Lecture 10

Managing Large Models

Overview

- Introduction
- Simplifying the Model
- Parallel Execution
- Techniques for Reducing CPU Time
- Tips
- Submodeling
- Output Filtering
- Restart
- Parts and Assemblies

Introduction

Introduction

- **What is a Large Model?**
  - Difficult to define
    - Absolute size changes as computers become more powerful.
  - Rough measures for large models based on number of variables:
    - 1980: Models with >100,000 variables
    - 1990: Models with >1,000,000 variables
    - 2000: Models with >5,000,000 variables
  - Turn-around time is a universal measure
    - Models that take at least 12 hours (overnight) to run are generally considered “large.”

Introduction

- **Large models often require:**
  - Simplification of the model
    - geometry, physics, element formulation, etc.
  - Intelligent use of system resources:
    - disk space, available processors, memory
  - Minimization of CPU time, number of increments, etc.
- **Several techniques are available to assist in the management of large models:**
  - Submodeling
  - History output filtering
  - Restart
  - Parts and assemblies

ABAQUS/Explicit: Advanced Topics

ABAQUS

Simplifying the Model

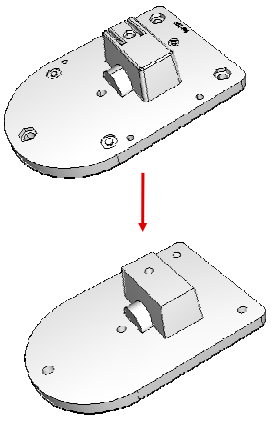
Copyright 2005 ABAQUS, Inc.

Simplifying the Model

- **There are a number of techniques that can be used to simplify a model:**
  - Geometry simplification
  - Modeling fewer details
  - Rigid body prototyping
  - Solid element formulation
  - Small strain shells

Simplifying the Model

- **Simplify geometry**
  - When building a complex model, start simple (and small), then add complexity gradually.
  - Making careful choices regarding your geometry and mesh:
    - Take advantage of symmetry
      - axial, planar, quarter, half, cyclic,…
    - Defeature the geometry
    - Avoid small features
      - They result in small elements which in turn reduce the stable time increment.



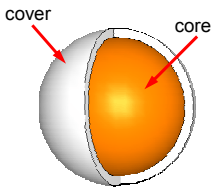
Defeaturing a mechanical component

Simplifying the Model

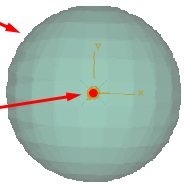
- **Model fewer details**
  - Consider what you want from the analysis and reduce the level of modeling detail in regions that are less critical to the analysis goals.
  - For example:
    - If you have two parts connected to each other and if you are not interested in detailed stresses of the connection, use connector elements.
    - Use beams or shells instead of solids when appropriate.
    - Focus mesh in regions of interest and use a coarser mesh elsewhere.
    - Simplify the physics.
      - Can a stiff portion of an assembly be replaced with a rigid body or a boundary condition?
      - Do you need all interactions?

Simplifying the Model

- **Example: Simplified model of a sports ball for use in an impact analysis.**
  - The ball consists of a solid core and a shell cover.
    - The detailed behavior of the ball is not of interest.
  - The ball is reduced to the following:
    - Analytical rigid surface for ball exterior contact surface
    - Lumped mass and rotary inertia for core
    - Lumped mass and rotary inertia for cover
    - Connector elements for stiffness and damping between:
      - Core and cover
      - Cover and rigid exterior surface



Schematic of modeled ball with a quarter of the cover cutaway



Ball Model

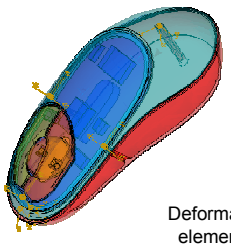
Simplifying the Model

- **Prototype with rigid bodies**
  - Make deformable parts rigid with a rigid body constraint for faster initial runs of the analysis.
  - Once you have verified the model definition, remove the rigid body constraints.

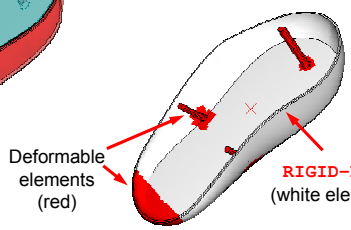
Simplifying the Model

- **Example: Drop Test Simulation of a Cordless Mouse**
  - Rigid body constraints are used so that initial versions of the analyses run quickly.

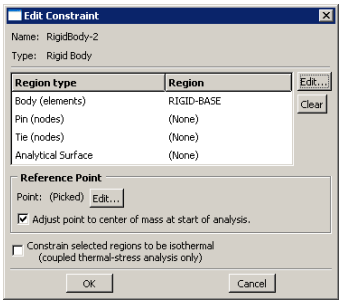
```
*RIGID BODY, ELSET=RIGID-BASE,
 REF NODE=BASE_RP, POSITION=CENTER OF MASS
```



mouse

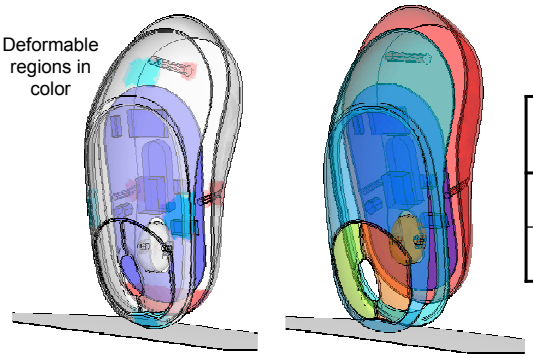


mouse base



Simplifying the Model

- **Example (cont'd): Drop Test Simulation of a Cordless Mouse**



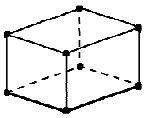
Mouse type	Portion of mouse that is deformable	Relative CPU time
rigid prototype	23%	0.35
fully deformable	100%	1

Video Clip

final deformed shapes

Simplifying the Model

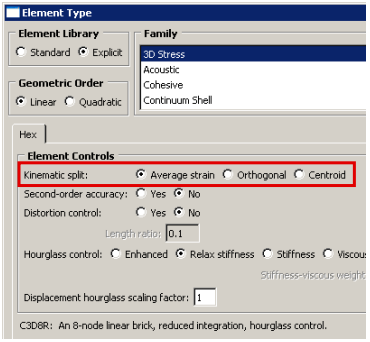
- **Solid element formulation**
  - ABAQUS/Explicit offers three alternative kinematic formulations for the C3D8R solid element:
    - The average strain formulation (default)
      - Most expensive; highest level of accuracy
    - The orthogonal formulation
      - Provides a balance between computational speed and accuracy
        - i.e., less expensive than the average strain formulation and more accurate than the centroid formulation
    - The centroid formulation
      - Least expensive; lowest level of accuracy



C3D8R

Simplifying the Model

- **Average strain kinematic formulation for hexahedral solid elements**
  - The default average strain kinematic formulation for solid elements is based on the uniform strain operator and orthogonal hourglass shape vectors.
  - The resulting elements:
    - Pass the constant strain patch test for a general configuration.
    - Have zero strain under large rigid body rotation.
  - The default kinematic formulation can be very expensive, especially for three-dimensional problems.



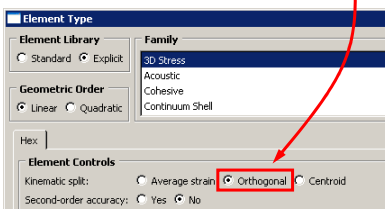
Simplifying the Model

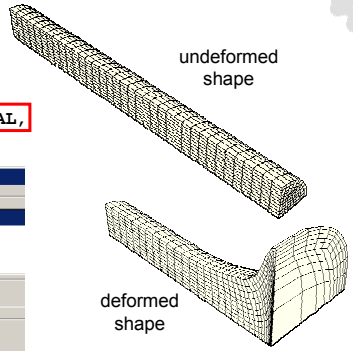
- **Orthogonal kinematic formulation for hexahedral solid elements**
  - The orthogonal formulation is based on the centroidal strain operator and a slight modification to the hourglass shape vectors.
    - The centroidal strain operator requires 3× fewer floating point operations than the uniform strain operator.
  - These elements pass the patch test only for rectangular or parallelepiped element configurations.
    - However, they converge well for general element configurations as the mesh is refined.
  - This formulation is recommended for all analyses **except** those involving:
    - highly distorted elements,
    - very coarse meshes, or
    - high confinement.

Simplifying the Model

- **Example: Copper rod impact**

```
*SOLID SECTION, ELSET=ROD, MATERIAL=COPPER,
  CONTROL=ORTHOGONAL
*SECTION CONTROLS, KINEMATIC SPLIT=ORTHOGONAL,
  HOURGLASS=COMBINED, NAME=ORTHOGONAL
```





C3D8R formulation	Shortening (mm)	Widening (mm)	Relative CPU time
default (average strain)	–13.08	5.473	1
orthogonal kinematics	–13.09	5.472	0.63

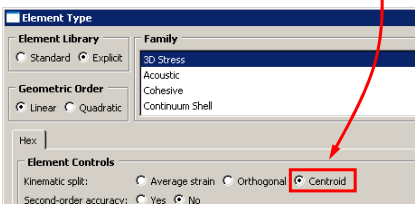
Simplifying the Model

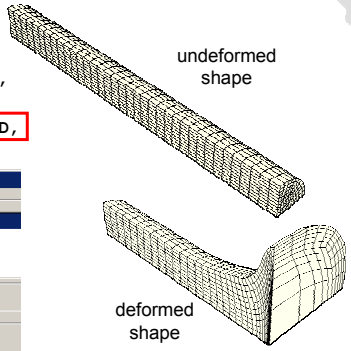
- **Centroid kinematic formulation for hexahedral solid elements**
  - The centroid formulation is the most economical.
  - The centroid formulation is based on the centroidal strain operator and simple hourglass base vectors.
    - Using the simple hourglass base vectors instead of the orthogonal hourglass shape vectors reduces hourglass mode computations by a factor of three.
    - The hourglass base vectors are not orthogonal to rigid body rotation for general element configurations.
      - Therefore, hourglass strain may be generated with large rigid body rotations with this formulation.
  - This formulation is recommended only for problems with little rigid body rotation and reasonable mesh refinement.

Simplifying the Model

- **Example (cont'd): Copper rod impact**

```
*SOLID SECTION, ELSET=ROD, MATERIAL=COPPER,
 CONTROL=ORTHOGONAL
*SECTION CONTROLS, KINEMATIC SPLIT=CENTROID,
 HOURGLASS=COMBINED, NAME=ORTHOGONAL
```

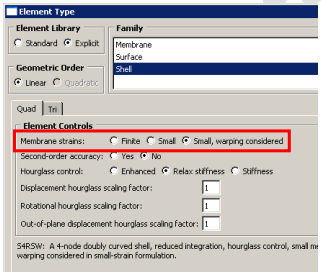


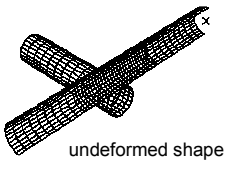


C3D8R formulation	Shortening (mm)	Widening (mm)	Relative CPU time
default (average strain)	–13.08	5.473	1
orthogonal kinematics	–13.09	5.472	0.63
centroidal kinematics	–13.07	5.231	0.49

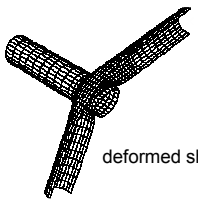
Simplifying the Model

- **Small strain shell elements**
  - Small strain shell elements were discussed in Lecture 2, *Elements*.
  - Small strain shell elements are more computationally efficient than their large strain counterparts.
  - They are appropriate for analyses involving small membrane strains and arbitrarily large rotations





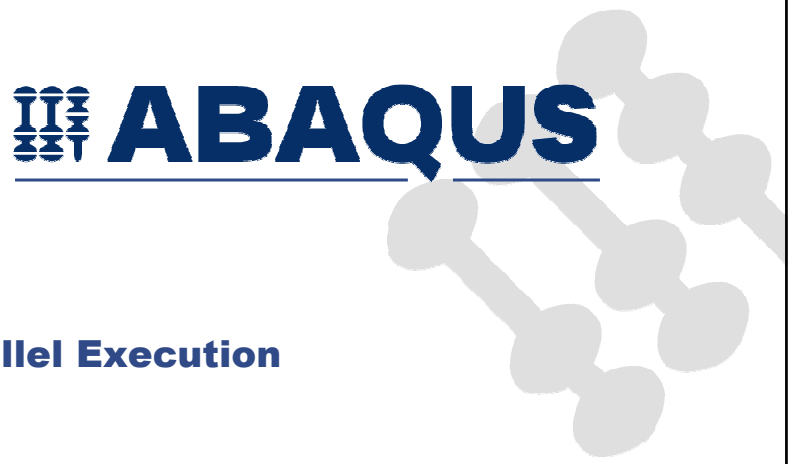
undeformed shape



deformed shape

Element type	Relative CPU time
S4R	**1.0**
S4RSW	**0.78**
S4RS	**0.66**

Pipe whip simulation analyzed with various shell element types



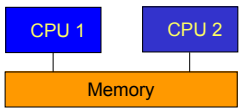
Parallel Execution

Parallel Execution

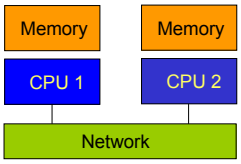
- **Introduction**
  - Parallel execution:
    - reduces run time for analyses that require a large number of increments and/or contain a large number of nodes and elements
    - produces analysis results that are independent of the number of processors used for the analysis
- **Parallelization in ABAQUS/Explicit is available for:**
  - shared memory architecture platforms using a thread-based loop-level or thread-based domain-level decomposition implementation
  - both shared and distributed memory architecture platforms using an MPI-based domain decomposition parallel implementation

Parallel Execution

- **Shared memory**
  - Separate processors share data on the same memory space
    - i.e., these are multiple processors of a single machine
  - Data shared via pointers (thread-based) or passed via an interface (MPI)
- **Distributed memory**
  - Processors have independent memory space
    - i.e., these are multiple processors on separate machines (clusters).
  - Data passed via an interface (MPI)



shared memory architecture



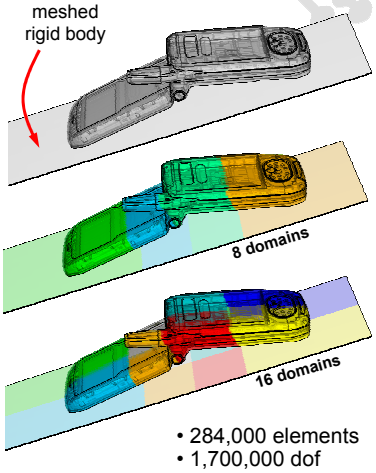
distributed memory architecture (clusters)

Parallel Execution

- **Communication between processors**
  - The MPI (Message Passing Interface) mode:
    - is available for both shared and distributed memory architecture platforms
      - The MPI facilitates messaging between processors
    - is not available for Windows platforms
    - requires that the MPI components be installed
  - The thread-based mode:
    - is available only on shared memory architecture platforms
      - Data are shared between processors via pointers
    - is available on all supported systems

Parallel Execution

- **Domain-level decomposition method**
  - With this method ABAQUS automatically breaks the model up into topological domains and assigns each domain to a processor.
  - Domains are distributed evenly among the available processors.
  - The analysis is then carried out independently in each domain.
    - However, information must be passed between the domains in each increment because the domains share common boundaries.



- 284,000 elements
- 1,700,000 dof
- 12 ms simulation

Courtesy of Motorola

Parallel Execution

- **Domain-level decomposition method (cont'd)**
  - The domain-level method is the default.
    - It is recommend for most large analyses.
    - The domain-level speedup factor may be significantly more than what can be achieved with loop-level parallelization (discussed later).
  - Both MPI- and thread-based communication schemes are supported with the domain-level method.
    - The MPI mode is required for distributed memory systems (clusters).
  - During the analysis, separate output database (`.odb`), selected results (`.sel`), and state (`.abq`) files are created for each processor.
    - At the end of the analysis the individual files are merged automatically.
  - There are some restrictions when using the domain-level method.
    - Refer to the manuals for more information.

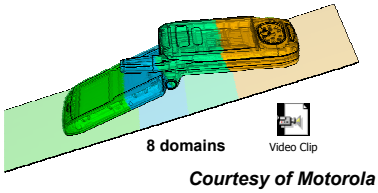
Parallel Execution

- **Domain-level decomposition method (cont'd)**
  - Execution procedure (at the command line):

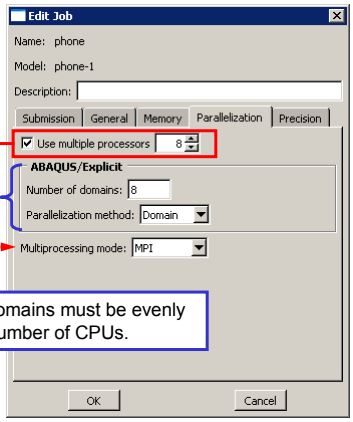
Required — `abaqus job=phone cpus=8`

Defaults — `parallel=domain domains=8`, `mp_mode=mpi`

Defaults can be modified using `abaqus_v6.env`.
On Windows the thread-based parallelization method is the default and only option (`mp_mode=threads`).



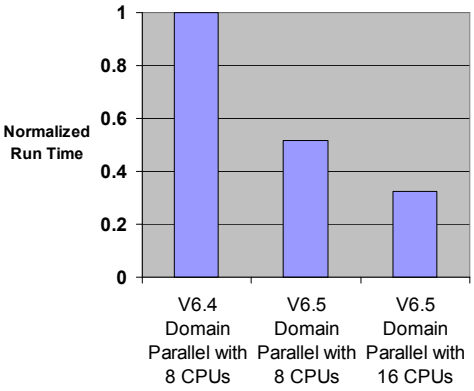
Courtesy of Motorola

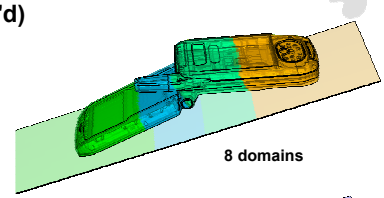


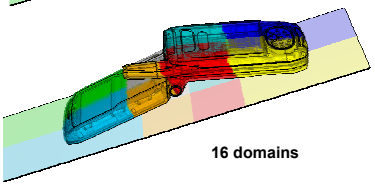
The number of domains must be evenly divisible by the number of CPUs.

Parallel Execution

- **Domain-level decomposition method (cont'd)**
  - Run times normalized relative to Version 6.4 domain parallel with 8 CPUs







- 284,000 elements
- 1,700,000 dof
- 12 ms simulation

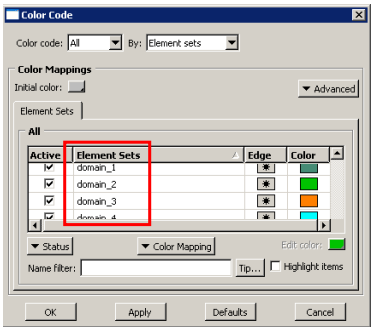
Courtesy of Motorola

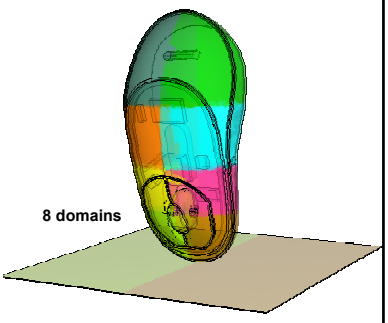
Parallel Execution

- **Domain-level decomposition method (cont'd)**
  - When a parallel analysis is in progress, the separate output database files (*job-name*.odb.*n*) need to be combined before they can be post-processed.

```
abaqus oldjob=job-name job=combined-odb-name convert=odb
```

  - ABAQUS automatically creates element and node sets for each domain.

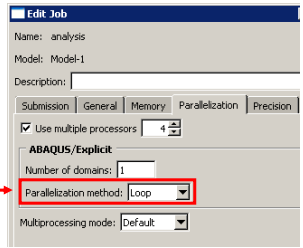




Parallel Execution

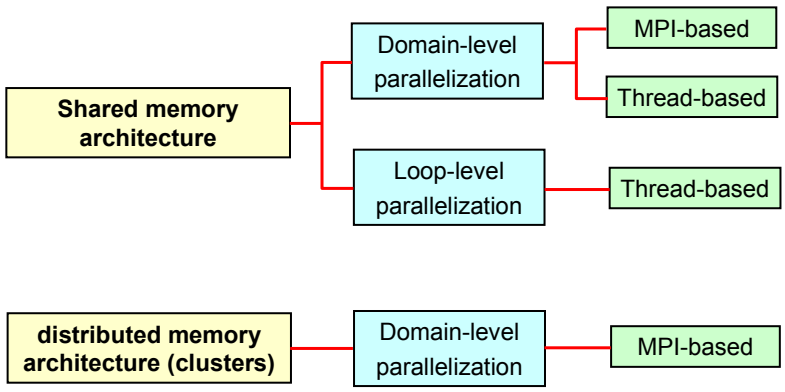
- **Loop-level method**
  - The loop-level method parallelizes low-level loops that are responsible for most of the computational cost.
    - The element, node, and contact pair operations account for the majority of the low-level parallelized routines.
  - There are no restrictions when running with this method;
    - however, the speedup factor may be significantly less than what can be achieved with domain-level parallelization.
      - The speedup factor will vary depending on the features included in the analysis.
  - Execution procedure (at the command line):

```
abaqus job =job-name cpus=n parallel=loop
```



Parallel Execution

- **Parallelization options summary for ABAQUS/Explicit :**



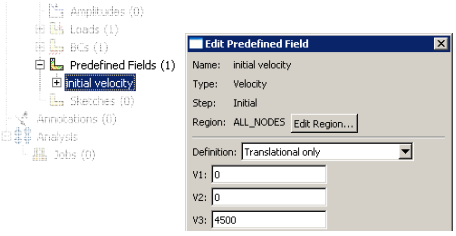
Techniques for Reducing CPU Time

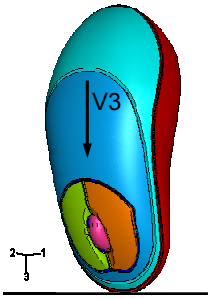
Techniques for Reducing CPU Time

- **Features that can be used to reduce analysis run time include:**
  - Initial conditions
  - Static initialization and import
  - Monitoring for extreme values of output variables
  - Monitoring for steady state
  - Mass scaling
  - Increasing load rates

Techniques for Reducing CPU Time

- **Initial conditions can be used to reduce the length of some analyses.**
  - You can define initial conditions (stress fields, hardening conditions, velocities,…) rather than modeling the history that lead to the initial conditions.
  - Example: Drop test of a cordless mouse
    - Apply initial velocity conditions consistent with a 1m free fall.



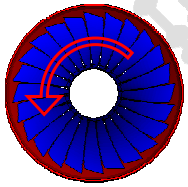


Mouse initial velocity

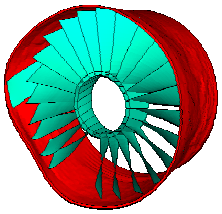
```
*INITIAL CONDITIONS, TYPE=VELOCITY
 ALL_NODES, 3, 4500
```

Techniques for Reducing CPU Time

- **Static initialization and import**
  - ABAQUS provides a capability to transfer the deformed mesh and associated material state between an ABAQUS/Standard analysis and an ABAQUS/Explicit analysis (and vice-versa).
  - Reduce the analysis run time by solving portions of an analysis with ABAQUS/Standard.
  - Examples:
    - Simulate a transient event in ABAQUS/Explicit after steady-state conditions have been simulated in ABAQUS/Standard
    - Simulate springback in ABAQUS/Standard after forming analysis in ABAQUS/Explicit
  - This functionality is discussed in Lecture 11, *ABAQUS/Explicit–ABAQUS/Standard Interface*.



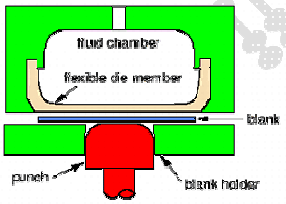
steady-state spinning solution in ABAQUS/Standard



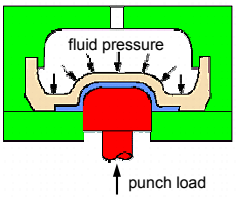
dynamic blade release analysis in ABAQUS/Explicit

Techniques for Reducing CPU Time

- **Monitoring for extreme values of output variables**
  - Monitoring the values of critical variables and halting the analysis when those variables exceed a given criterion can increase modeling efficiency.
    - For example, in a hydro-forming simulation the time needed to model the completion of the physical process may be unknown.
      - It may depend on the magnitude of the displacement of a node or a group of nodes in the model.
    - Terminating the analysis when that displacement is reached may save many CPU hours.



(a) Typical hydroforming configuration



(b) Forming a part

Techniques for Reducing CPU Time

- Element integration point variables, element section variables, or nodal variables can be monitored.
  - The variable values are checked in every increment.
  - Only scalar quantities can be monitored.
    - Tensors and vectors cannot be monitored; only their individual components can be monitored.
  - The variables can be monitored for either minimum, maximum, or absolute maximum values.
- At the first occurrence of a variable exceeding the user-specified bounds, the variable name, the associated element or node number, and the increment number are written to the status (`.sta`) file.
  - You can also request that:
    - the analysis be stopped
      and/or
    - the output state be written in the increment following the one in which the monitored variable exceeded the user-specified bound.

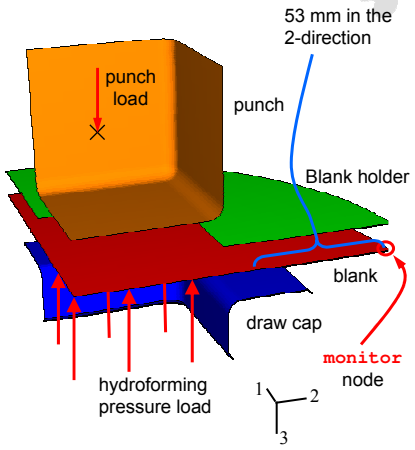
Techniques for Reducing CPU Time

- Example: Hydroforming of a box

terminate the analysis when the extreme value is reached

```
*EXTREME VALUE, HALT=YES
*EXTREME NODE VALUE, NSET=monitor, MIN
U2, -53
```

Monitor node `monitor` for the time when displacement in the 2 direction is less than -53

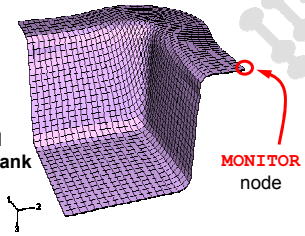


Exploded view of initial configuration

- These options are not currently supported by ABAQUS/CAE
  - They can be included using the Keywords Editor.

Techniques for Reducing CPU Time

– Example (cont'd): Hydroforming of a box



The end of the the status (`.sta`) file:

```
  Variable U2  violated the extreme value criterion at node 2 at
  the end of increment 26079. The current value of the variable is
 -53.001 in nset ASSEMBLY_MONITOR

           EXTREME VALUE MONITORING INFORMATION FOR STEP   2
-------------------------------------------------------------------------
VARIABLE   ELEMENT SET/  ELEMENT/NODE    EXTREME       GLOBAL TIME
           NODE SET      LABEL           VALUE         OCCURRED(STEP)
-------------------------------------------------------------------------

U2      ASSEMBLY_MONITOR    2           -53.003       5.042E-03(  2)
-------------------------------------------------------------------------
A special ODB Field Frame for extreme value output written at  4.992323E-03

  THE ANALYSIS HAS COMPLETED SUCCESSFULLY
```


Techniques for Reducing CPU Time

- **Monitoring for steady state**
  - Steady-state detection can be used to terminate an ABAQUS/Explicit analysis when specified steady-state criteria are met.
    - This feature is available for quasi-static uni-directional processes.
      - For example rolling, wire drawing, and extrusion processes
    - The following steady-state detection norms are available for the specification of steady-state detection criteria:
      - Normalized cross-section plastic strain at a specified cutting plane
      - Largest of the area moments of inertia of the plane cross section
      - Average force magnitude at a rigid body reference node
      - Average torque magnitude at a rigid body reference node
  - Terminating the analysis when steady state is reached may save many CPU hours.

Techniques for Reducing CPU Time

– Steady-state detection was used to terminate the Eulerian flat rolling simulation described in Lecture 6, *Adaptive Meshing and Distortion Control*:

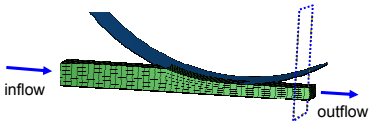
Edit keywords, Model: Flat Rolling

```
*Step, name=Step-1
*Dynamic, Explicit
, 0.5
*STEADY STATE DETECTION,ELSET=BAR,SAMPLING=UNIFORM
1.0, 0., 0., .1, 0.0, 0.0
*STEADY STATE CRITERIA
SSPEEQ, , .0409, 0., 0.
SSSPRD, , .0409, 0., 0.
SSTORQ, , .0409, 0., 0., Assembly.Roller.10001, 0., 0., 1.
SSFORC, , .0409, 0., 0., Assembly.Roller.10001, 0., 1., 0.
```

Request sampling at uniform intervals for an Eulerian adaptive meshing analysis.

Cutting plane
Analysis terminates when the steady state is detected at the cutting plane.

steady-state criteria definition
Only when all of the criteria specified have been satisfied will the analysis be considered to have reached steady state.

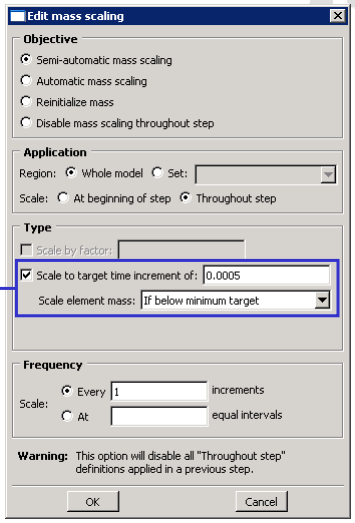


Techniques for Reducing CPU Time

- **Mass scaling speeds up an analysis by increasing the stable time increment size.**
  - Artificially increasing the material density by a factor of f^2 increases the stable time increment by a factor of f.
  - Mass scaling for quasi-static analyses was discussed in Lecture 5.
  - Mass scaling for high-speed dynamic events was discussed in Lecture 9.

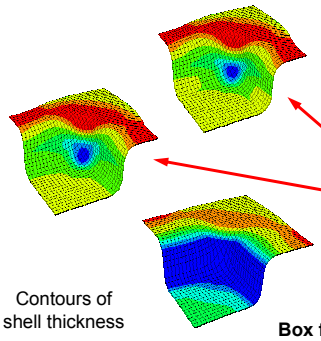
```
*VARIABLE MASS SCALING, FREQUENCY=1,
DT=0.0005, TYPE=BELOW MIN
```

Scale density of elements whose stable time increment is less than .0005 s



Techniques for Reducing CPU Time

- **Increasing load rates to speed up a quasi-static analysis**
  - Artificially reduce the time scale of a quasi-static process by increasing the loading rate.
  - The details of this technique were discussed in Lecture 5.



Contours of shell thickness

Punch speed (m/s)	Time increments	Normalize CPU time
3 (1×)	27929	1.0
30 (10×)	2704	0.097
150 (50×)	529	0.019

Box forming example with various load rates

ABAQUS

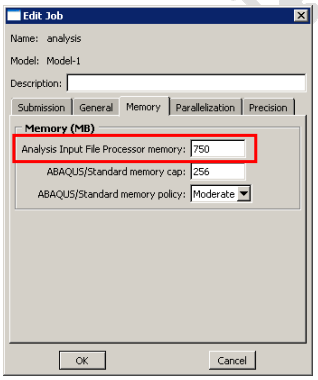
Tips

Tips

- **If a large model size is unavoidable, input file processing may require more than the maximum amount of memory allowed by default.**
  - Use the following table as a rough guideline.

Degrees Of Freedom	pre_memory
Up to 250,000	250 MB
Up to 1,000,000	750 MB
Up to 5,000,000	2000 MB

  - Set values only as necessary, otherwise ABAQUS may take more memory than it needs.
  - If input processing detects that the memory specified is insufficient, ABAQUS will issue an error.



Environment File: `abaqus_v6.env`

```
pre_memory="750 mb"
```


Tips

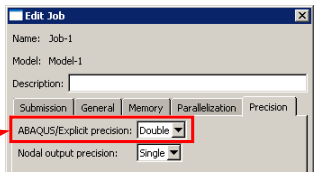
- **Review results during a long ABAQUS/Explicit analysis to avoid wasting resources running an incorrect model.**
  - When reviewing an output database before an analysis is complete, you will need to run the postprocessing calculator to:
    - combine the multiple output database files created when domain-level parallelization is used
    - obtain history results for tracer particles
    - extrapolate of integration point quantities to the nodes or centroid of an element.
  - Execution procedure (at the command line):

```
abaqus oldjob=job-name job=converted-odb-name convert=odb
```

Tips

- **Analyses that require a large number of increments should be solved with the double-precision executable.**
  - Typically single-precision (32-bit word lengths) results in a CPU savings of 20% to 30% compared to the double-precision executable (64-bit word lengths).
  - Single precision provides accurate results in most cases.
    - Exceptions for which double-precision is recommended:
      - analyses of more than 300,000 increments
      - when typical nodal displacement increments are less than 10^{-6} times nodal coordinate values.
      - models with hyperelastic materials
      - analyses with multiple revolutions of deformable parts
  - Execution procedure (at the command line):

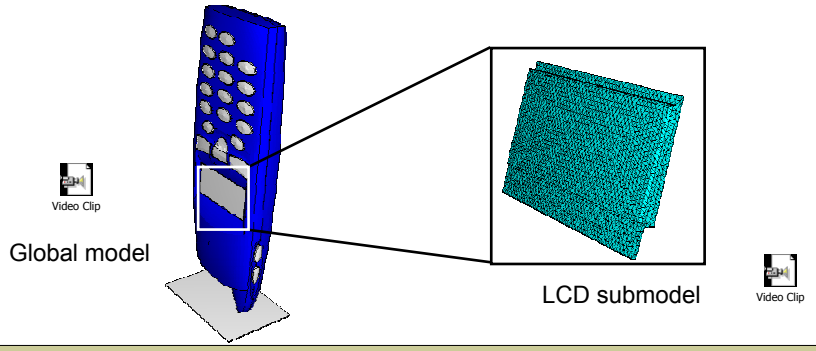
    abaqus **job=***job-name* **double**



Submodeling

Submodeling

- **Introduction**
  - Submodeling allows the analyst to focus on a subset of a model based on an existing solution from a global model.
  - For example submodeling can be used to perform a detailed analysis of a cell phone display based on the results of a phone drop test.

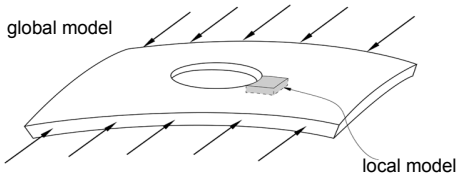


Submodeling

- **Motivation**
  - An initial global analysis of a structure usually identifies critical regions.
    - Submodeling provides an easy modeling enhancement of these areas without having to resort to remeshing and reanalyzing the entire model.
  - Submodeling can reduce analysis costs while providing detailed results in local regions of interest.
    - Use a coarse model to obtain the solution around the local region of interest.
    - Use a refined mesh of the local region only for the local analysis.
  - Further refinement is possible if desired, since submodeling can be used through several levels;
    - the local model for one level can be used as the global model for the next.

Submodeling

- **Basic procedure**
  - Obtain a global solution of the whole model using a level of mesh refinement that allows acceptable solution accuracy.
  - Interpolate this solution onto the boundary of a locally refined mesh.
  - Obtain a detailed solution in the local area of interest.
  - The solution from the global model is interpolated automatically as needed to “drive” the displacements on the boundary of the local model.



Submodeling

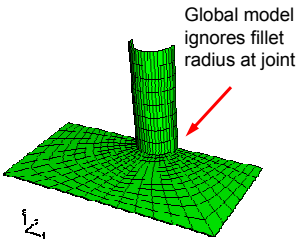
- **Location of the submodel boundary**
  - Deciding the actual location of the local boundary requires some engineering judgment.
  - The submodel boundary should be far enough away from the area of the submodel where the response is changed by the different modeling.
    - That is, the solution at the boundary should not be altered significantly by the different local modeling (Saint-Venant's Principle).

Submodeling

- **Submodeling can be applied quite generally in ABAQUS**
  - The material response defined for the submodel may be different from that defined for the global model.
  - The global and submodel procedures do not have to be the same.
    - For example, the nonlinear dynamic response of the global model can be used to drive the static, nonlinear response of the submodel on the grounds that the submodel is too small for dynamic effects to be significant in that local region.
  - The global procedure can be performed in ABAQUS/Standard to drive a submodeling procedure in ABAQUS/Explicit and vice versa.
    - For example, a static analysis performed in ABAQUS/Standard can drive a quasi-static ABAQUS/Explicit analysis in the submodel.
  - The step time used in the analyses can be different; the time variation of the driven nodes can be scaled to the step time used in the submodel.

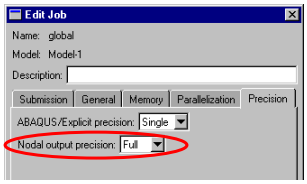
Submodeling

- **Example: Shell-to-Solid Submodel of a Pipe Joint**
  - Global model
    - Joint between an aluminum pipe and a plate.
    - Pipe is loaded with 10 N in the 1–direction at the free end.
    - The edges of the plate are clamped.
    - Nodal displacements written to the output database file (`.odb`) will be used for interpolation to the driven nodes in the submodel.



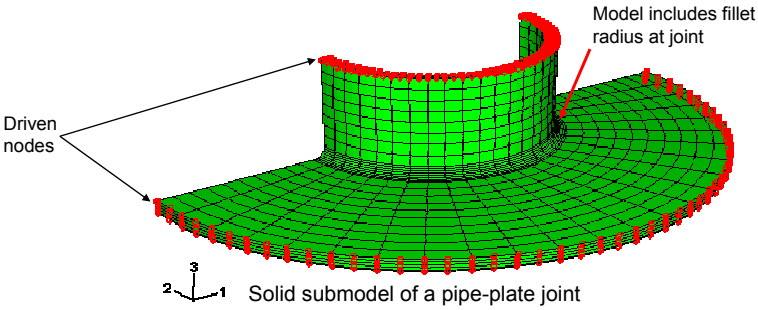
Global shell model (S4R) of pipe-plate structure

```
abaqus job=coarse
 output_precision=full
```



Submodeling

- **Example (cont'd): Shell-to-Solid Submodel of a Pipe Joint**
  - Submodel
    - Made up of C3D20R elements.
    - Four elements are used through the thickness.
    - Nodes are driven along two surfaces of the submodel, one in the pipe and the other in the plate

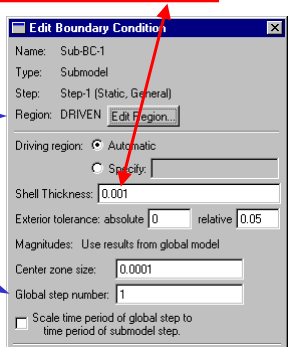


Submodeling

- **Example (cont'd): Shell-to-Solid Submodel of a Pipe Joint**
  - Input for submodel:

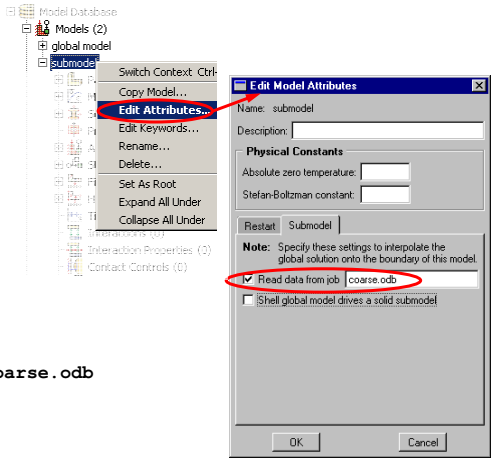
Maximum shell thickness in plate

```
*HEADING
:
*SUBMODEL, SHELL TO SOLID, SHELL THICKNESS=0.001
 DRIVEN,
*STEP
:
*BOUNDARY, SUBMODEL, STEP=1
 DRIVEN,
*END STEP
```



Submodeling

- **Example (cont'd): Shell-to-Solid Submodel of a Pipe Joint**
  - Associate the global model with the submodel.

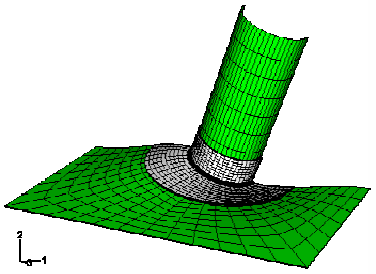


At the command prompt:

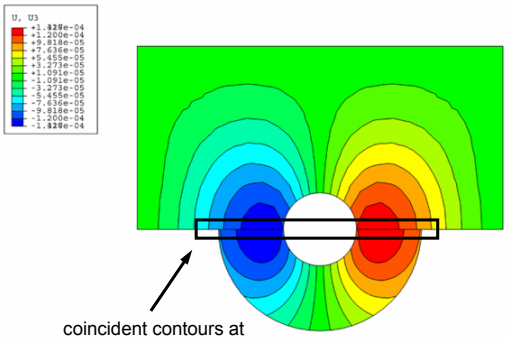
```
abaqus job=submodel globalmodel=coarse.odb
```


Submodeling

- **Example (cont'd): Shell-to-Solid Submodel of a Pipe Joint**



Solid submodel overlaid on the global shell model in deformed state



coincident contours at the boundary indicate that the results are valid

Comparison of out-of-plane displacement (z-direction) in the plate for both models

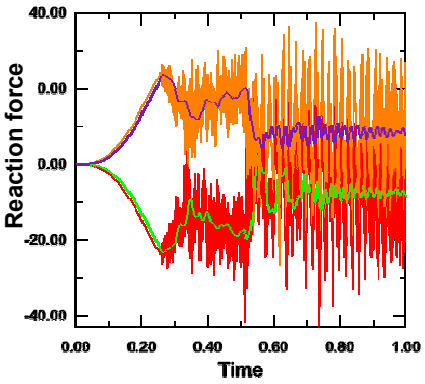
Submodeling

- **Submodeling in dynamic impact problems**
  - For significantly dynamic problems in ABAQUS/Explicit, a sufficiently large number of output frames need to be written to the output database or results file for the global model.
    - A high frequency of output is necessary in particular for problems with elastic materials that involve elastic impact to avoid possible aliasing (under sampling), which can cause solution distortion in the submodel.
      - These problems exhibit excessive vibration.
      - The displacement results for the nodes that are used to drive the submodel should be saved for each increment.
    - Problems with significant plastic deformations require less frequent global output since oscillations in the displacements are greatly reduced.

Output Filtering

Output Filtering

- **Real-time filtering**
  - You can eliminate the noise associated with an analysis output signal by real-time filtering of the history output results.
  - Element, nodal, contact, integrated, and fastener interaction history output can be pre-filtered before they are written to the output database.
  - Real-time filters allow you to significantly reduce frequency of history output,
    - which in turn may reduce the size of the output database.
    - If real-time filters are not used for history output, output needs to be requested at each increment to avoid aliasing of the results



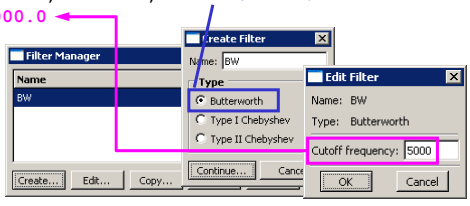
Comparison of raw history data with real-time filtered history data

Output Filtering

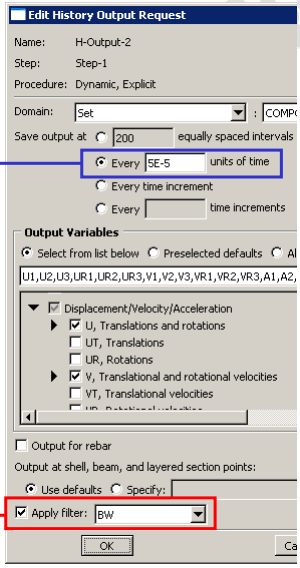
- **Example: Drop Test Simulation of a Cordless Mouse**

Define **BUTTERWORTH** filter. Type I Chebyshev (**CHEBYS1**) and Type II Chebyshev (**CHEBYS2**) filters are also availabe.

```
*FILTER, NAME=BW, TYPE=BUTTERWORTH
 5000.0
.
.
.
```




```
*OUTPUT, HISTORY, FILTER=BW,
  TIME INTERVAL=5E-5
*NODE OUTPUT, NSET=COMPONENT_REFS
 U,V,A
```



Output Filtering

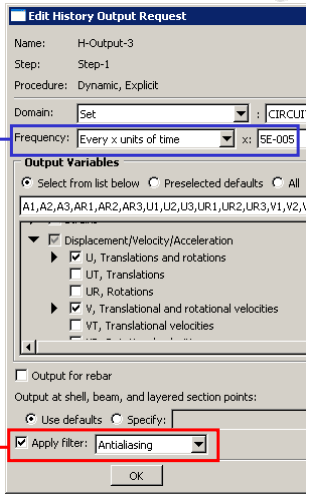
- **In addition to the user defined filters, ABAQUS/Explicit also includes a built-in anti-aliasing filter.**
  - The anti-aliasing filter is internally based on the sampling interval that is specified in the history output request.
    - The cutoff frequency is one-third of the sampling frequency.
      - In this case the sampling frequency is the inverse of the time interval.
      - For example, if the desired output time interval is 1e-4, the cut off frequency used by the anti-aliasing filter is 1/(3*1e-4) ~ 3300 Hz.
    - Hence, the anti-aliasing filter filters only the extremely high frequency noise content of the signal while preventing aliasing of the signal due to a large output time interval size.
      - Therefore, additional filtering operations can be performed after the analysis during post-processing.

Output Filtering

- **Example (cont'd): Drop Test Simulation of a Cordless Mouse**

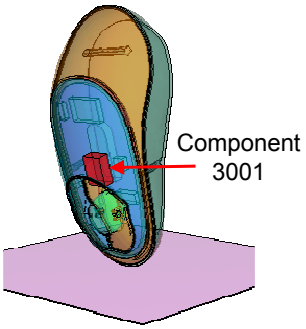
```
*FILTER, NAME=BW, TYPE=BUTTERWORTH
 5000.0
.
.
.
*OUTPUT, HISTORY, FILTER=BW,
  TIME INTERVAL=5E-5
*NODE OUTPUT, NSET=COMPONENT_REFS
 U,V,A
*OUTPUT, HISTORY, FILTER=ANTIALIASING,
  TIME INTERVAL=5E-5
*NODE OUTPUT, NSET=COMPONENT_REFS
 U,V,A
```

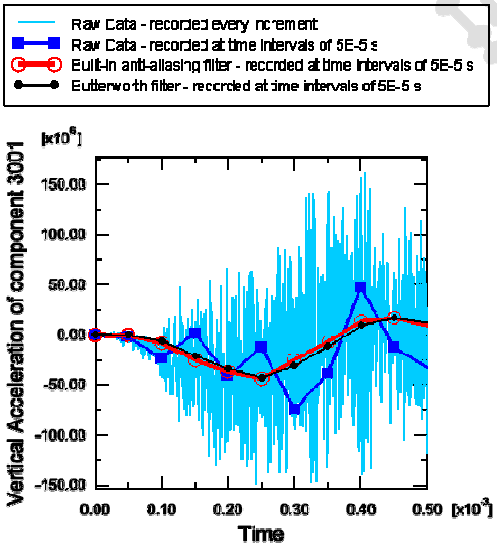
Output time interval (DT) is 5e-5, so the cut off frequency used by the built-in anti-aliasing filter is 1/(3*5e-5) = 6,667 Hz.



Output Filtering

- Example (cont'd): Drop Test Simulation of a Cordless Mouse





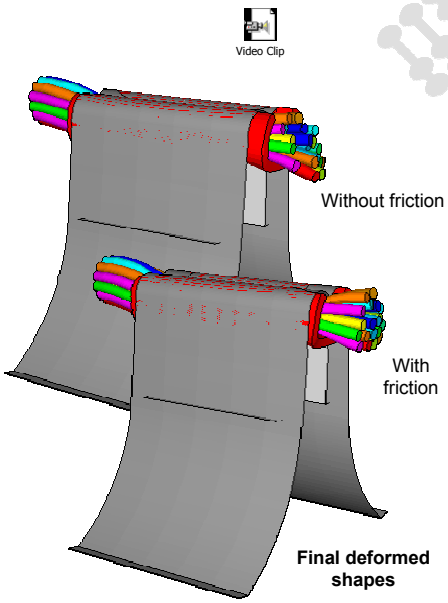
Restart

Restart

- **The restart functionality allows you to continue an analysis from a particular point in a previous analysis.**
  - Large models that take a long time to run are more likely to stop before completion than smaller analyses that run quickly.
    - A job may stop because:
      - There was not enough disk space
      - The machine failed (e.g., due to a power failure)
  - When solving large problems you may wish to continue an analysis (without rerunning it from the beginning) after:
    - Examining results up to a particular point.
    - Modifying history: procedure, loading, output, or controls.

Restart

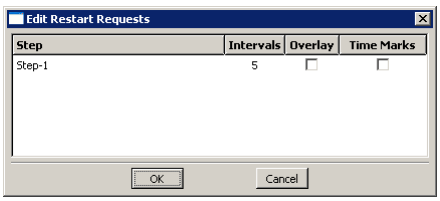
- **Example: Wire crimping analysis**
  - Previously, the wire crimping analysis was run both with and without friction.
  - A follow up investigation is proposed to determine how sensitive the crimp forming is to the friction coefficient for the wires.
  - The earlier results indicate that friction only plays a significant role towards the end of the analysis (the last 20%).
    - Therefore, we can use restart to avoid reanalyzing the first 80% of the analysis.



Restart

- **Example (cont'd): Wire crimping analysis**
  - In the original models restart data were requested at 5 evenly spaced intervals.

```
*RESTART, WRITE, NUMBER INTERVAL=5
```



  - ABAQUS produced the following files, which are required to restart the analysis from any of the 5 points when restart data were written:
    - restart (`.res`),
    - analysis database (`.abq`, `.mdl`, `.pac`, and `.stt`),
    - part (`.prt`),
    - selected results (`.sel`), and
    - output database (`.odb`) files.

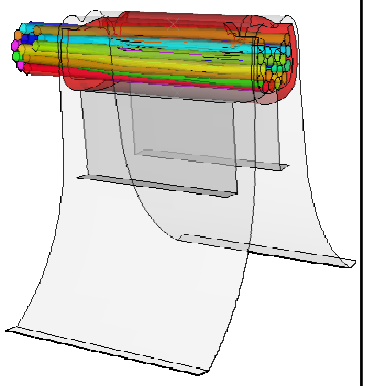
Restart

- **Wire crimping analysis using keywords**
  - Restart the analysis with friction from the 4th point at which restart data were recorded.
    - The analysis is 80% complete at this point.

1a Add restart option:

```
*RESTART, READ, STEP=1, INTERVAL=4, END STEP
```

The END STEP parameter is used to end the original step so that the wire friction coefficient can be modified before the analysis continues.



Deformed shape
friction analysis 80% complete

Restart

- **Wire crimping analysis using keywords**

1b Define amplitude and friction:

```
*RESTART, READ, STEP=1, INTERVAL=4, END STEP
**
*AMPLITUDE, NAME=PUNCH_DOWN_DISP-2,
   DEFINITION=SMOOTH STEP,
   VALUE=ABSOLUTE, TIME=TOTAL TIME
 0.,0.,0.018,-0.9,0.030833,-4.75,
 0.090833,-6.25, 0.122333, -6.88
**
*SURFACE INTERACTION, NAME=WIRES-IncreaseFriction
*FRICTION
 0.3,
```

Punch displacement amplitude curve redefined with TIME=TOTAL TIME so that punch displacement will continue in the restart step as it had in the original step.

New surface interaction for the wires with a coefficient of friction 2× larger than the value used in the original analysis.

Restart

- **Wire crimping analysis using keywords**

2 Define the step:

```
*STEP, NAME=STEP-2
PUNCHDOWN - INCREASE FRICTION
*DYNAMIC, EXPLICIT
 ,0.024466
**
*BOUNDARY, AMPLITUDE=PUNCH_DOWN_DISP-2
PUNCH_REFNODE, 2, 2
**
*CONTACT, OP=NEW
*CONTACT INCLUSIONS, ALL ELEMENT BASED
*CONTACT EXCLUSIONS
 ANVIL, PUNCH
*CONTACT PROPERTY ASSIGNMENT
 ,  , "GLOBAL PROPERTY"
 WIRES,  , WIRES-IncreaseFriction
 GRIP, ANVIL, GRIP_ANVIL
 GRIP, PUNCH, GRIP_PUNCH
 GRIP,  , GRIP_ANVIL
```

New step time is 20% of the original step time

Punch displacement amplitude curve based on total analysis time (defined on the pervious slide)

Surface interaction with increased friction (defined on the pervious slide) used for wire contact.

Restart

- **Wire crimping analysis using keywords**
  - 3 Keyword interface job submission:
    - At the command prompt

```
abaqus job=job-name oldjob=oldjob-name
```

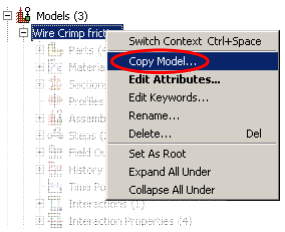
job=*job-name*: name of the new input file

oldjob=*oldjob-name*: name of the restart file created by the previous run

Restart

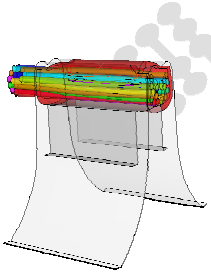
- **Wire crimping analysis using ABAQUS/CAE**
  - In ABAQUS/CAE, the model used in the restart analysis must be the same as the model used in the original analysis up to the restart location.
    - Must not modify or add any geometry, mesh, materials, surfaces, etc. that are already defined in the original analysis model
    - Must not modify any step, load, boundary condition, field, or interaction at or before the restart location
    - New sets and amplitude curves may be defined in the restart analysis model.

1 Copy the original model (optional).



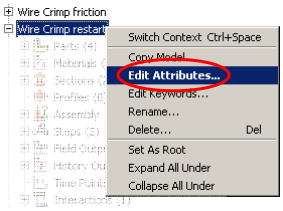
Restart

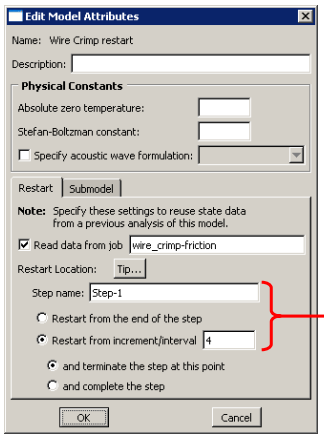
- **Wire crimping analysis using ABAQUS/CAE**
  - Restart the analysis with friction from the 4th point at which restart data were recorded.
    - The analysis is 80% complete at this point.



Deformed shape
Step-1, interval 4
(80% complete)

2 Modify the model attributes to define restart data.

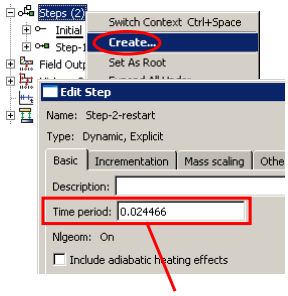




Restart

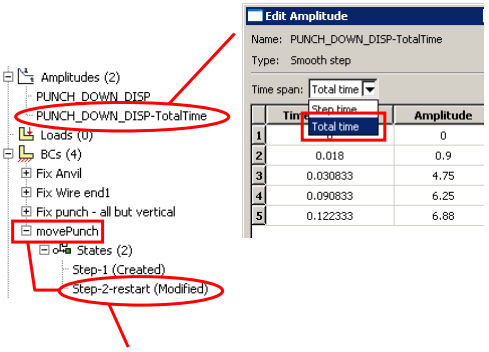
- **Wire crimping analysis using ABAQUS/CAE**

3 Create a new step.



New step time is 20% of the original step time

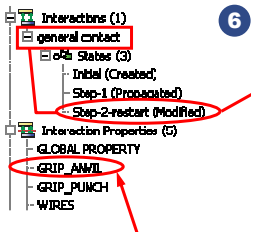
4 Copy amplitude curve and modify new curve to be based on total time.



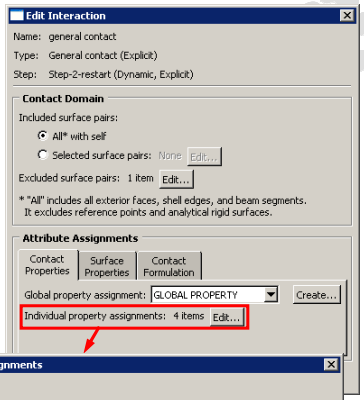
5 Modify the punch displacement boundary condition in the new restart step to use total time amplitude curve.

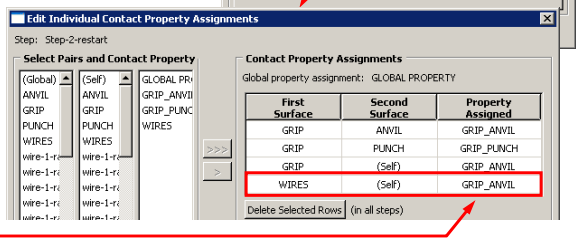
Restart

- **Wire crimping analysis using ABAQUS/CAE**



6 Modify the general contact definition to use the new wire contact interaction property.



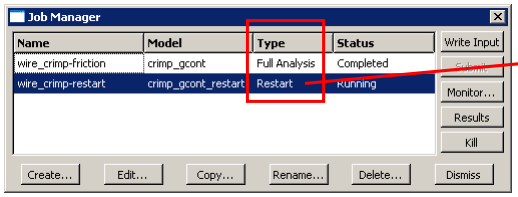


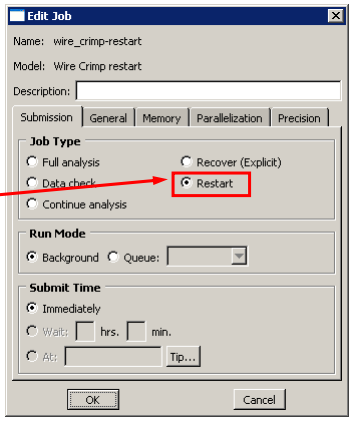
Existing contact interaction property with friction coefficient of to 0.3 (2× larger than the value used for wire contact in the original).

Restart

- **Wire crimping analysis using ABAQUS/CAE**

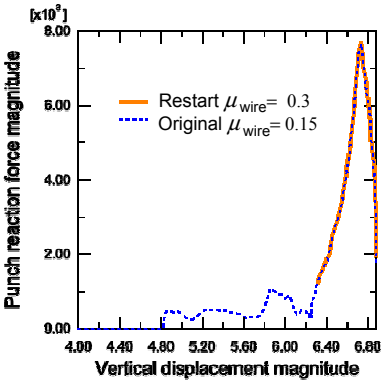
7 Create and submit the restart analysis job



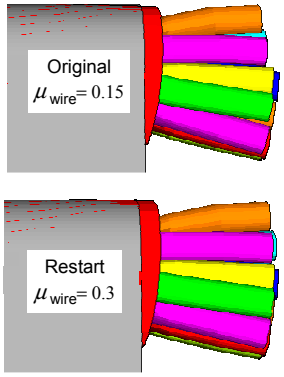


Restart

- **Wire crimping analysis results**
  - Doubling the friction coefficient for contact between the wires does not significantly influence the analysis results.



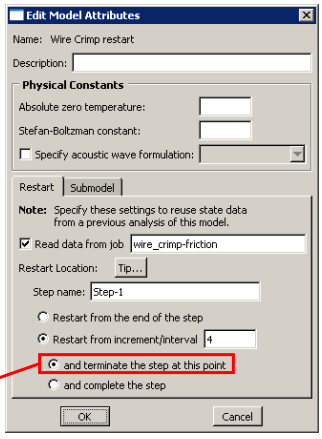
Force displacement curves

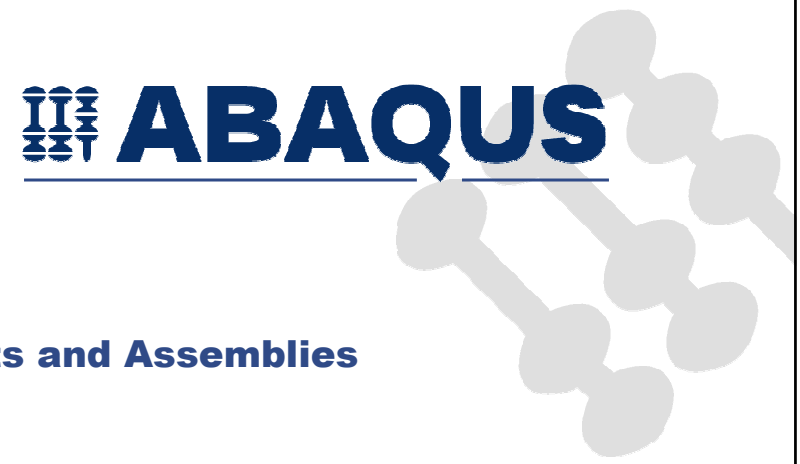


Final deformed shape

Restart

- **Summary**
  - The flexible restart capabilities in ABAQUS are very helpful for managing complex analyses.
  - There are three important rules to remember:
    - It is not possible to append to a restart file.
    - All output requests and loads from the previous run remain in effect upon job restart unless explicitly modified in a new step.
    - If restarting from an “unfinished” run, ABAQUS will first try to finish the step it was working on during the original analysis before starting any new steps.
      - Use the END STEP parameter to override this.





Parts and Assemblies

Parts and Assemblies

- **Consider defining complex models as an assembly of part instances.**
  - Allows reuse of part definitions
    - Parts that are used more than one time in the assembly only need to be defined once.
  - Allows for independent node and element numbering for each part
    - Advantageous when modeling complex assemblies comprised of parts that have been designed and analyzed independently.
- **ABAQUS/CAE is designed around the concept of an assembly of part instances.**
  - Input files written by ABAQUS/CAE are written in terms of an assembly of part instances by default.
- **For input files not written by ABAQUS/CAE, the use of part and assembly definitions in the input file is optional.**

Parts and Assemblies

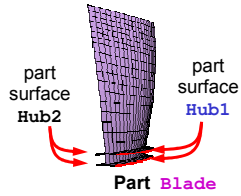
- **Example: Fan assembly based on multiple instances of a blade**

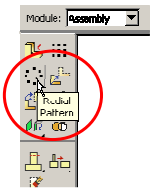
part definition

```
*PART, NAME=Blade
*NODE
 :
*SURFACE, TYPE=ELEMENT, NAME=Hub1
 Edge1,E2
*END PART
```

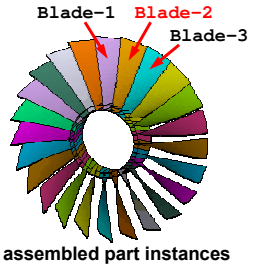
assembly definition

```
**
*ASSEMBLY, NAME=ASSEMBLY
*INSTANCE, NAME=Blade-1, PART=Blade
*END INSTANCE
**
*INSTANCE, NAME=Blade-2, PART=Blade
 0., 0., 0.
 0., 0., 0., 0., 1., 0., 15.0
*END INSTANCE
**
*INSTANCE, NAME=Blade-3, PART=Blade
 0., 0., 0.
 0., 0., 0., 0., 1., 0., 30.0
*END INSTANCE
 :
*END ASSEMBLY
```





Position instance `Blade-2` with no translation and 15° rotation of part `Blade`



assembled part instances

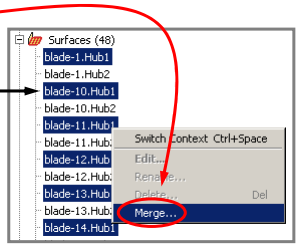
Parts and Assemblies

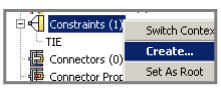
- **Example (cont'd):**

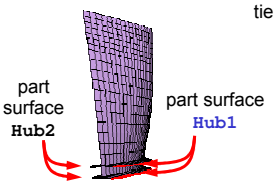
```
*END ASSEMBLY
**
*SURFACE, NAME=TIE-1, COMBINE=UNION
Blade-1.Hub1, Blade-2.Hub1, Blade-3.Hub1,
Blade-4.Hub1, Blade-5.Hub1, Blade-6.Hub1,
Blade-7.Hub1, Blade-8.Hub1,
Blade-9.Hub1, Blade-10.Hub1,
.
.
.
**
*SURFACE, NAME=TIE-2, COMBINE=UNION
Blade-1.Hub2, Blade-2.Hub2, Blade-3.Hub2,
 :
*TIE, NAME=TIE, ADJUST=YES
TIE-1, TIE-2
 :
```

Merge surfaces into a single surface

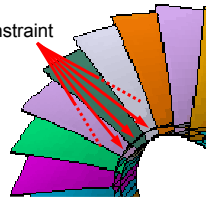
Surface `Hub1` (defined on the part `Blade`) of instance `Blade-10`







tie constraint



fan assembly