reformations

on *CONTROL_IMPLICIT_SOLUTION. Reforming the stiffness matrix is computationally expensive but stabilizes the solution procedure, and the best strategy in this context is very much problem dependent. The recommendation is to start with the default strategy and if necessary, change them. For hard problems decrease ILIMIT while for problems that converge well increase ILIMIT. For really bad problems switch to full Newton (the safe bet) by for instance setting

ILIMIT = 1
MAXREF = 30.

Keep the maximum number of reformations reasonably low or otherwise it may take an unnecessary amount of time just to reach convergence failure.

Convergence

Tolerances

Convergence is based on changes in displacements, energies and optionally residual forces, and the tolerance levels are set by

DCTOL = Relative displacement tolerance
ECTOL = Relative energy tolerance
RCTOL = Relative residual tolerance
ABSTOL = Absolute tolerance
DMTOL = Maximum displacement tolerance
EMTOL = Maximum energy tolerance
RMTOL = Maximum residual tolerance

on *CONTROL_IMPLICIT_SOLUTION. The tolerances on the left are based on Euclidian norms of the involved vectors, whereas the ones on the right are based on maximum norms. The maximum norm tolerances are optional and should be activated if increased accuracy is desired at the price of more implicit iterations. To keep things simple, the discussion that follows pertains only to the Euclidian norm tolerances.

By default, the progress of the equilibrium search is shown to the screen. The box below shows a typical iteration sequence, where the norms of displacement and energy are displayed. When these norms are reduced below user prescribed tolerances (default 0.001 and 0.01, respectively), the iteration process is said to have *converged*, and the solution proceeds to the next time step. The message files, messag in SMP and mesxxxx in MPP, typically contain a whole lot more information that will be dealt with further in output diagnostics and convergence issues.

```
BEGIN static time step     3
============================================================
               time =  1.50000E-01
  current step size =  5.00000E-02
Iteration:   1     *|du|/|u| =  3.4483847E-01     *Ei/E0 =  1.0000000E+00
Iteration:   2     *|du|/|u| =  1.7706435E-01     *Ei/E0 =  2.9395439E-01
Iteration:   3     *|du|/|u| =  1.6631174E-03     *Ei/E0 =  3.7030904E-02
Iteration:   4     *|du|/|u| =  9.7088516E-05     *Ei/E0 =  9.6749731E-08
```

Premature Convergence

The last of these parameters (ABSTOL) overrides the other three (DCTOL, ECTOL, RCTOL) in the sense that if the residual force is small enough, convergence is detected regardless of if the other three criteria are fulfilled or not. It has been seen that this sometimes give rise to so called *premature convergence*, converged states are not *really* converged in the sense that the residual norm is small enough. This gives bad results and it is usually recommended to tighten this tolerance to 10^{-20} to prevent this. On the other hand, if the problem is very hard this may prevent convergence to the extent that going back to the default is more or less necessary. It is difficult to give a general recommendation that holds for every problem.

As for the other three parameters, the one that usually comes into practice is the displacement criterion DCTOL. The energy criterion ECTOL is often easy to fulfill and the residual criterion is disabled by default and there may be no reason to activate this (that is unless a completely different strategy is used, or if convergence issues arise). Using the default values on all three is usually good enough to give acceptable results in decent time. For problems with poor convergence, the question of tightening or relaxing these parameters (that is the displacement convergence criterion) is up for debate. It is tempting to believe that relaxing the constraints will give better convergence which may or may not be true. Sloppy convergence criteria may once again result in premature convergence, and this will have a negative effect on the subsequent steps. The general recommendation would have to be to leave these unchanged, and be aware that relaxing them may not result in getting further in the implicit simulation.

Using Residual Tolerance

A novel strategy that seems to make sense and works well for some problems is to only use tolerances on residual forces RCTOL. This relies on the degree of nonlinearity of the simulation model being moderate, but it could be well worth trying. The first thing to do would be to set DCTOL and ECTOL to large numbers to disable them while setting RCTOL to a relatively small number (start with 0.01 and see how that works out). This must be complemented with an absolute tolerance on the residual forces to get past the initial stages where the residual forces are small enough to not fulfill the relative tolerance. To do this, set ABSTOL to a negative number, which is a different absolute criterion compared to setting it to a positive number. Now, this number is very problem dependent as this says that convergence is attained as soon as the Euclidian norm of the residual force vector is smaller than the absolute value of ABSTOL. It goes without saying that this requires running the problem for a few steps to monitor the residual norm in the message files as described below; this should give an indication on how to determine the ABSTOL value.

Convergence Norms

The degrees of freedom encountered in an LS-DYNA simulation are either *translational* or *rotational,* where the latter comes from the presence of beams and shells. Historically, convergence check is on norms of the translational degrees of freedom only, which is unsettling because rotational residual forces (moments) are equally important in an implicit simulation. It is therefore recommended to use

NLNORM = choice of convergence norms

on *CONTROL_IMPLICIT_SOLUTION to change this default (which is = 2). While NLNORM = 3 will incorporate residual degrees of freedom separately and is a more satisfying approach, the recommended option is to use NLNORM = 4 or a negative number. NLNORM = 4 will treat the entire force and displacement vector, respectively, as one complete mathematical vector and the convergence norms and scalar products used for checks are simply the Euclidian norms of these vectors. To make the displacement and residual norms unit consistent, a scale factor l, corresponding to a characteristic element size in the model, is introduced and accounted for when computing the norms. This means for instance that displacements are summed to l length unit times angular increments, and moments are summed to l length unit times forces. For NLNORM = 4 the scale factor is echoed to the output files and screen. If the internally calculated value is inappropriate for some reason, it may be convenient to use NLNORM = -(length scale) to supersede this scale factor.

Time Stepping Strategies

By default, LS-DYNA will terminate when a step fails to converge. This is unfortunate since it may just be that the step is too large to achieve convergence and taking smaller steps would solve this problem. Instead of starting all over with a smaller step size, set

IAUTO = 1 to activate automatic time stepping

on *CONTROL_IMPLICIT_AUTO. When the problem fails to converge LS-DYNA will with this option go back to the previously converged state and retry with a smaller step size. If the problem converges well, the step size is increased for subsequent steps. This process of backing up and retrying difficult steps lends much persistence to the analysis and is often the only procedure for solving highly nonlinear problems short of adjusting the step size manually. We recommend always turning automatic time stepping on.

Another parameter worth mentioning in this context is

DTMAX = Max time step allowed, or a negative number

on the same card. This is the maximum step size possible and should be chosen to not lose necessary information in the results, such as resolving frequencies, contacts and non-linear material response to a satisfactory degree. A negative value of this parameter will give the maximum step size as a function of the simulation time and allows for hitting key points that are of interest. The user may for instance look for the peak stress for a certain external load and hitting the point in time when this happens is crucial. It also allows for taking smaller steps during critical stages in the simulation without wasting resources away from these stages.

Line Search

A good line search strategy is crucial in solving nonlinear implicit problems; without it only simple problems would be solvable. There are a few line search options available and these are activated by

LSMTD = Line search method

on *CONTROL_IMPLICIT_SOLUTION. The default method is based on minimizing a potential energy along the search direction and at the same time keeping track of the residual force magnitude. It also detects when a BFGS step results in negative initial energy which causes the stiffness matrix to be reformed for robustness (this can only happen when the stiffness matrix has negative eigenvalues). Line search type 2 is based on residual forces and is more robust than the others; the drawback is that it typically results in too small steps and is not practically useful. Line search type 3 is very similar to type 4, but it does not track the residual force or avoid negative volumes to the same extent. In sum, there is no practical reason for choosing methods 1 through 3. Worth mentioning is however line search type 5 which combines the energy and residual method in a stricter sense. That is, it minimizes the potential energy just like type 4, but it only allows the residual norm to double at each implicit iteration. This has shown to be robust but slow in convergence and is to be used for problems that have difficulties converging with the default line search strategy. This has had a huge impact on the treatment of rubber models for instance. An interesting combination is to use line search method 5 together with the strategy of converging based on residual forces only.

Finally, the line search tolerance

LSTOL = Line search tolerance

on *CONTROL_IMPLICIT_SOLUTION is fine as it is; don't change it.

Output

As previously mentioned, the output to the message file is more extensive than that to the screen/stdout. To novice or average implicit users this information may be more than can be digested at first and this section may be skipped, but with experience the output can become really useful when running into convergence issues. For this discussion it is important to distinguish between different *types* of iterations, and we use the term *time/load step* to mean the actual advance in simulation time, *implicit iteration* to indicate points in the iterative solution procedure where a new search direction is obtained (corresponding to either Newton or BFGS) and *line search iteration* to indicate the search for an optimal step size along a given search direction. We begin by showing a typical iteration output for when NLPRINT = 3 following by a numbered list explaining the content.

```
[1]BEGIN implicit dynamics step      2 t = 2.5849E-01
 ============================================================
                time =  2.58489E-01
   current step size =  1.58489E-01
.
.
.

[2]Newton step computed
 Initial translational energy =  1.4031740E-01
[3]Initial total energy         =  1.4031740E-01
 Initial residual norm        =  7.5755334E-01
[4]Translational nodes norm     =  7.1314670E-01
 Translational nodes max      =  1.9159278E-01 at node         440788
 Translational rigid body norm =  3.5324028E-03
 Translational rigid body max =  2.3897586E-03 at body        4000024
 Rotational rigid body norm   =  2.5553155E-01
 Rotational rigid body max    =  1.7834357E-01 at body        4000024

[5]Evaluated residual for full step
 Current translational energy =  9.0141343E-02
[6]Current total energy         =  9.0141343E-02
 Current residual norm        =  2.2038666E+00
 Translational nodes norm     =  2.1950775E+00
 Translational nodes max      =  1.0295509E+00 at node         440984
 Translational rigid body norm =  3.4677960E-02
 Translational rigid body max =  2.6701145E-02 at body        4000024
 Rotational rigid body norm   =  1.9354585E-01
 Rotational rigid body max    =  1.2669458E-01 at body        4000024

[7]Evaluated residual for step size  6.6666667E-01
 Current translational energy =  1.1123844E-01
 Current total energy         =  1.1123844E-01
 Current residual norm        =  1.5007375E+00
 Translational nodes norm     =  1.4922142E+00
 Translational nodes max      =  7.1187918E-01 at node         440984
```

```
 Translational rigid body norm =  1.6121536E-02
 Translational rigid body max =  1.2659274E-02 at body        4000024
 Rotational rigid body norm   =  1.5890244E-01
 Rotational rigid body max    =  1.0852930E-01 at body        4000024

 Evaluated residual for step size  3.3333333E-01
 Current translational energy =  1.2543971E-01
 Current total energy         =  1.2543971E-01
 Current residual norm        =  8.3754668E-01
 Translational nodes norm     =  8.1775945E-01
 Translational nodes max      =  3.0290699E-01 at node         440984
 Translational rigid body norm =  5.4770134E-03
 Translational rigid body max =  4.5377150E-03 at body        4000024
 Rotational rigid body norm   =  1.8089755E-01
 Rotational rigid body max    =  1.2913973E-01 at body        4000024

 Line search continues
 Max relative step            =  0.3499001E+00
[8]Lower bound                =  3.3333333E-01
 Upper bound                  =  1.0000000E+00

 Evaluated residual for step size  7.7777778E-01
 Current translational energy =  1.0487486E-01
 Current total energy         =  1.0487486E-01
 Current residual norm        =  1.7421066E+00
 Translational nodes norm     =  1.7342352E+00
 Translational nodes max      =  8.2830775E-01 at node         440984
 Translational rigid body norm =  2.1464316E-02
 Translational rigid body max =  1.6693623E-02 at body        4000024
 Rotational rigid body norm   =  1.6402030E-01
 Rotational rigid body max    =  1.0799892E-01 at body        4000024

 Evaluated residual for step size  5.5555556E-01
 Current translational energy =  1.1674955E-01
 Current total energy         =  1.1674955E-01
 Current residual norm        =  1.2575584E+00
 Translational nodes norm     =  1.2472759E+00
 Translational nodes max      =  5.8455890E-01 at node         440984
 Translational rigid body norm =  1.1655353E-02
 Translational rigid body max =  9.2884622E-03 at body        4000024
 Rotational rigid body norm   =  1.6006264E-01
 Rotational rigid body max    =  1.1222751E-01 at body        4000024

[9]Line search converged in    2 iterations
[10]Max relative step          =  7.1106671E-01

[11]Iteration:    5    displacement            energy              residual
 ----------           not conv.             not conv.            converged
 norm ratio           1.013E+00             1.000E+00              n/a
 current norm         8.446E-01             1.403E-01            1.258E+00
 initial norm         8.338E-01             1.403E-01            7.575E+00
-------------    trans       rot      trans       rot        trans       rot
max node norm   8.533E-02  0.000E+00  2.052E-02  0.000E+00  5.846E-01  0.000E+00
   at node ID    4000674    9007526     440984    9007526     440984    9007526
-------------
       RB max   3.449E-02  6.613E-04 -5.507E-04  7.681E-05  9.288E-03  1.122E-
01
     at RB ID   4000024    4000024    4000024    4000024    4000024    4000024
```

Referring to the superscripts at the beginning of the lines in the excerpt above, and taking them in order of appearance, we have at

1. Beginning a *dynamic* implicit time/load step, the goal is to get to a given time point (in this case 2.5849E-01 using a step size of 1.58489E-01). Implicit time/load steps are either dynamic, *static* or *semidnmc*, where the last one indicates a transition phase between dynamics and statics.

2. A *Newton* implicit iteration is performed, meaning that a full reformation of the stiffness matrix has just been made. Other implicit iterations can be either a *linear* if at the first iteration of a time/load step, or *BFGS* if the implicit iteration corresponds to a quasi-Newton step.

3. Initial norms are displayed at the beginning of each implicit iteration; these numbers are used as reference for the line search algorithm when checking line search convergence.

4. Detailed initial norms are displayed, separating out translational/rotational as well as nodal/rigid body Euclidian/max norms. The node number and rigid body attaining the global maximum norms are displayed and can be used to spot critical points in the model. In this particular case node 440984 and part 4000024 would be the likely candidates for trouble.

5. A full step along the search direction is taken, as a candidate for the next implicit iteration.

6. Norms for the full step is displayed. The line search algorithm compares these with the ones from 3 to deduce if the line search convergence criterion is fulfilled.

7. Line search is in this case needed, and information for further iterates along the search direction is displayed in analogy to previous line search iterates. If no line search had been needed, the string *Line search is skipped* would have been displayed.

8. The way line search is performed, each line search iteration narrows in on the optimal step and the bounds within which the optimal step is sought is continuously displayed. The size of this interval becomes smaller and smaller with increasing number of line search iterations, but convergence should preferably be attained before it tends to zero.

9. Line search converged here in 2 iterations; as a rule of thumb the number of iterations for line search should be about 10 or less. For critical steps one might accept 20 iterations but that should be among the exceptions. If the string *Line search did not converge* is displayed, that is an indication of something being terribly wrong.

10. After each implicit iteration, *Max relative step* indicates how large steps have been taken during line search so far. It also indicates how well prescribed motion is approximated where a value of 1 is perfect. For values below 1 and in the presence of non-zero prescribed motion, convergence will not be accepted even though all norms are within prescribed tolerances. This will instead issue the warning *Convergence prevented due to unfulfilled boundary conditions* and iterations continue.

11. Diagnostics for iteration 5 is displayed; similar numbers as shown during line search but in another format. Furthermore, the initial norms here refer to the initial norms from the beginning of the implicit time/load step and not from the beginning of the implicit iteration. This table can be used to deduce how close the implicit time/load step is at converging. The single important number for comfortably assessing the accuracy of an implicit iterate is the residual norm, currently displayed as 1.258E+00 (bold-faced in the table), since this should be small for good results.

At equilibrium, the output ends with a summary on how convergence was achieved. It may look like

```
Convergence detected as a combination of
      1.Ratio of euclidian displacement
         Value =   2.3417E-03 vs Tolerance =  2.5000E-3
      2.Ratio of euclidian energy
         Value =   3.2269E-08 vs Tolerance =  1.0000E-2
```

which tells us that the ratio of displacement was below DCTOL = 2.5E-3 and the ratio of energy was below ECTOL = 1E-2. If other criteria are satisfied, these will be listed as well.

So, with NLPRINT = 3 information from every force evaluation is given, including type of step taken, the line search step size, the potential energy value used in line search and the magnitude of the residual force. A point is made here, already touched upon and will be repeated, and it is the following: the nonlinear implicit solver is solving for zero residual force, so basically the number observed for the magnitude of the residual force should be small upon convergence (bold-faced in the table). If it is not, then the convergence is premature, and the results may not be correct and subsequent steps are in danger. What "small" means in this context is hard to say since this depends on units as well as loads and geometry of the problem, but a good sign is that the residual force does not grow by more than the external loads in the problem do. Another thing to observe is how the line search is doing; if the line search starts needing many iterations and very large variations in the residual force along the search direction is observed, then it is likely that something in the model is causing this behavior. A safe bet is that contact states are changed along the line search direction causing discontinuities in the residual force and indicates that some remodeling has to be done.

APPENDIX P

We will come back to this section when discussing convergence problems and strategies to prevent them.

Linear Equation Solver

General

Within each equilibrium iteration, a linear system of equations of the form $\mathbf{K}\Delta\mathbf{u} = \mathbf{R}$ must be solved. To do this, the stiffness matrix **K** is factorized and applied to the out-of-balance load or residual **R**, yielding a displacement increment $\Delta\mathbf{u}$. Storing and solving this linear system represents a large portion of the memory and CPU costs of an implicit analysis. LS-DYNA uses a multi-frontal sparse direct solver. For a problem with N number of nodes, the number of operations and memory storage is asymptotically proportional to $N^{3/2}$ and $NlogN$, respectively. This should give a ballpark indication on the growth when increasing the size of a given model. It must be stressed, however, that it is *a priori* difficult to predict the actual value of these numbers as they are highly dependent on the problem solved, in particular on the nodal connectivity through elements and contacts.

We here focus on the memory, and the purpose is twofold. First it is a documentation of the memory diagnostics that is written to the output message files (messag in SMP and mesxxxx in MPP) of LS-DYNA for the linear solvers. The level of information is regulated by the

LPRINT = Linear solver diagnostics level

input parameter on *CONTROL_IMPLICIT_SOLVER, and this text will cover LPRINT = 2 as the information for higher values is of no particular interest to others than developers. Second it should help understanding the interrelationship between the size of a model, and the memory required to obtain a solution. We emphasize however that different classes of problems most likely require different guidelines (shell and solid structures typically result in different matrix topologies for instance) and to this end the user is left at earning this experience on his own.

Memory

The memory is specified and reported in terms of *words*, where 1 word is equivalent to 8 bytes in double precision arithmetic. To simplify things, a word can in this context be seen as the equivalent of a real number or an integer. For implicit calculations the executable is only provided in double precision.

When starting a simulation, the user typically specifies the *static* memory, for instance in SMP

```
ls-dyna i = in.k memory = 200m
```

meaning that LS-DYNA tries to allocate 200 Mwords for storing statically allocated data. In MPP, the physical memory is distributed among processes and this option applies to each individual process. In MPP there is also an option to add a second memory flag, for instance

```
mpp-dyna i = in.k memory = 1000m memory2 = 200m
```

and this means that 200 Mwords are allocated for each process while 1000 Mwords is allocated on the first process to handle everything up to the point when the model data has been distributed among all the involved processes. The memory size may also be specified on *KEYWORD.

Beginning with LS-DYNA version R11.0.0, more and more of the statically allocated memory is being moved to dynamically allocated memory. For explicit simulation, the will allow real time adaptivity and dynamic load balancing. Implicit has migrated all of the linear algebra storage (matrices, factorization) to dynamically allocated storage. This vastly simplifies the specification of memory and memory2 for implicit jobs.

If the model is initially explicit, the following message will be printed:

```
Memory required to process keyword: 38147436
```

For this case memory = 39m should be used on the command line.

For SMP, implicit will require an additional $(19 + 2 \times \text{ILIMIT}) \times (6 \times \text{NUMNP})$ words of memory for a transient nonlinear simulation where ILIMIT is specified on *CONTROL_-IMPLICIT_SOLUTION (default of 11; for modal or linear analysis ILIMIT = 0) and NUM-NP is the number of nodes in the model. To approximate the minimum value of memory needed to run the implicit analysis using SMP, one could add this additional value to the "Memory required to process keyword". In practice, one doesn't need to go to the trouble of calculating the required memory since the code will tell you what is needed as explained below.

For MPP, the nodes will be distributed across p processors, so a rough estimate of the addition to memory will reduced by $1/p$.

For an actual implicit execution look for

ALERT

which will tell you exactly the setting to use.

When a certain feature requires a slot in the memory, this is reported in the output text files, for instance as

```
contracting memory to    2306940 d 696365    implicit friction
expanding   memory to    2306955 d 696365    joints
```

The first number in each line above is the current value of statically allocated memory. The second number (following the "d") is the current value of dynamically allocated memory.

In the example above, the static memory pointer after having allocated for implicit friction is at 2306940; allocating the joints reserves 15 words and the static memory pointer is incremented accordingly. So the memory used up to this point is roughly 2.3 Mwords for this particular process. At the end of the initialization a memory report is written and can look like

```
S t o r a g e  a l l o c a t i o n
Memory required to begin solution      :     929 k
Linear Alg dynamically allocated memory:     196 k
Additional dynamically allocated memory:    2285 k
                                  Total:    3409 k
```

The total amount of memory used up to this point is roughly 3.5 Mwords, which is partitioned in the static memory specified and *dynamic* memory. Whenever a memory violation occurs, for instance when trying to allocate more memory than physically available or if more memory needs to be specified, LS-DYNA terminates with appropriate error information. The dynamic memory for the linear solver is allocated thereafter, which is the topic for the next section.

Linear Solver Memory Consumption

For LPRINT = 2, some memory information is given in the output files, and this is here presented in order of execution, thus as they appear in the files. We repeat that this is the information obtained in the message files and is thus referring to the memory consumption for this particular process if MPP is considered. For the sake of completeness we also cover the diagnostics related to the CPU time for the involved stages in the linear solution.

The last statically allocated memory for implicit is reserved and reported as

```
expanding   memory to    125081 d 2652145 matrix assembly allocation
```

After this allocation all remaining memory is dynamically allocated. All storage for the linear algebra modules is dynamically allocated. Typically this will be a 3 phase process of symbolical factorization, numeric factorization, and numeric solve.

Before the symbolic factorization takes place, current information on the system matrix and workspace reserved for implicit is reported

```
local number of rows      =             68490
local size of matrix      =           2338945
```

The first number refers to the number of independent degrees of freedom in the model, meaning the number of rows in the system matrix, and the second to the number of non-zero entries before factorization takes place, that is, after assembly.

The symbolic factorization is now performed and information on the CPU time for doing this together with the estimated size of the factorized matrix is reported.

```
CPU: symbolic factor      =          0.200
WCT: symbolic factor      =          0.203
storage currently in use  =        1038915
storage needed            =       12913838
factor speedup            =     1.9197E+00
solve  speedup            =     1.9378E+00
est.  factor nonzeroes    =     5.1432E+07
est.  factor operations   =     1.2099E+11
est.  total factor nz.    =     9.9666E+07
est.  max.  factor op.    =     1.2099E+11
est.  total factor op.    =     2.3293E+11
est.  max.  factor nz.    =     5.1432E+07
```

The first two rows reports the CPU and wall clock time for doing the symbolic factorization in terms of seconds. The next two are storage requirements for the symbolic factorization, of which the latter is the memory reserved for this. After this we have estimations of the speedup in the factorization and subsequent solve of the linear system of equations; this is a report from an MPP run on 2 processors. The last lines refer to the estimated requirement for storing and factorizing the system matrix and distinguish between the local, total and maximum number of each of these two entries. This allows for determining the memory scaling in the case of an MPP run.

Now LS-DYNA is in the position of reserving space for the factorization of the system matrix, and a report on the storage needed for this is given.

```
symbolic storage        1 =           1038915
in-core numeric  storg  1 =          62762269
out-of-core num.  storg 1 =          13168676
```

Here basically the same information is repeated, except for a slight increment in order to account for potential penalty when doing the actual factorization. The first is again the memory reserved for the symbolic factorization and the other two are the memory required to perform an in-core and out-of-core factorization, respectively. If the memory available is not sufficient for doing an in-core factorization, LS-DYNA warns about this and attempts to do an out-of-core factorization, which means that the hard disk is used to store information during factorization.

At this point a redistribution of the system matrix is performed and a short report is given on this

```
symbolic storage        2 =           1038915
in-core numeric  storg  2 =          68238926
out-of-core num.  storg 2 =          11553502
```

Now the actual factorization takes place and a report from this is given where the information of interest is

```
CPU: factorization        =            59.025
WCT: factorization        =            50.151
act.  factor nonzeroes    =       5.1236E+07
act.  factor operations   =       1.2092E+11
act.  max.  factor nz.    =       5.1236E+07
act.  max.  factor op.    =       1.2092E+11
act.  total factor nz.    =       9.9666E+07
act.  total factor op.    =       2.3293E+11
```

which basically is the same information as from the symbolic factorization except that now it is the actual instead of the estimated values that is reported.

Finally, the forward and backward substitutions are allocated for and performed. The information given is

```
symbolic storage        3 =          1038915
in-core numeric storg   3 =         68299122
out-of-core num.  storg 3 =         16476693
CPU: numeric solve        =            0.191
WCT: numeric solve        =            0.194
```

More information is available with the command line options DIAG = 1 (memory) and IMPCON = 1 (time stamps).

Elements and Materials

Implicit Accuracy

Implicit and explicit analysis differ in many respects. An important one is that the deformation during a single step is much larger in implicit than a typical one in explicit. In the context of elements and materials, the demand for stronger objectivity and higher accuracy in implicit is obvious. The notion of implicit executing the same algorithms as a corresponding explicit analysis is therefore not adopted in general. This can optionally be further extended, using

IACC = 1 to increase implicit accuracy

on *CONTROL_ACCURACY will make selected elements strongly objective and enhance the accuracy for selected materials. For instance finite rotations will be represented exactly and fully iterative plasticity adopted. In addition, the flag applies to tied contacts as will be elaborated on below. Currently the following elements are supported for this option

Solid elements -2,-1,1,2

Shell elements 4,-16,16,23,24
Beam elements 1,2,9

and materials 24 and 123 use fully iterative plasticity.

Contacts

Contacts are probably among the hardest features to treat in a nonlinear implicit simulation. They are divided into the categories *tied* (bilateral) and *sliding* (unilateral) contact, and the characteristics of the two are quite different. Tied contacts are most often applied as constraints, only sometimes using a penalty formulation, and are fairly easy to deal with as they are only moderately nonlinear. In contrast, sliding contacts are exclusively implemented as penalty contacts and hard to deal with because of the unilateral condition. While the number of contacts, see *CONTACT, in LS-DYNA is abundant, we will here only present the few contacts that make most sense to use in implicit.

Tied Contacts

Tied contacts in implicit analysis should be accompanied with the implicit accuracy option. This restriction essentially reduces the number of relevant contacts to six. These six contacts are

*CONTACT_TIED_NODES_TO_SURFACE
*CONTACT_TIED_NODES_TO_SURFACE_CONSTRAINED_OFFSET
*CONTACT_TIED_NODES_TO_SURFACE_OFFSET
*CONTACT_TIED_SHELL_EDGE_TO_SURFACE
*CONTACT_TIED_SHELL_EDGE_TO_SURFACE_CONSTRAINED_OFFSET
*CONTACT_TIED_SHELL_EDGE_TO_SURFACE_BEAM_OFFSET

The first and fourth of these project the nodes on the tracked side to the reference surface, while the rest are offset contacts for which nodes will remain in place. The fourth, fifth and sixth constrain rotations while the others do not. The third and sixth are penalty-based, while the others are constraint-based. In sum, these six contacts cover most reasonable scenarios.

The implicit accuracy option will in this context make these six tied contact strongly objective, and built-in intelligence will detect whether nodes contain rotations to constrain or not. A nice feature with the latter is that torsion is automatically applied with physical consistency. Thus, it is no problem to use single beams to model spot welds between solid elements. When using the same connection technique between shells, the beam axial rotational degrees of freedom will be constrained to the translational degrees of the

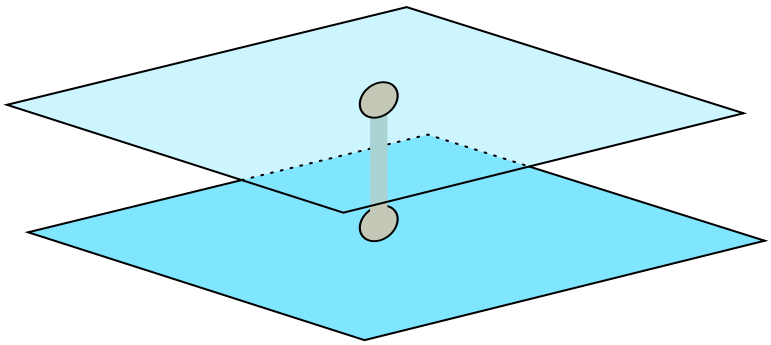


Figure 66-1. Modeling spotwelds by connecting a beam element to two shell elements.

shells, avoiding the relatively weak (and non-physical) drilling degree of freedom. This situation is depicted in Figure 66-1.

The choice of contact depends on the situation, but from a conceptual point of view it should be alright to use *CONTACT_TIED_SHELL_EDGE_TO_SURFACE_CONSTRAINED_OFFSET for most cases. The only problem then would be if complicated geometry results in termination from overconstraining, for which a switch to the corresponding penalty version *CONTACT_TIED_SHELL_EDGE_TO_SURFACE_BEAM_OFFSET is motivated. See General Remarks in the *CONTACT section of the LS-DYNA keyword manual for more information.

Sliding Contact

In this section we will discuss Mortar contact as this seems to be the best implicit contact algorithm when considering a combination of speed, accuracy and robustness. Mortar contact features smoothness and continuity that is highly appreciated in implicit analysis but is expensive enough to not be recommended for explicit analysis. Many other contact algorithms are supported for implicit analysis and execute faster than the Mortar contact, but this is often seen when IGAP flag is set to the default. This flag manipulates the stiffness matrix to the extent that accuracy may be deteriorated. If used, the results should be thoroughly checked. For Mortar contact IGAP has a different meaning as will be described IGAP. See General Remarks for *CONTACT_OPTION in the LS-DYNA keyword manual for more information.

Basics

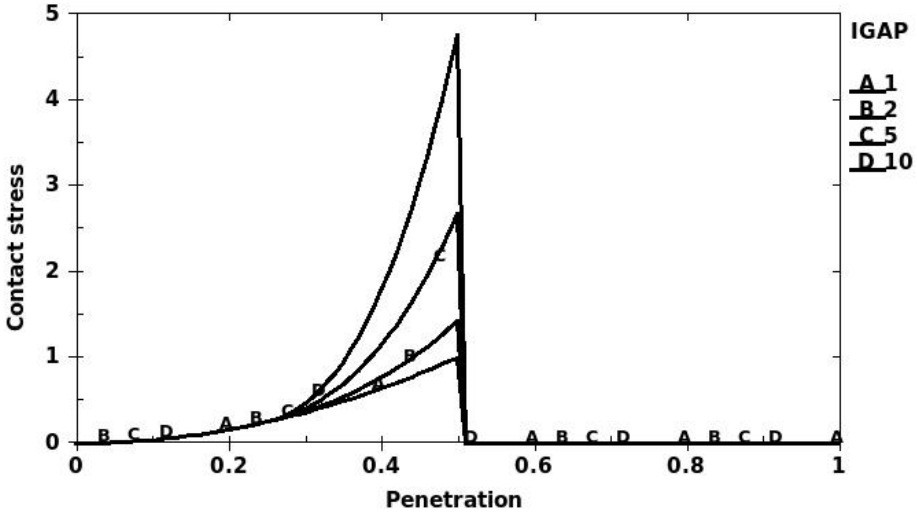
Mortar contact is activated by typically appending the suffix MORTAR to automatic single surface, automatic surface-to-surface or forming surface-to-surface keywords. It can also be run as tied and tiebreak contacts. All Mortar contacts are segment-to-segment and penalty-based. The tied and tiebreak contacts are always offset, meaning the tie occurs on the outer surfaces of shells and not on the mid-surfaces. For automatic contacts, edge contact with flat edges is always active. It possesses features that are of particular interest to implicit and are not available for other contacts. It is supported in both SMP and MPP, but the keyword option MPP does not apply. The SOFT flag does not apply. To summarize it is a contact algorithm especially intended for implicit analysis.

Recommendations

For Mortar forming contact the rigid tools must be meshed so that the normal vectors are directed towards the blank. Contacts from above and below the blank must be separated into two or more interfaces because contact can only occur from one side of the blank for a given contact interface. Rigid shells on the tracked side have no contact thickness. This is not the case for automatic Mortar contacts. For these contacts, there are no restrictions on the mesh and even rigid shells have contact thickness. For all Mortar contacts, part or part set based tracked and reference sides are recommended but not mandatory. If the two sides in the contact interface have different stiffnesses, use the weak part as the tracked side to get the best possible implicit convergence behavior. The single surface Mortar contact automatically takes care of this.

Characteristics

The contact pressure in Mortar contact is a parabolic function of the penetration in combination with a cubic stiffening phase. In short, the contact stiffness depends on the tracked side material and a characteristic length of the tracked side segment. The characteristic length for shells is the shell thickness and for solids is a median of the edges in the tracked side of the contact interface. The maximum penetration allowed is given as 95% of the average characteristic lengths on reference and tracked sides. For solids the definition of the characteristic length may cause the stiffness to become unnecessarily high if some elements are much smaller than most, so stiffness adjustments may be necessary. Furthermore, the characteristic length also determines the maximum penetration as well as the search radius for finding contact pairs; for this reason, the characteristic length can optionally be increased by assigning it on PENMAX on optional card B. In most cases it is expected that default value (= 0.0) for this parameter will suffice.



IGAP

The contact stiffness is parabolic with respect to penetration up to a penetration depth corresponding to half of the maximum penetration. For IGAP = 1 it will remain parabolic

for even larger penetrations, but the user may increase IGAP which means that the contact will stiffen for larger penetrations. In fact it will become cubic according to the picture above. The purpose of increasing IGAP could be to prevent the penetration from becoming larger than the maximum allowed penetration, because if convergence is attained with penetrations larger than this value this contact will be released in subsequent steps and the simulation is likely destroyed. Penetrations of this depth are likely to cause discontinuities along line searches and other discouraging phenomena. The user may of course scale the stiffness by increasing SFSA but this also scales the stiffness for small penetrations and probably has a negative effect on convergence.

Output for Debugging

Just as for implicit in general, information is always a good thing to have when convergence starts deteriorating and considering the release of contact in the previous section, it would be interesting to know if penetrations are large enough for this to be a potential danger. First, initial penetrations are always reported in the message file(s), including the maximum penetration and how initial penetrations are to be handled. The latter depends on the value of the IGNORE flag and this is dealt with below. In addition, by setting

MINFO = 1

on *CONTROL_OUTPUT, LS-DYNA will report the absolute maximum penetration as well as the maximum penetration in percentage after each equilibrium. If the relative maximum penetration reaches above 99% a warning message is printed as this particular contact is close to being released. The output is exemplified in the following.

```
     Contact sliding interface          1
     Number of contact pairs          527

     Maximum penetration is  0.2447797E+00 between
     elements     306774 and     306672

     Maximum relative penetration is  0.2266694E+02 % between
     elements     306742 and     306733

     Contact sliding interface          2
     Number of contact pairs        16209

     Maximum penetration is   0.5027643E+00 between
     elements     219492 and      94935

     Maximum relative penetration is  1.0366694E+02 % between
     elements     219492 and      94935
 *** Warning Penetration is close to maximum before release
```

This percentage value should ideally be kept below some 90% to have some sort of comfort margin. There are three ways to reduce maximum relative penetrations, and these are *(i)* to increase IGAP, *(ii)* to increase PENMAX for solids or *(iii)* to increase SFSA.

Initial Penetrations

As mentioned above, initial penetrations are always reported in the message file(s), including the maximum penetration and how initial penetrations are to be handled. The IGNORE flag governs the latter and the options are listed below.

IGNORE < 0	Same functionality as the corresponding absolute value but contact between segments belonging to the same part is ignored completely.
IGNORE = 0	Initial penetrations will give rise to initial contact stresses, meaning the tracked contact surface is not modified. *This is not available for Mortar but listed for the sake of completeness*.
IGNORE = 1	Initial penetrations will be tracked, that is, the tracked contact surface is translated to the level of the initial penetrations and subsequently follows the reference contact surface on separation until the unmodified level is reached.
IGNORE = 2	Initial penetrations will be ignored, meaning the tracked contact surface is translated to the level of the initial penetrations, optionally with an initial contact stress governed by MPAR1 (*this is the default*).
IGNORE = 3	Initial penetrations will be removed over time, that is, the tracked contact surface is translated to the level of the initial penetrations and pushed back to its unmodified level over a time determined by MPAR1.
IGNORE = 4	Same as IGNORE = 3, but it allows for large penetrations by also setting MPAR2 to at least the maximum initial penetration.

The use of IGNORE depends on the problem. If no initial penetrations are present, there is no need to use this parameter at all. If penetrations are relatively small in relation to the maximum allowed penetration, then IGNORE = 1 or IGNORE = 2 seems to be the appropriate choice. For IGNORE = 2 the user may specify an initial contact stress small enough to not significantly affect the physics, but large enough to eliminate rigid body modes and thus singularities in the stiffness matrix. The intention with this is to constrain loose parts that are initially close but not in contact by pushing out the contact surface using SFSAT and applying the IGNORE = 2 option. It is at least good for debugging problems with many singular rigid body modes. IGNORE = 3 is the Mortar interference counterpart, used for instance if there is a desire to fit a rubber component in a structure. With this option the contact surfaces are restored linearly in time from the beginning of the simulation to the time specified by MPAR1. A drawback with IGNORE = 3 is that initial penetration must be smaller than half the characteristic length of the contact or otherwise they will not be detected in the first place. For this reason, IGNORE = 4 was introduced where initial penetrations may be of arbitrary size, but it requires that the user provides crude information on the level of penetration of the contact interface. This is done in MPAR2 which must be larger than the maximum penetration or otherwise and error termination will occur. IGNORE = 4 only applies to solid elements at the moment.

When eliminating penetrations by simulation for models with many parts, some parts may contain thin members that cause spurious self-contacts. These may be difficult to work around by only adjusting contact parameters, but fortunately there is rarely any loss of generality in ignoring contact within parts completely since those are usually not of interest in such a context. The option IGNORE < 0 was implemented for this purpose and is a way to approach this problem.

Damping
Damping can be activated in dynamic implicit analysis using VDC. A problem with contact damping in implicit is that the time step is usually large enough to not resolve the time in contact to get the desired damping effect. Therefore, it is not recommended to use damping.

Troubleshooting Convergence Problems

Convergence of the nonlinear equilibrium iteration process presents one of the greatest challenges to using the implicit mode of LS-DYNA. At risk of repeating what has already been mentioned, below are some useful troubleshooting approaches.

Eigenvalue Analysis

Many convergence problems in static implicit analysis are caused by unconstrained rigid body modes. These are created when an insufficient number of constraints are applied to the model, or when individual model parts are left disconnected. Eigenvalue analysis is an excellent diagnostic tool to check for these problems, both initially and at critical points in the simulation. The procedure for performing an eigenvalue analysis was discussed above.

D3Iter Plot Database

To diagnose convergence trouble which develops in the middle of a simulation, get a picture of the deformed mesh. Adjust the d3plot output interval to produce an output state after every step leading up to the problematic time. An additional binary plot database named d3iter is available which shows the deformed mesh during each equilibrium iteration. This output is activated by

D3ITCTL = 1 to activate d3iter plot database

on *CONTROL_IMPLICIT_SOLUTION. View this database using LS-PrePost to detect abnormal displacements. The problem may become obvious, especially as deformation is magnified. If not, there is yet another flag to activate to get the residual forces into both this database as well as d3plot for fringing. Setting

RESPLT = 1 to get residual data to binary databases

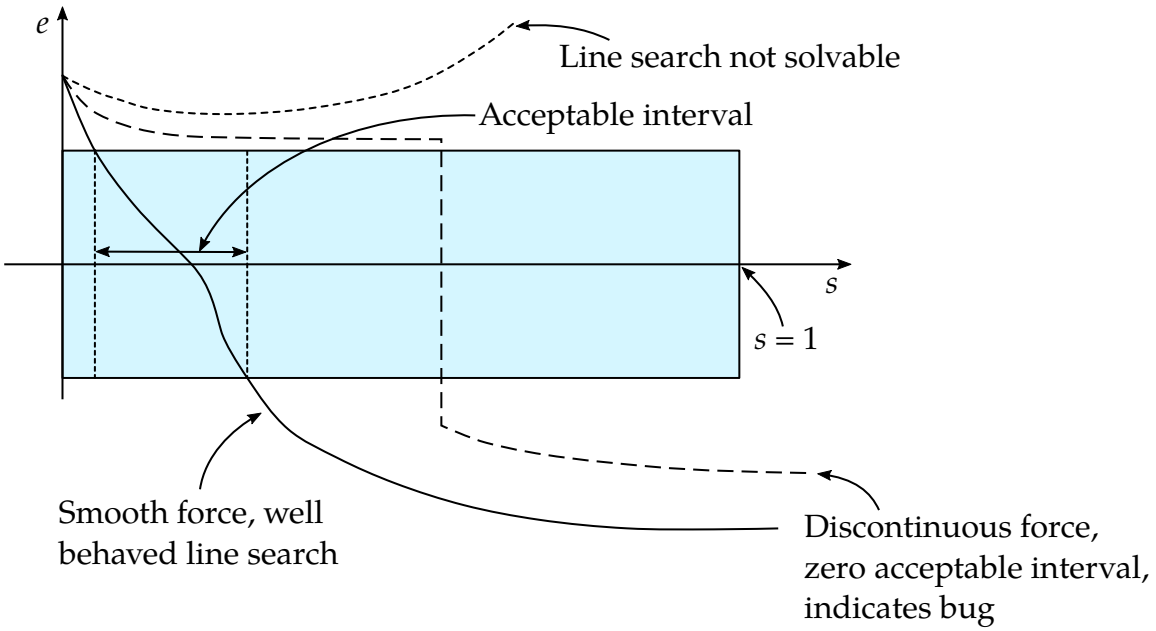
on *DATABASE_EXTENT_BINARY will do just that. With this option the residual forces are output to the d3plot and d3iter databases for fringing under the *NdV* menu. This is a great tool for locating areas in the model where the residual forces are not being reduced to a satisfactory level and take appropriate actions.

Taking Advantage of ASCII Information

If requested through NLPRINT = 3 on *CONTROL_IMPLICIT_SOLUTION and MINFO = 1 on *CONTROL_OUTPUT, a lot can be drawn from the information in the message file(s). This may be considered as a piece of advanced topic but may become useful in due time.

Residual Norm

Starting with the nonlinear output diagnostics, the following basic principle should be held in mind; *if the residual norm is zero, the problem is solved*. So it all comes down to make this number (11) small enough to trust the results. Therefore, we suggest monitoring the residual norm and interpreting it as an indicator of whether the convergence characteristics is good. Hopefully you would see this number decrease with implicit iterations and finally reach a relatively small number at convergence. Although it may (and will) increase on occasion, the trend should be a decreasing one. If this is a problem from the get-go, and you checked the model integrity through eigenvalue analysis and common practice, it may be that a feature is not properly supported in implicit.



Line Search

Another thing to look at is the line search convergence (9), as the relatively loose line search convergence tolerance should render a reasonably small number of line search iterations. A rule of thumb would be less than 10, at most 20. If more is used, or if the line search does not converge, there may be something in the model causing a jump in the residual forces. This is not supposed to happen in implicit and may suggest a bug.

APPENDIX P

Either of the two following scenarios for a feature discontinuity (and many line search iterations) is possible

- The line search step size becomes ridiculously small, and the current residual norm (7) is not approaching the initial one (3).
- The interval in which the optimal step is to be found (8) becomes ridiculously small, and the residual norm on the left and right interval points are not approaching each other. This situation is depicted in the figure above (dashed line).

Many line search iterations can of course be due to high nonlinearities but if the above is observed it should probably be investigated, for instance by trying to identify model features that may be causing the bad behavior. If it is obvious that a discontinuity exist, like if the step size goes down to ~1^{-100} or the interval size becomes zero, and it is not due to a model error, please send a bug report.

Stretching for Convergence

If the problem runs fine for a significant number of load/time steps with subsequent convergence issues, then the information received up to this point may be used intelligently and suggest a different (unorthodox) implicit strategy. To justify the approach, we begin by recalling that the convergence is detected by the following numbers becoming small; $d = \|\Delta \boldsymbol{u}\|$ (displacement norm), $r = \|\boldsymbol{R}\|$ (residual norm) and $e = \left|\boldsymbol{R}^T \Delta \boldsymbol{u}\right|$ (energy norm). But all these numbers are linked through the linear system of equations, $\boldsymbol{K}\Delta \boldsymbol{u} = \boldsymbol{R}$, so if $\boldsymbol{K}$ reasonably "nice" at all times it doesn't matter which numbers we use for detecting convergence. A mathematical way to state this is; if the condition number of $\boldsymbol{K}$ is "good," then all the norms are equivalent throughout the solution process. An intuitive statement would be to say that the problem is only moderately nonlinear, and convergence is usually never a problem. *But*, what happens if the properties of $\boldsymbol{K}$ are not that nice, or if $\boldsymbol{K}$ shifts character every time it is reformed? This is sort of saying that the problem is highly nonlinear, for instance due to frequent change of contact state or onsets/offsets of plasticity. In those cases the equivalence between d and r is lost, and typically an oscillatory behavior of the displacement norm is observed *even though the residual norm is reducing*, which in turn may lead to potential convergence problems just because the displacement criterion cannot be satisfied. Or equally bad, it could lead to premature convergence just by the coincidence that the displacement norm happens to become small. In those cases we essentially want to come up with a strategy where the displacement criterion is taken out of the convergence check and instead detect convergence based on residual only.

So assume we have say 10 converged states, after which the convergence problems begin. Then we can look at what the residual norm is for each of these converged states (11). For instance, we may see the sequence

Step 1, residual norm 2859
Step 2, residual norm 1581

Step 3, residual norm 2119
Step 4, residual norm 2511
Step 5, residual norm 11570
Step 6, residual norm 4904
Step 7, residual norm 3586
Step 8, residual norm 3157
Step 9, residual norm 3315
Step 10, residual norm 2825.

For each of these steps we may validate the results, by post-processing contact force curves, checking force/moment balance etc., in LS-PrePost, and usually there is a correlation between "good results" and small residual norms. Step 5 above is for instance an outlier and may be associated with a prematurely converged step, something that can be confirmed or rejected from looking at the result. The goal with these observations is to establish a reasonable residual norm for which we can safely say that the results are good, and the numbers above indicates that 3000 may be a good candidate. The strategy is then to simply put DCTOL = 10^{-16} to ignore the displacement and set ABSTOL = -3000, which means we can be assured that convergence will not be detected until the residual norm is below 3000. We "know" from having learned about the problem that this will yield good results. It should be mentioned that this may not work if the character (force level) changes significantly later on in the simulation, as 3000 then may not be the number we would trust.

Contacts
Continuing with the contact output, a general rule is that there should be *no* warnings of large penetrations and preferably the maximum relative penetration should be below 90%. These numbers can be monitored after each load/time step. If large penetrations occur, either increase IGAP or contact stiffness, whatever makes most sense, but first make sure that the converged step is *really* converged (no premature convergence) by monitoring the residual forces and checking results as indicated above.

Checklist

So, to summarize

- Activate implicit by setting IMFLAG = 1 on *CONTROL_IMPLICIT_GENERAL
    - Set DT0 to a reasonable initial time step
- Check possible singularities in an eigenvalue analysis by requesting NEIG eigenvalues on *CONTROL_IMPLICIT_EIGENVALUE
    - Find a way to constrain spurious modes

- Initially use dynamic analysis if rigid body modes are present in a static problem by setting IMASS = 1 on *CONTROL_IMPLICIT_DYNAMICS
    - Use TDYBIR, TDYDTH and TDYBUR
    - Use numerical damping by putting GAMMA = 0.6 and BETA = 0.38
- Only activate geometric stiffness contribution as a last resort, except for arc length methods
- Use default BFGS parameters ILIMIT and MAXREF on *CONTROL_IMPLICIT_-SOLUTION
    - Increase or decrease ILIMIT based on convergence characteristics
    - Use full Newton (ILIMIT = 1) for hard problems
    - Keep MAXREF to a reasonably low number
- Use default convergence tolerances DCTOL, ECTOL and RCTOL on *CONTROL_IMPLICIT_SOLUTION
    - ABSTOL may be set to 10^{-20} to prevent premature convergence
    - Relaxing DCTOL may not necessarily result in better convergence
    - Future strategy may be to focus on residual
- Activate automatic time stepping on *CONTROL_IMPLICIT_AUTO
    - Set DTMAX to a number that resolves features of interest
        - Contacts
        - Material nonlinearities
        - Frequencies
- Use default line search method on *CONTROL_IMPLICIT_SOLUTION
    - Switch to LSMTD = 5 if hard problem (typically rubbers)
    - Don’t change LSTOL
- Use at least NLPRINT = 2 on *CONTROL_IMPLICIT_SOLUTION to get convergence diagnostics into log files
    - Use NLPRINT = 3 to thoroughly track the residual norm and monitor line search behavior if debugging model is necessary
- Use *CONTROL_IMPLICIT_SOLVER only in special occasions
- Set D3ITCTL = 1 on *CONTROL_IMPLICIT_SOLUTION to get a database with Newton iterates when debugging a model

    - Complement this with RESPLOT = 1 on *DATABASE_EXTENT_BINARY to get the possibility to fringe the residual force vector

- For forming Mortar contact, make sure

    - Tools are oriented towards the blank
    - Contact on top and bottom of blank are separated among contact interfaces

- Use part (set) definitions of SURFA and SURFB in Mortar contact
- Always use weak part as SURFA (tracked) in a Mortar contact definition to get best possible convergence
- Set SAST to a reasonable characteristic length for SURFA side consisting of solid elements

    - With this option, separate solids and shells into different contact interfaces in order to not manipulate the contact thickness for shells

- If penetrations are large, activate penetration diagnostics on *CONTROL_CONTACT
- To avoid release of contact pairs, either

    - Increase stiffness scaling factor SFSA
    - Increase IGAP for progressive stiffness increase
    - Increase SAST for solids while at the same time increasing SFSA

- Use IGNORE appropriately to deal with initial penetrations

    - Check initial penetrations in messag file

- Don’t use contact damping