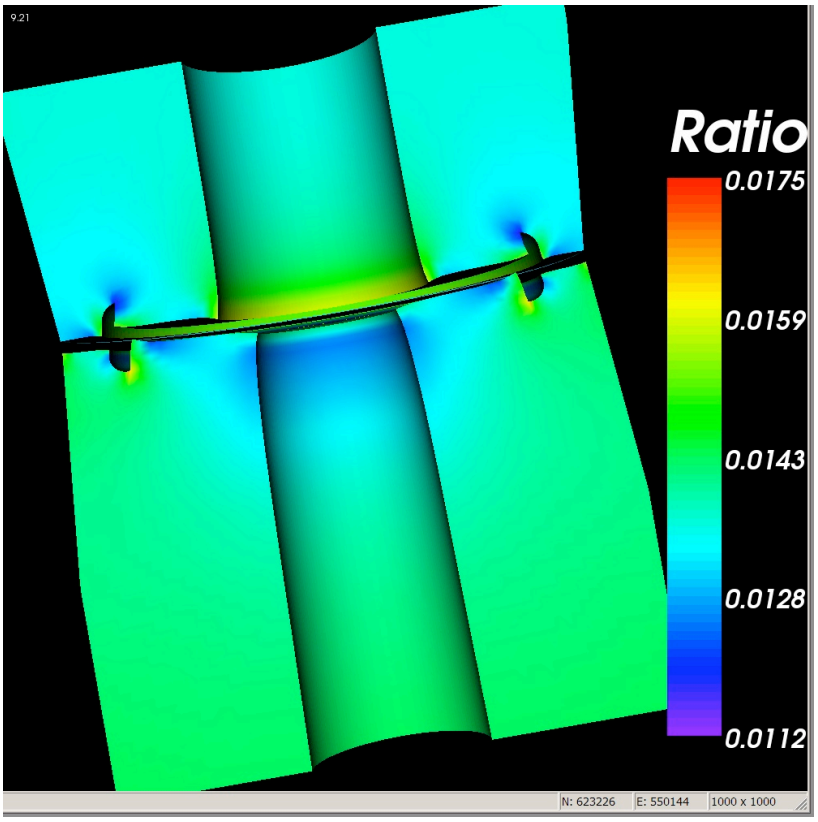
ROCSOLID User's Guide

Version 3.4

March 17, 2005

Ali Namazifard



Center for the Simulation of Advanced Rockets
University of Illinois
Urbana, Illinois

Table of Contents

1. Purpose and Methods

ROCSOLID is an implicit finite element code to solve very large scale, 3D structural mechanics problems subjected to static and dynamic loads. ROCSOLID is under continuing development as a research code and has the following capabilities:

- Three new nonlinear constitutive material models are now available in ROCSOLID. They are compressible and incompressible Neo-Hookean models and porous viscoelastic (with void growth) material model.
- Heat conduction. To obtain the initial temperature distribution for use with temperature dependent burn rate in Rocstar.
- Nonlinear kinematics (large deformations) since ROCSOLID 3.2. A new element type called b8_ld is implemented for this purpose. Using this type of element invokes the nonlinear solver automatically.
- A J2 plasticity model based on rate independent von Mises material model with the associated flow rule utilizing a bilinear uniaxial material response in addition to linear elasticity exists in this version of ROCSOLID but has not yet been used in coupled simulations.
- Eight node hexahedral elements are used. This element uses the consistent material tangent operator (to preserve the quadratic convergence rate when the Newton nonlinear solver is called). The B-bar integration rule is used to avoid locking for near incompressible conditions.
- Dynamic problems are solved using the implicit Newmark time integrator with full Newton iterations.
- The linear matrix equations encountered at each time step are solved using a scalable parallel multigrid solver. The preconditioned conjugate gradient method is used as the relaxation scheme. All of the main components in the multigrid solver are implemented in an element-by-element framework. Matrix free element level computations are used to reduce the storage and CPU time.
- During each multigrid solution, interprocessor communications are performed in matrix-vector multiplications, scalar products and fine-to-coarse mesh restrictions. Nonblocking MPI communication routines are used to perform point-to-point communications.
- ALE formulation is used for moving interfaces.
- Mixed-enhanced elements are used for shell elements.
- ROCSOLID uses unstructured meshes. *Truegrid* is used to produce a sequence of nested, uniformly refined hexahedral meshes. Mesh partitioning is performed on the

coarsest mesh using *Metis* to achieve perfect load balance between the processors. Uniform refinement of the coarsest mesh partitions produces the required partitions on all of the fine meshes. Thus, perfect element load balance is maintained through the mesh hierarchy, although the resulting communication pattern may not be optimum. The files generated by *Truegrid* are processed by ROCSOLID preparation tool to generate the necessary HDF input files for ROCSOLID.

More detailed descriptions of the algorithms used in ROCSOLID are provided in the *Developer's Guide*. In addition, the documentation that accompanies the component codes in Rocstar should also be consulted for coupled simulations.

2. Building and Running

To date, ROCSOLID has been tested and is supported on many platforms including the following :

Machine	OS	Fortran	MPI
SGI Origin 2000	IRIX 6.5	SGI f90	SGI MPI
Turing cluster	Mac OS X	xlf90	MPICH
IBM SP	AIX 4.3.3	IBM xlf90	IBM MPI

ROCSOLID is provided with a makefile containing options for all these platforms. To compile on a specific platform, the user needs to modify the makefiles in the directory Src in order to uncomment the appropriate portion for the specific platform. The code can be compiled in the Src directory by invoking "gmake". This will create the executable file rocsolid.x.

To run the code, the user needs to run rocsolid.x in the data directory. For example, on the Origin 2000, the code can be run interactively by using the command line:

mpirun -**np** *n PATH*/**rocsolid.x**

On Blue Horizon, the command line is:

poe *PATH*/rocsolid.x -nodes *n* -tasks_per_node *m* -rmpool 1

The easiest way for using ROCSOLID is through the Rocstar framework. Rocstar can be compiled as usual, which would also compile ROCSOLID modules. ROCSOLID can then be executed in the stand-alone mode. Please consult the documentations for Rocstar for more details.

3. Input and Output (User Interface)

3.1 Input Data

The ROCSOLID input data set can either be generated by calling Rocprep or by using ROCSOLID preparation tool rsolidprep. Consult the Rocprep documentation for more information. ROCSOLID preparation tool is described later in this section.

3.1.1 Data placement information (Not needed for Rocstar 3)

The file `solidsinput` contains the following information. All items are required and can have a maximum of 30 characters.

```
line 1: OutDirectName
line 2: InputFileName
line 3: ControlFile
line 4: IntefaceFile1
line 5: InterfacFile2
line 6: hdfdir
```

`OutDirectName`	Name of a directory that can be used for output. This directory should be created by the user.
`InputFileName`	The main input file has the format of Name**** where the name is specified here and the **** is the partition id in I4.4 format.
`ControlFile`	Name of the control data file explained in the next section.
`InterfaceFile1`	Name of the file containing interface nodal displacement (used for checking the ALE formulation in the stand-alone mode).
`InterfaceFile2`	Name of the file conatining interface element tractions (used for checking the ALE formulation in the stand-alone mode).
`hdfdir`	The name of the directory used for HDF files

3.1.2 ROCSOLID Control Data

The ROCSOLID control data is listed in the file `ControlFile` which its name is specifiled in the `solidsinput` file (only for GEN2.5 or GEN2.6). The data should be entered in free format in the following manner. Line numbers are used to make this guide easier to read and they do not refer to the actual line numbers in the file (because of the if statements for example).

```
line 1: title
line 2: num_elem_group, num_mat_sets, num_dof, num_meshes,
block_size
line 3: gamma, nu1, nu2, num_mg_meshes, mg_tol, mg_maxcycle
line 4: pcg_tol, pcg_maxcycle
line 5: preconditioner
line 6: nl_solver
line 7: total_num_steps, arc_tol, newton_max, arc_length
         (if nl_solver=ARC_LENGTH)
line 8: num_loadsteps, newton_tol, newton_max
         (if nl_solver=NEWTON)
line 9: problem
line 10: MassMatrix (if problem=DYNAMIC)
line 11: num_time_steps, delta_t, newmark_gamma,
          (if problem=DYNAMIC)
line 12: force_lambda(i), (i=0,num_time_steps)
          (if problem=DYNAMIC)
line 13: subsp_flag
line 14: grav_x, grav_y, grav_z
line 15: ASCIIOutputFlag, PatranOutputFlag, HDFOutputFlag,
         RestartBinFlag, RestartHDFFlag
line 16: results_mesh_motion_output_flag,
patran_mesh_motion_flag
line 17: FilesPerDump
line 18: equ_solver
line 19: mesh_motion_equ_solver
line 20: mem_limit
```

The following material information should be entered for each material set:

```
line 21: material_type
line 22: name
line 23: e, nu (if type=elastic or neohookean) OR e, et, nu
         (if type=j2_plasticity)
line 24: beta, yield_stress (if type=j2_plasticity)
line 25: density
```

```
If material type = porous_viscoelastic, the following
information should be supplied after name:

line 23: shear_mod_p, total_shear_mod
line 24: total_bulk_mod
line 25: time_const
line 26: initial_porosity
line 27: density
```

All of the element types should be listed next:

```
line 26 (or 28): element_name
line 27 (or 29): num_elem, Num_int_pts (this line should be
entered for each level of multigrid if this solver is used)
```

These variables are defined as follows:

`title`	Title of the problem (string)
`num_elem_group`	Number of element groups (integer)
`num_mat_sets`	Number of material sets (integer)
`num_dof`	Number of degrees of freedom (integer)
`num_meshes`	Number of meshes (integer)
`block_size`	Block size (integer)
`gamma`	gamma in multigrid (real)
`nu1, nu2`	nu1, nu2 in multigrid (integer)
`num_mg_meshes`	Number of MG meshes (integer)
`mg_tol`	Multigrid tolerance (real)
`mg_maxcycle`	Maximum number of MG cycles (real)
`pcg_tol`	PCG tolerance (real)
`pcg_maxcycle`	Maximum number of PCG cycles (integer)
`preconditioner`	Preconditioner (string), Only JACOBI preconditioner is available for now

`nl_solver`	Nonlinear solver (string), either ARC_LENGTH or NEWTON is available.
`problem`	Problem type (string), STATIC or DYNAMIC
`MassMatrix`	Mass matrix type (string), CONSISTENT or LUMPED
`num_time_steps`	Number of time steps (integer)
`delta_t`	Time increment (real)
`newmark_gamma`	Gamma value in Newmark (real)
`force_lambda`	Force magnitude (real)
`subsp_flag`	Subspace flag (logical)
`grav_x, …`	gravitational forces (real)
`ASCIIOutputFlag`	ASCII output flag (logical)
`PatranOutputFlag`	Patran output flag (logical)
`HDFOutputFlag`	HDF output flag (logical)
`RestartBinFlag`	Binary restart flag (logical)
`RestartHDFFlag`	HDF restart flag (logical)
`results_mesh…`	Flag to output mesh motion results (logical)
`patran_mesh…`	Flag to output patran mesh motion results (logical)
`FilesPerDump`	HDF control parameter (integer)
`equ_solver`	Equation solver (JAC_PCG or MULTIGRID)
`mesh_motion_equ_solver`	Mesh motion equation solver (only BICGSTAB is available for now)
`mem_limit`	Memory limit allocated on each processor (MB)
`material_type`	material type (elastic or j2_plasticity)
`name`	Name of the material
`e`	Elastic modulus
`et`	Yield modulus (for j2 plasticity model)
`nu`	Poisson ratio
`beta, yield_stress`	material parameters for j2 plasticity model

`density`	Density of the material
`element_name`	Element name (b8_bbar or b8_me or b8_ale or b8_ld or b8_heat)
`num_int_pts`	Number of integration points (= 2 for now)

A discussion of many of these parameters is contained in the *ROCSOLID Developer's Guide.*

3.1.3 ROCSOLID Preparation tool rsolidprep (Rocstar 3)

Two files are needed for rsolidprep; RocsolidPrep.inp which includes the control parameters for rsolidprep and the mesh file with extension trugrd. This mesh file has to be in Nike3D format generated by Truegrid mesh generation software. During the mesh generation, there are a few guidelines that need to be followed.

When using the FDI command in Truegrid for boundary condition, the values for different components must be 10 for fixed and -1 for prescribed boundary condition (ALE) at each node. Everywhere else would be assumed free. The load curve number in FDI has to be selected 1 for regular boundary condition or 2 for coupled or "wet" boundary. The PRI command is usually used to apply pressure boundary condition. A pressure value should be assigned at all wet surfaces.

The rsolidprep executable exists at Codes/bin directory. To run this tool, you should do the following:

In your running directory, (where you must have the Rocstar executable or link along with modules directories) cd to Rocsolid directory. You should copy the files RocsolidPrep.inp and *.trugrd to the Modin directory. You can now run rsolidprep in the Rocsolid directory. The HDF or CGNS input files will be generated in the required place for Rocstar, i.e., Rocsolid/Rocin.

3.1.3.1. RocsolidPrep Control file (RocsoldiPrep.inp)

The data should be entered in free format in the following manner. Line numbers are used to make this guide easier to read and they do not refer to the actual line numbers in the file.

```
line 1: title
line 2: Nike3DFile (without the extension trugrd)
line 3: prefx (will specify the case name for HDF files)
line 4: NumberOfMeshes, NumberOfBlocks
line 5: CoupledFlag, InchesToMetersFlag, MilimitersToMetersFlag
        (.True. or .False.)
line 6: NumberOfPartitions (partitioning is done by Metis)
```

```
line 7: RocsolidOutFlag, PatranOutputFlag (.True. or .False.)
line 8: AsciiInputFlag, HdfInputFlag (only one of them can be
        .true., the Ascii format is used for GEN2.5 or GEN2.6)

The following line should be specified for each block in the
mesh, (NumberOfBlock times):

line 9 to 9+NumberOfBlocks: ni, nj, nk (number of elements at
                            the coarsest level)
```

3.1.4 Large Deformations (ROCSOLID 3.2)

In order to use the large deformation algorithm available in ROCSOLID 3.2, elements of type b8_ld have to be used. This type of element will automatically invoke the Newton solver to solve the nonlinear equations. The parameters related to the Newton solver need to be defined in the control file. The static analysis with this type element is not available yet.

3.1.5 Mesh Files (Ascii format)

The main input file for each partition is described here. The names of the files are the string specified in the solidsinput file plus the partition id in I4.4 format (e.g. rocsolid0000, rocsolid0001, …).

The first item is the total number nodes in the partition followed by their **coordinates**:

`num_nodes` (Integer, free format)

`node_num coorX coorY coorZ` (Integer, 3*real)

The data should continue one line per node until all of the nodes defined in the partition are covered.

Next block of input data is the **nodal boundary conditions**. For each node, 1 specifies free and 0 denotes fixed boundary condition in each global XYZ direction.

`node_num D1 D2 D3` (Integers, D1,D2,D3 = 1 or 0)

The input should continue one line per node until all of the nodes with specified boundary conditions are covered. To show the end of the input block, a line with –1 as the node number is used. If there are no nodes with fixed boundary conditions in the partition, only the termination line is needed.

Nonzero specified nodal loads and displacements are read in the next section. This feature of the code is not yet fully tested.

`node_num  r1  r2  r3` (Integer, 3*real)

The termination line has –1 as the node number. For now, only the termination line is required.

For each element group (b8_bbar, b8_me, b8_ale and b8_ld) the following element level information is required next.

For each level of multigrid, from the finest to the coarsest, number of elements is specified one line per multigrid level. For example, if there are three multigrid meshes, the next three lines will have the number of elements for each mesh.

`num_elem1` (Integer, number of elements on the finest level)

`num_elem2` (Integer, number of elements on the next coarse level)

The number of lines (or the multigrid levels) is specified in the control file.

The finest mesh **material and connectivity arrays** are listed next:

`elem_num  material_num  conn(1:8)` (Intergers)

The first integer is the element number which does not have any significance other than making the data file more readable. The data should be entered one element per line starting from element one until the last element. Depending on the order of the materials defined in the control file, each element is assigned with a material number. For example, if an element has the properties of the second material, its material number would be 2. The connectivities are entered as eight integers (free format) with the order explained in the *ROCSOLID Developer's Guide.*

The number of finest mesh **element traction boundary condition** for this partition is read next:

`num_trac_faces` (Integer)

If this number is nonzero (i.e., there are tractions applied on the faces of some elements in this partition), element number, face number and three components of tractions should be listed next:

`elem_num  face_num  tracX  tracY  tracZ` (2*integers, 3*real)

The number of enteries (lines) are equal to num_trac_faces (number of traction faces).

When the multigrid solver is used, **element component arrays** should be listed for each coarse mesh. Please refer to the *ROCSOLID Developer's Guide* for more information about this array.

`element_num  component_array(1:8)` (Integers)

The element numbers are only used to make the file more readable. The input starts from element one until the last element in the coarse mesh. This set of data is repeated for each coarse mesh.

The **communication data** or the border nodes between the partitions are listed in the next section. The mesh file corresponding to this partition should define the communication between itself and all other partitions. The partition numbers start from one.

`partition1  partition2` (Integers)

`num_border_np` (Integer, number of border nodal points)

`border_nodes(1:num_border_np)` (20I7, border nodes on partition1)

Since the border nodes are the ids of the nodes on partition1, the reverse combination should also be listed to specify the nodes on partition2. When two partitions do not share any nodes num_border_np is zero and the next line for border_nodes is not needed. Twenty numbers per line should be listed. If multigrid is used, the same information has to be repeated for the coarser meshes.

Next block of input is the **mesh motion nodal data**. For each node, 1 specifies free, 0 denotes fixed and –1 assigns prescribed motion (interface) property to the node in each global XYZ direction.

`node_num  D1  D2  D3` (Integers, D1,D2,D3 = 1 or –1 or 0)

The input should continue one line per node until all of the nodes with prescribed motion or interface property are covered. To show the end of the input block, a termination line with –1 as the node number is used. If there are no nodes with prescribed motion property in the partition, only the termination line is needed.

3.2 Output Data

In Rocstar 3, Rocout is used for output. You can find all the resulted HDF output files at Rocsoldi/Rocout and visualize them by ROCKETEER.

ROCSOLID 3.2 inside GEN2.5 uses the capability of the integrated code for output and visualization. The data needed for output and visualization is registered with ROCCOM and the required HDF files are produced. These results can then be visualized by ROCKETEER.

ROCSOLID also produces files containing output for the following data:

- PATRAN mesh and results files;

- Binary mesh and results files;
- PATRAN interface mesh files;

4. Examples, Test Problems and Verification

4.1 Scalability Data Sets

4.1.1 Problem Description

The `Scalabilty` subdirectory in the ROCSOLID `Tests` subdirectory contains the ROCSOLID input files for a simple model of a solid rocket motor. Data are provided for mesh partitions on 1, 8, 16, 32, 64, 128, 256 and 512 processors.

4.1.2 Problem Objective

The objective of this sequence of simulations is to measure the scalability of ROCSOLID by solving a series of scaled problems.

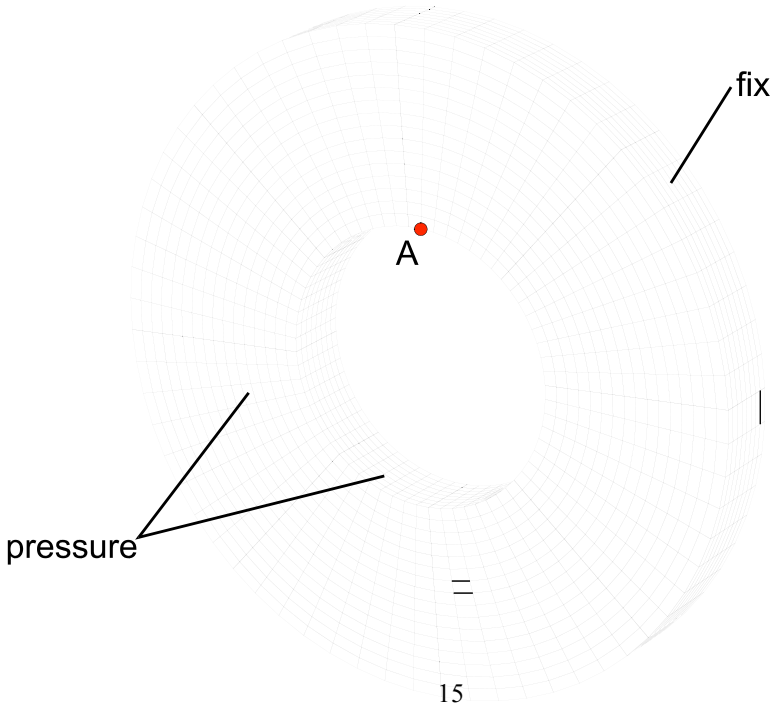
4.1.3 Problem Solution

Timing data is available that measures the scalability of ROCSOLID on several platforms.

4.2 Transient Solution for Pressurized Disk

4.2.1 Problem Description

A simple disk is discretized with hexahedral elements and pressurized on one face and from inside. The pressure is increased linearly with time and the displacement of a node denoted as node A in the next figure is compared with the result produced by ABAQUS.

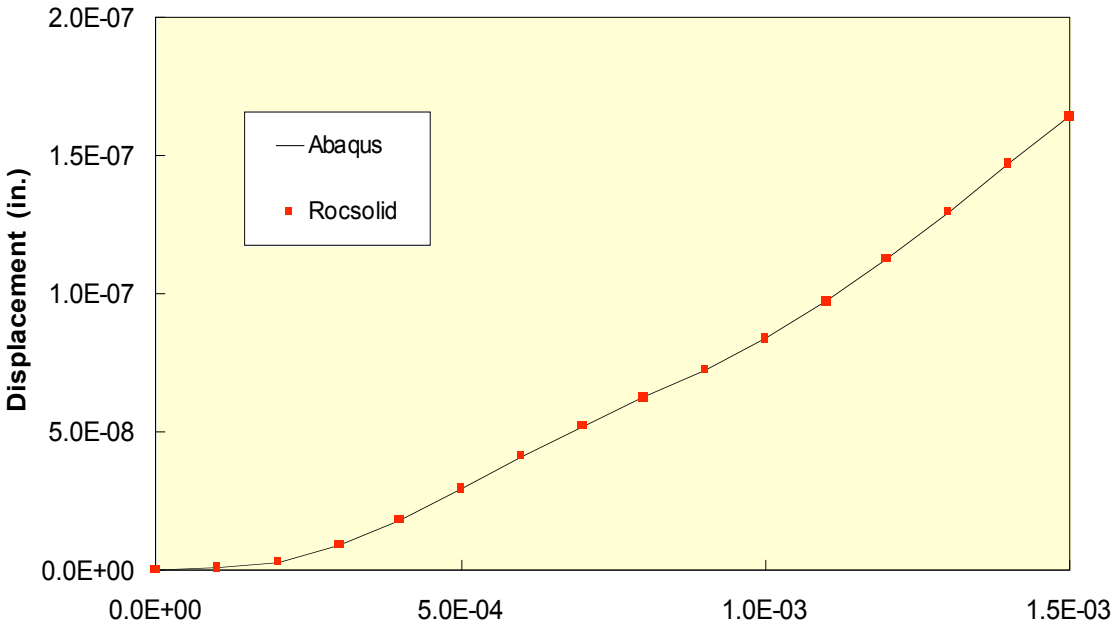


4.2.2 Problem Objective

This is a single partition problem. It verifies all of the non-ALE ROCSOLID routines needed to compute displacements without any complications from the parallel processing subroutines.

4.2.3 Problem solution

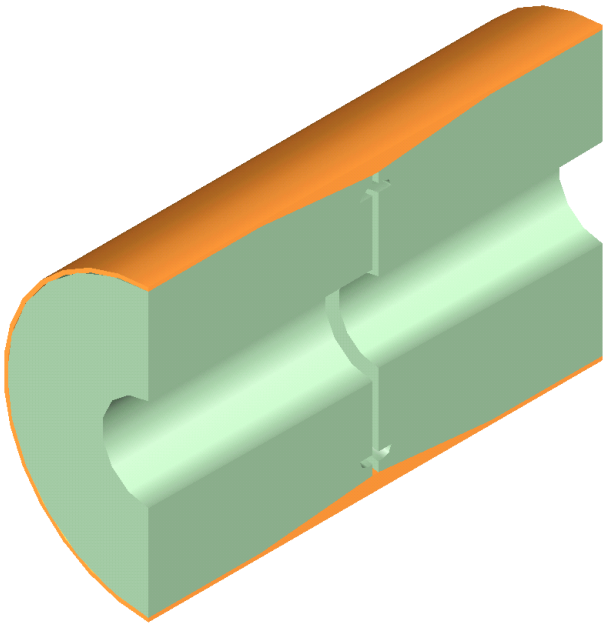
The solution (axial displacement at A) and the comparison with ABAQUS results is shown in the next figure.



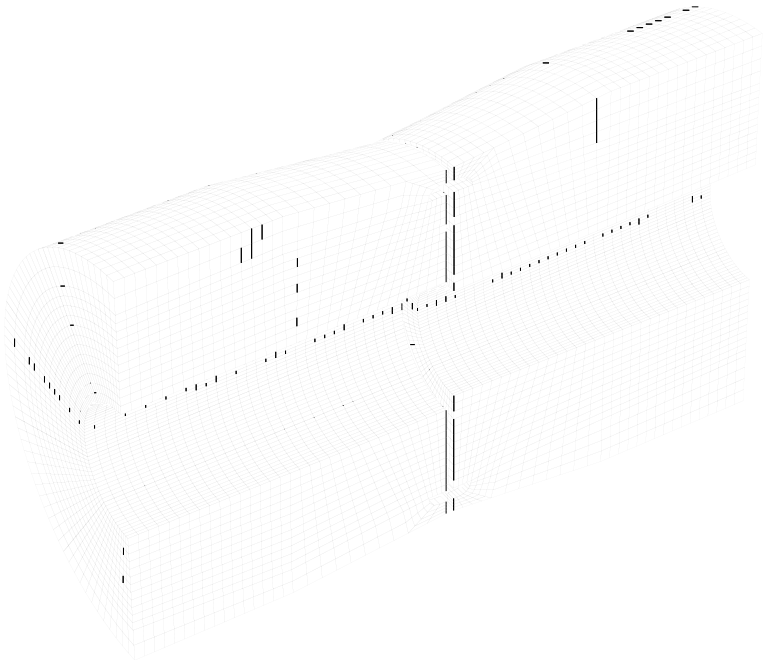
4.3 Transient Solution for the Titan IV problem

4.3.1 Problem Description

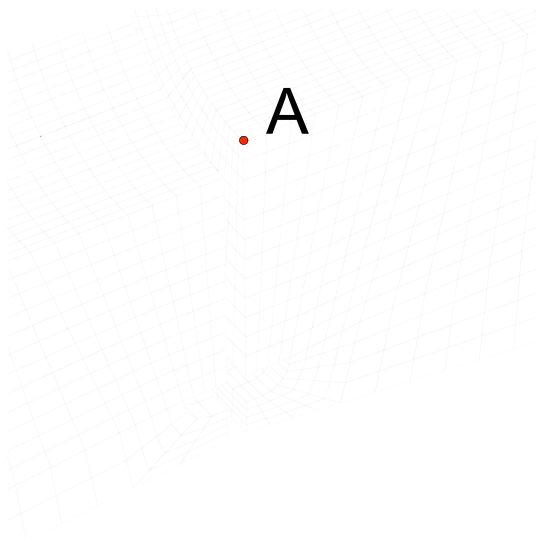
A structural analysis is performed using a segment of the Titan IV shown in the next figures.



The Solid Edge Model for a segment of Titan IV



The Truegrid mesh for a segment of Titan IV

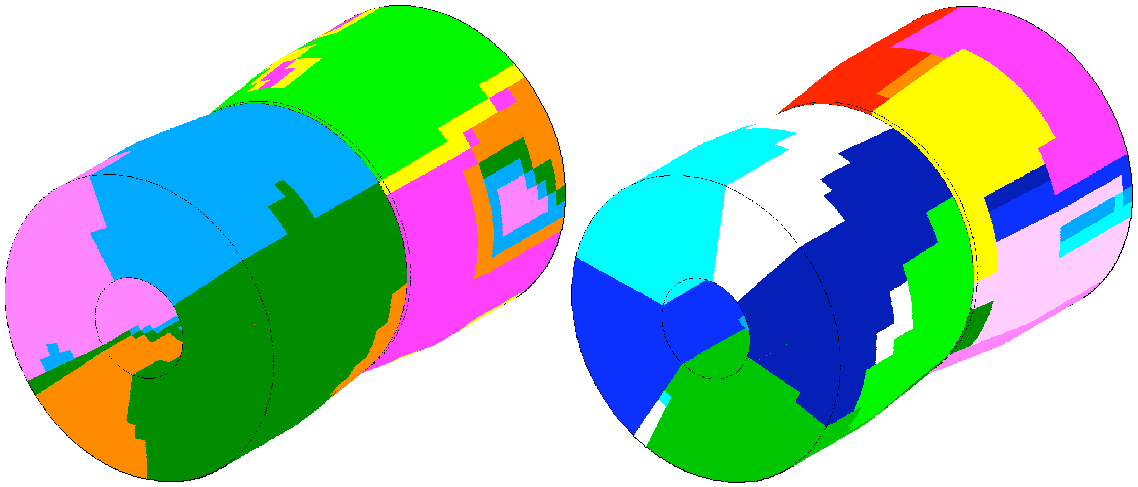


The selected node for displacement verification

The model is again pressurized from inside including inside the joint. The displacements are measured at node A shown above.

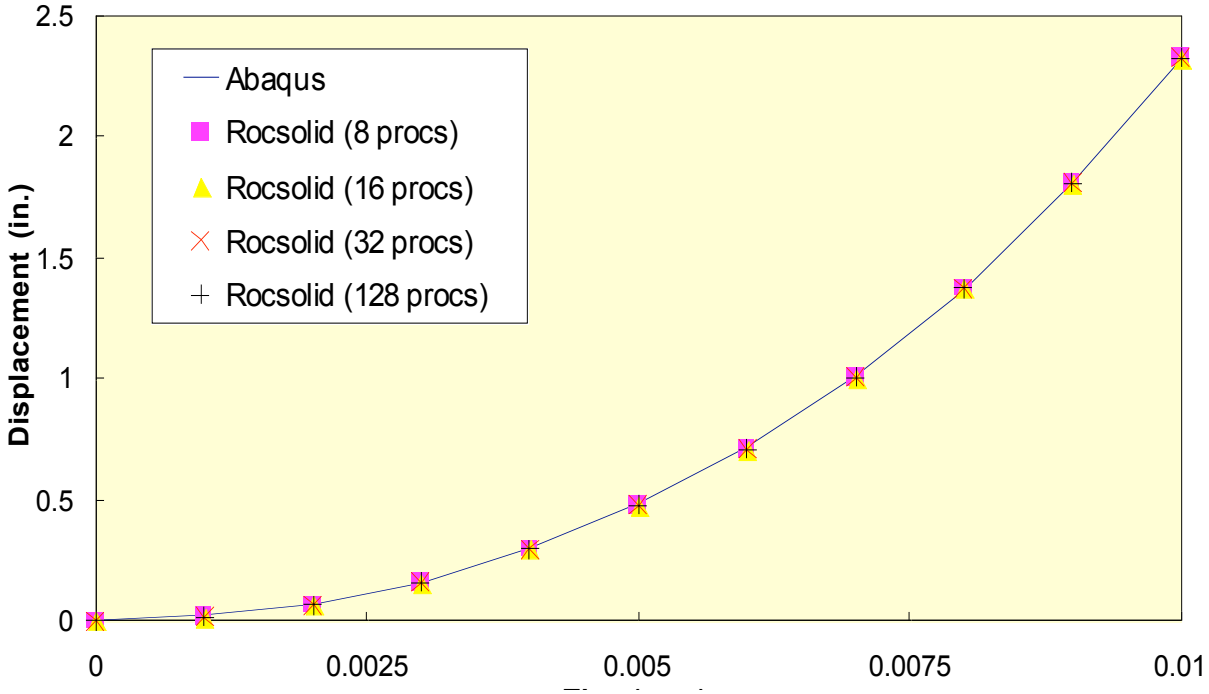
4.3.2 Problem Objective

This problem is intended to check the parallel implementation of different routines for displacement computation. The domain is decomposed into different partitions including 8, 16, 32 and 128. We should get the same result for each decomposition.



4.3.3 Problem Solution

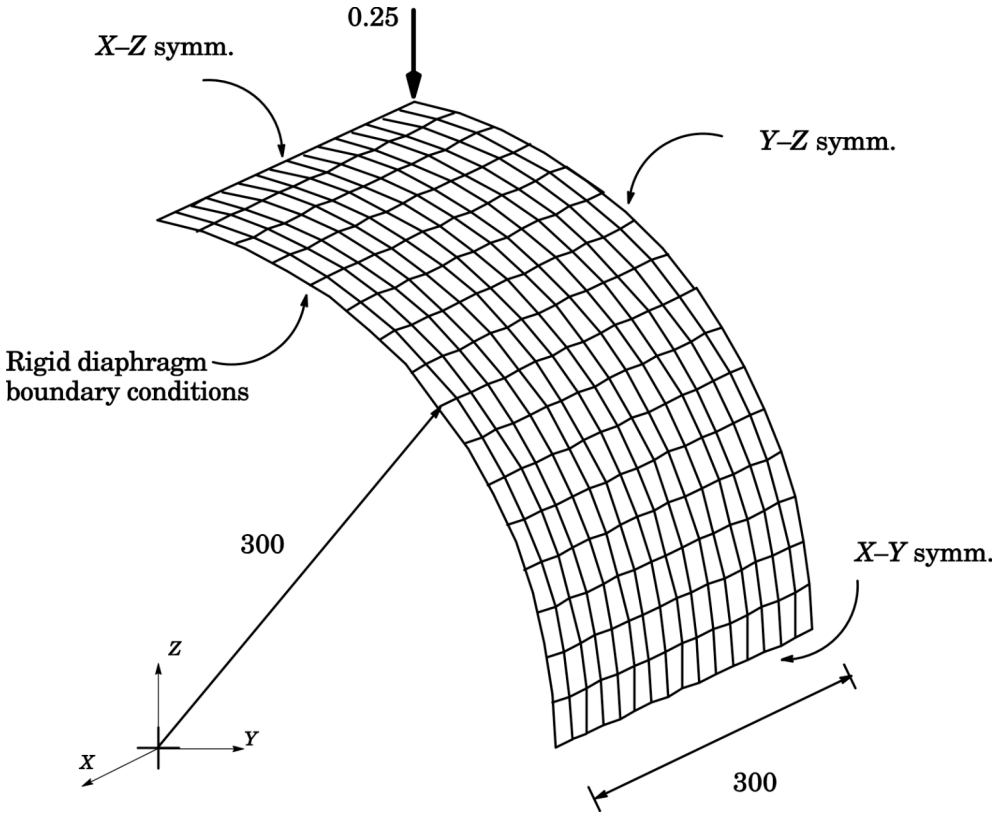
The solution (axial displacement at A) is shown in the next figure together with the ABAQUS results.



4.4 Pinched Cylinder Problem

4.4.1 Problem Description

A segment of a cylinder subjected to the loading and boundary condition shown in the next figure is analyzed using ROCSOLID's mixed enhanced elements.



4.4.2 Problem Objective

The results produced by the mixed enhanced elements are verified in this problem. The results are compared with the analytical solution and ABAQUS. Two discretizations are also considered.

4.4.3 Problem Solution

The solution (normalized vertical displacement) is shown in the following figure. The analytical solution is equal to one.

Mesh	16 × 16 × 1	32 × 32 × 1
C3D8 (ABAQUS)	0.152	0.340
C3D8I (ABAQUS)	0.838	0.975
ME9 (ROCSOLID)	0.914	0.990
S4R5 (ABAQUS Shell)	0.954	0.996

4.5 Thick Sphere under Internal Pressure

4.5.1 Problem description

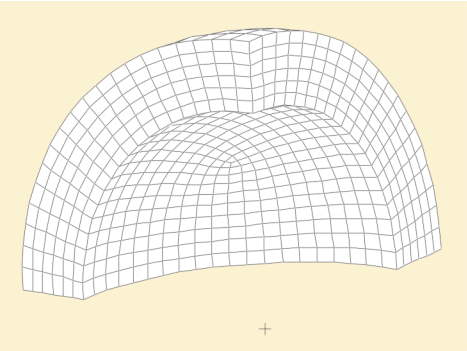
A section of a thick sphere under internal pressure is analyzed using the ROCSOLID's mixed enhanced elements. Next figure shows the mesh.

4.5.2 Problem Objective

The results produced by ROCSOLID using its mixed enhanced elements for a non-shell problem are verified in this problem.

4.5.3 Problem Solution

The solution (normalized radial displacement) is shown in the following figure together with the ABAQUS results. The analytical solution is equal to one.



Normalized Radial Displacement

ν	B8UEL (Basic Solid)	ME9 (ROCSOLID)
0.49000	0.971	0.999
0.49900	0.767	0.999
0.49990	0.247	0.999
0.49999	0.031	0.999

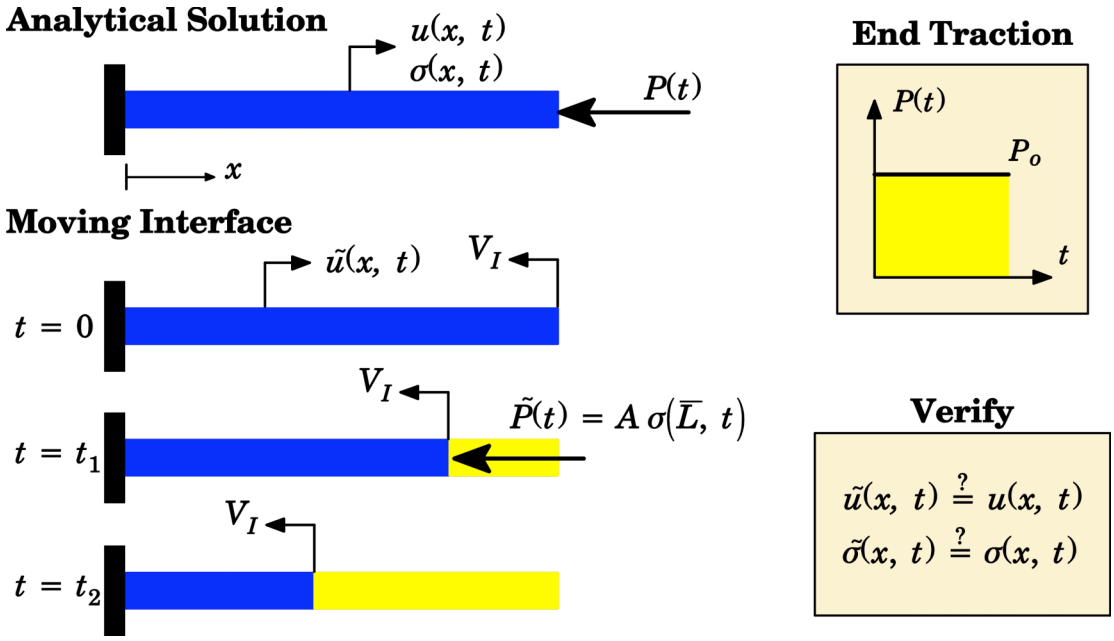
4.6 Burning Bar problem

4.6.1 Problem Description and Objective

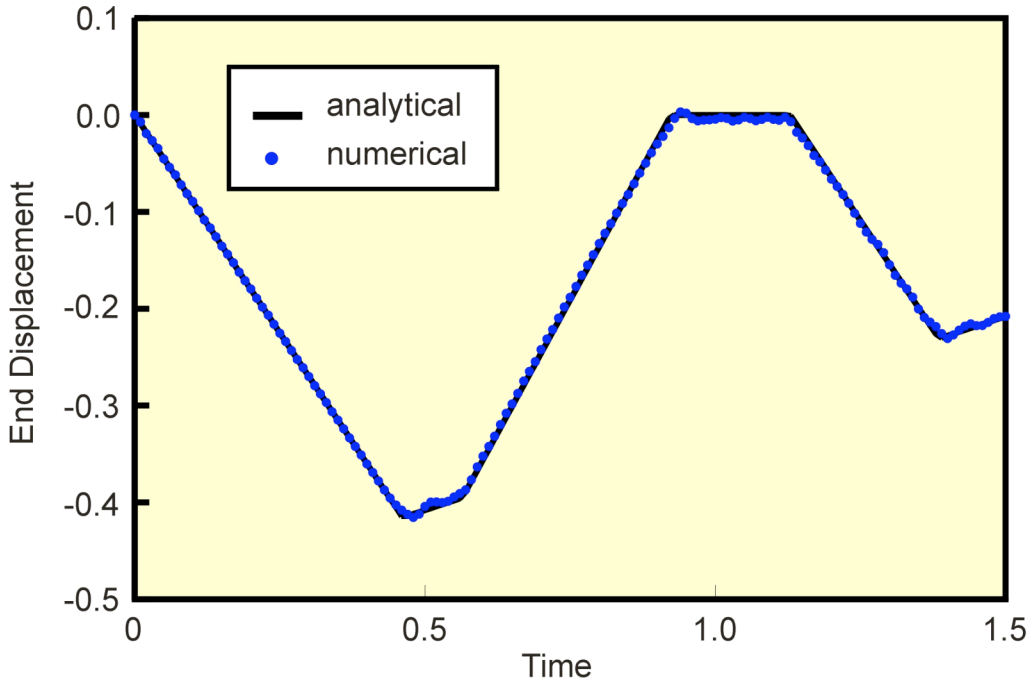
The analytical solution at each cross section along a 1-D bar subjected to pressure at one end (the interface end) is known. Using this solution, a problem can be manufactured to verify the ALE implementation in ROCSOLID. The state of stress at each cross section at any time can define the applied load that we need to use in our problem. A constant velocity is used to move the interface. We can then compare the displacements and the resulted stresses from our ALE computation with the analytical solution. Next figure explains the problem and this process further.

4.6.2 Problem Solution

The solution (end displacement) together with the analytical solution are plotted in the next page. Very good agreement is observed.



Burning Bar Problem Description

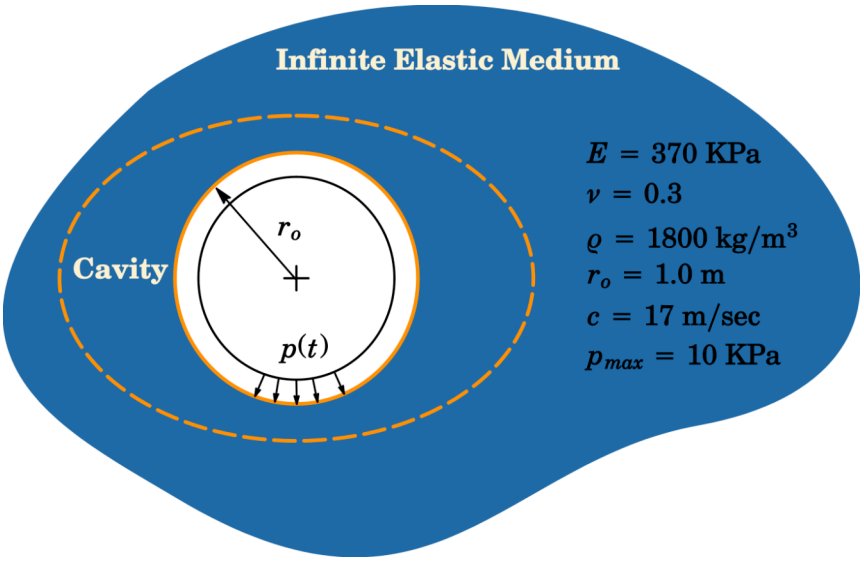


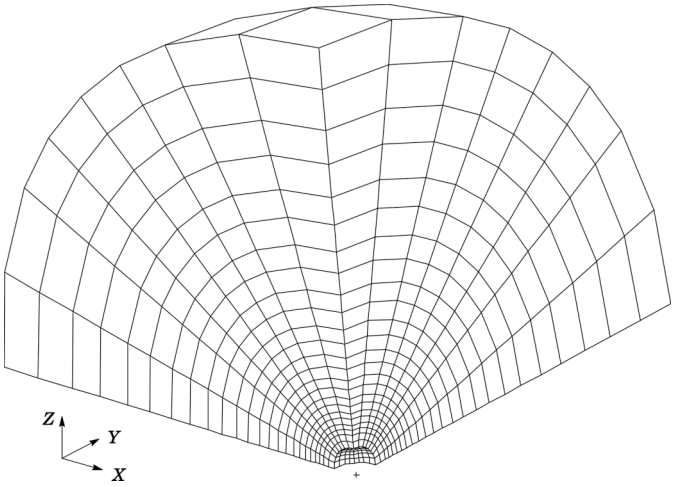
Burning Bar Numerical and Analytical Result

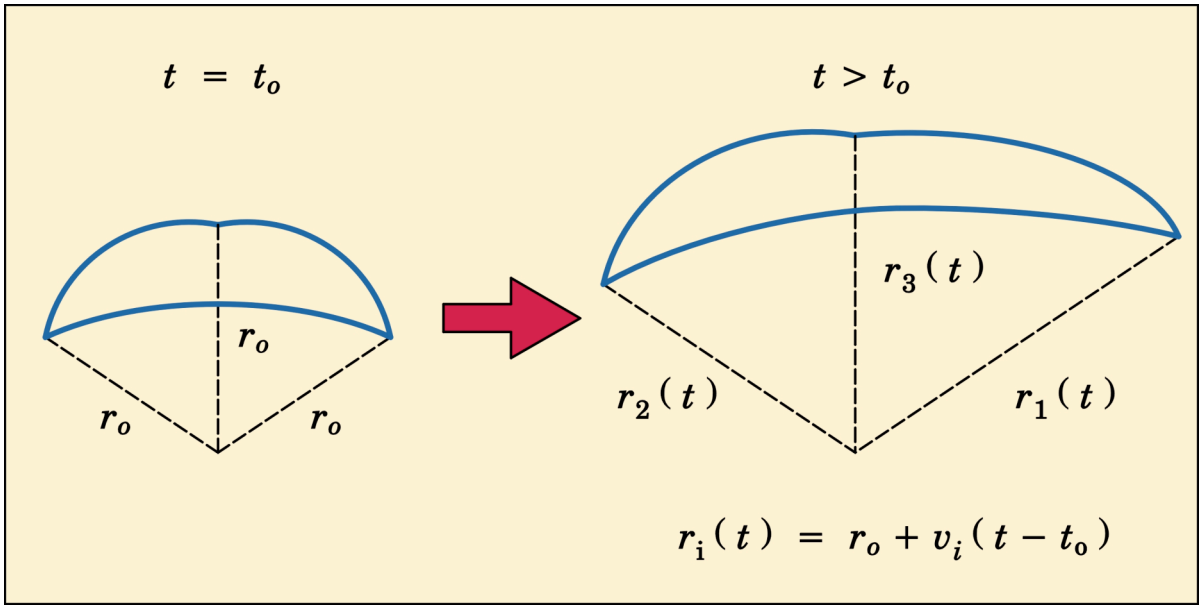
4.7 Spherical Cavity Problem

4.7.1 Problem Description and Objective

The same idea for the burning bar problem explained in the previous section can be used for a 3-D sphere with a cavity at the center. The ALE implementation to solve a 3-D problem is verified in this section. The analytical solution for an infinite elastic domain with a pressurized cavity is known. We use ellipsoidal cavity surface motion described in the third figure.



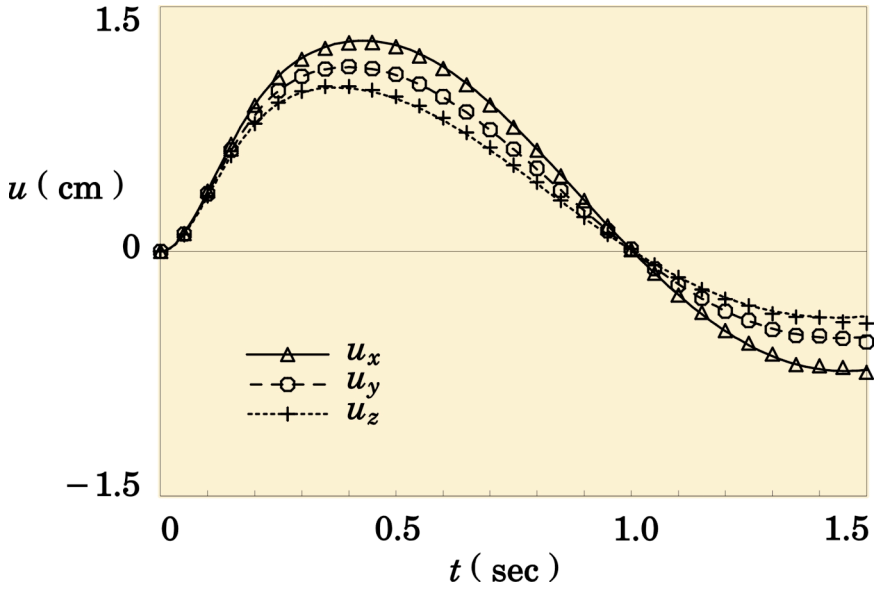




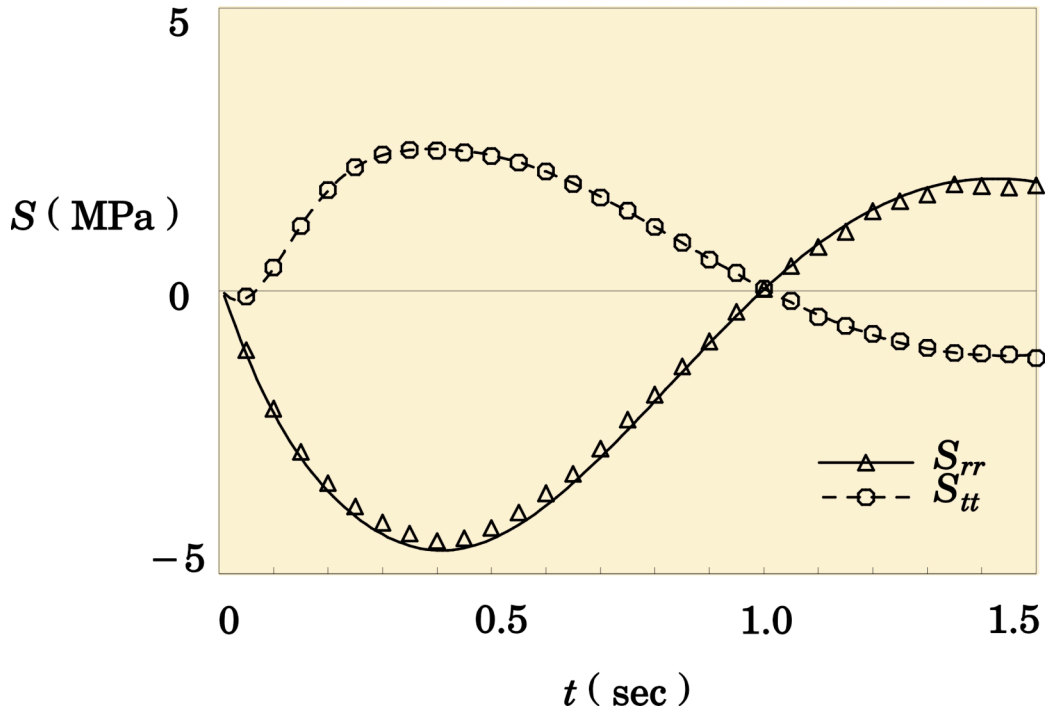
Ellipsoidal cavity surface motion

4.7.2 Problem Solution

The solutions (interface displacements and radial and tangential stresses) are shown in the following figures and compared with the analytical solution.



Interface Displacements



Radial and Tangential Stresses

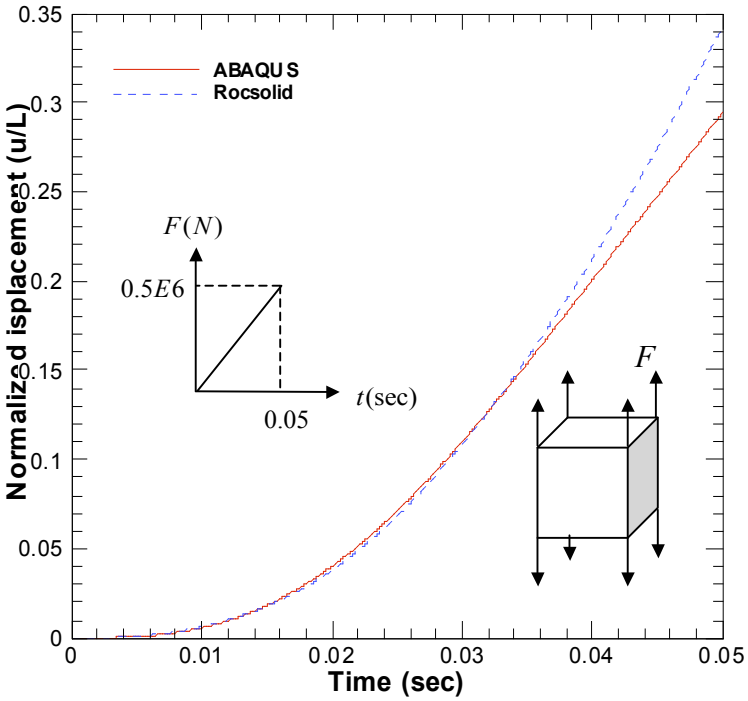
4.8 Uniaxial Tension Problem (Porous Viscoeleastic Material Model)

4.8.1 Problem Description and Objective

To verify the implementation of the porous viscoelastic material medel, a uniaxial tension problem is solved by ABAQUS using a UMAT procedure and the results are compared with ROCSOLID solution.

4.8.2 Problem Solution

The solution is shown in the next figure.



Verification: Using Abaqus UMAT