SANDIA REPORT
SAND2019-11133
Printed September 18, 2019

Sandia National Laboratories

Sierra/SolidMechanics 4.54 Verification Tests Manual

SIERRA Solid Mechanics Team
Computational Solid Mechanics and Structural Dynamics Department
Engineering Sciences Center

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

Presented in this document is a small portion of the tests that exist in the Sierra/SolidMechanics (Sierra/SM) verification test suite. Most of these tests are run nightly with the Sierra/SM code suite, and the results of the test are checked versus the correct analytical result. For each of the tests presented in this document, the test setup, a description of the analytic solution, and comparison of the Sierra/SM code results to the analytic solution is provided. Mesh convergence is also checked on a nightly basis for several of these tests. This document can be used to confirm that a given code capability is verified or referenced as a compilation of example problems. Additional example problems are provided in the Sierra/SM Example Problems Manual. Note, many other verification tests exist in the Sierra/SM test suite, but have not yet been included in this manual.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This document presents example verification results for Sierra/SolidMechanics (Sierra/SM)[1]. These are only 'example results' in the sense that the verification manual contains a small subset of the total Sierra/SM verification test suite.

1.1. OBJECTIVES

The audience for this document is rather diverse and as such we seek to both provide strong evidence of the code's 'correctness' (tending to have a more mathematical nature), and evidence that has more of a practical bent and can thus have potential utility for the analyst in defining a model (e.g., by seeing how the mesh density affects the accuracy of a contact calculation). Complete verification of Sierra/SM would be a very long-term undertaking, especially since the code is under continuous development. Oberkampf and Roy [4] note that, 'V&V are ongoing activities that do not have a clearly defined completion point, unless additional specifications are given in terms of intended uses of the model and adequacy.' As such, the current verification manual represents a snapshot of the verification tests that have been more formally documented, but it is under ongoing development to address current applications of the code.

[1] Significant verification evidence exists in other SAND reports, some problems of which are also included in the test suite.

1.2. SCOPE

To make this document more useful to some analysts, Section 1.3 contains some introductory material on verification; that section focuses more upon tests that examine convergence (which address two of the test types to be explained) than upon the other (five) test types that do not address convergence. This emphasis is not because these are the only tests that are important, but rather because they are more complex tests and thus their correct interpretation requires more explanation of issues like 'what is the effect of using a linear elastic "exact solution" when it is not an exact solution to the underlying mathematical model that the code approximates?' The intent is to provide discussions of these different issues that can be referenced by the individual test write-ups for more details. The interested reader can consult more complete treatments of the topics from textbooks such as those that influenced our work [4,5]. A much less comprehensive discussion than the textbooks is presented in a report on the initial efforts to use Sierra tools to perform verification of Sierra/SM using field responses [2], some text of which is incorporated in this introduction.

1.3. BACKGROUND[2]

Verification seeks to prove that a code is "solving the equations right," not "solving the right equations" [4,3,1]. The latter endeavor is the subject of validation. As such, verification seeks to 'prove' that a code will obtain the correct solution of the underlying mathematical model – partial differential equations with corresponding initial and boundary values that define a boundary-initial-value problem (BIVP), or equivalently the weak or variational statements of the BIVP. Of course, the code solutions are based upon approximation theories that 'reduce' the solution of our BIVP to the solution of algebraic equations amenable to computation.

1.3.1. Convergence

Two categories of tests that will be discussed below incorporate some measure of a code's ability to converge to a solution – for the best category, to the exact solution. That is, we seek to show that successive approximations with finer discretizations (mesh and/or time steps) will be increasingly closer to the exact solution[3], *i.e.*, that we have convergence. The concept of convergence has a rich mathematical foundation, but in this document we merely touch on a few basic definitions to facilitate interpretation of the verification results.

As noted above, we describe convergence as occurring when a sequence of refined numerical solutions becomes increasingly closer to the exact solution. This implies we have a way of measuring the distance between two solutions (a metric, denoted by $d(\bullet,\bullet)$). For our verification of Sierra/SM, we know that our exact solutions live in a function space with additional topological measures for size (a norm, denoted by $\| \bullet \|$) and for angle (an inner product, denoted by $< \bullet, \bullet >$). Our distance measure is then just defined in terms of the norm; that is, we measure the distance as the size of the difference between two solutions (*i.e.*, the size of the error):

$$d(u_{approx}, u_{exact}) \equiv \|u_{approx} - u_{exact}\| \tag{1.1}$$

where $u_{approx} \sim$ an approximate solution, and $u_{exact} \sim$ the exact solution. Note that the variable u in this context represents an arbitrary field, not necessarily displacement. These are just generalizations of concepts we are familiar with in three-dimensional Euclidean space.[4] If we have two vectors, one representing the exact solution and one representing the approximate solution, their difference is the error vector, and the magnitude of that vector indicates the size of the error.

The norm used for many of the verification problems is the L_2 norm of the error:

[2]For the reader familiar with verification, the only section that may be of interest is Section 1.3.2 which describes the classification of verification problems for Sierra/SM.

[3]The weaker category of "convergence tests" generally deviates from using an exact solution. The issue of using an inexact "reference solution" will be discussed further in sections below.

[4]A very brief mathematical description that provides additional context for the concepts of function spaces and convergence is presented in [2]; details were omitted but commonly used terminology was introduced. For more mathematical details on these concepts in the context of boundary value problems see, e.g., [6].

$$\|u_{approx} - u_{exact}\|_2 \equiv \left[\int_\Omega [u_{approx}(x) - u_{exact}(x)]^2 d\Omega\right]^{1/2} \tag{1.2}$$

Currently, the norm calculations are done with Encore [7] using Gaussian quadrature. For a vector- or tensor-valued quantity, the difference in each component is squared in the integrand. All results given in this document for L_2 norms of symmetric (second order) tensors are based upon 'vector' storage of the tensor components - consistent with Voigt notation and the Exodus file storage scheme. As such, the L_2 norm applied to the vector of components reduces the contribution of the off diagonal terms by a factor of 2, since symmetry is exploited to reduce the number of terms in the vector. The Encore input can be modified to yield the true L_2 norm of the tensor, but it complicates the input and the 'vector' form constitutes an equivalent norm.

For some verification tests, we seek to know not only whether the increased resolution of a refined mesh or time step produces better results, but also the rate at which these improvements are realized. For the description below, assume the refinement is in the mesh (*i.e.*, spatial). Ideally the error in the approximation will satisfy a theoretically derived relationship of the form

$$\|e_h\| = \|u_h - u_{exact}\| = ch^p + O(h^{p+1}) \tag{1.3}$$

for some constant c, where p denotes the theoretical rate of convergence, h is a measure of the element size, u_h denotes the approximate solution for h, and e_h denotes the error vector associated with h. When we apply the above ideas to quantities of interest, like a beam tip displacement, the tensors become scalars and we use an absolute value for the norm. Note that until h becomes sufficiently small, the higher order terms on the right-hand side of Equation (1.3) can affect the observed rate of convergence when evaluating a sequence of approximations. As h decreases, the right hand side of Equation (1.3) asymptotically approaches the first term, ch^p. When h is sufficiently small for this first term to dominate, the approximate solutions are described as being in the *asymptotic range*, and thus the theoretical rate of convergence, p, is often referred to as the *asymptotic rate of convergence*. In the V&V literature, the rate obtained from theoretical analysis is also referred to as the *formal order of accuracy* [4]. In the literature for finite element methods, it is also often referred to as the *optimal convergence rate*, or just the *convergence rate*. In a later section, we will describe how an observed convergence rate is measured from a sequence of numerical solutions.

1.3.2. Types of Verification Tests

Several types of tests are used in verification, and authors group them differently. For the verification of Sierra/SM, we have adopted the following types of tests. The list of test types is nominally presented in an order ranging from simplest to most complex, with the most complex tests often being considered to be the most rigorous (with respect to being able to reveal subtle code errors[5]).

[5]Note that in referring to the error as "subtle," we are not implying that its affect in an analysis would necessarily be insignificant but rather that the source of the error in the coding is not obvious.

1. **Conservation test** - checks the conservation of physical quantities such as mass, momentum, and energy.
2. **Symmetry test** - checks the preservation of symmetry (symmetries).
3. **Sanity check** - determines if a qualitative 'sanity' understood for a test is preserved. An example would be inertial reference frame invariance, *i.e.*, objectivity tests.
4. **Code-to-code benchmark test** - compares results of one code to another code that was previously verified.
5. **Discretization error test** - compares a single numerical analysis to an analytical solution. The analytical solution may or may not be exact. A common solid mechanics test of this type is the patch test, where the reference solution is a constant stress/strain result.
6. **Convergence test** - loosely demonstrates the proper order of convergence at best, or at least demonstrates a tendency to converge to a solution with mesh and/or time step refinement.
7. **Error quantification test** - generates empirical evidence that the code can enter the asymptotic regime and that the computational error trends toward zero (with mesh or time step refinement). This category of test requires the exact analytical solution. They are also referred to as order-of-accuracy tests.

A balanced verification suite would contain tests from each category. The first four categories are rather straightforward and will not be discussed further in this document (see [4] or [5] for additional discussion). As a generic term to address tests that examine the rate of convergence, we will call these tests *convergence-rate tests*. Category (6) tests may be convergence-rate tests, and category (7) tests are always convergence-rate tests.

1.3.2.1. Reference Solutions

Before discussing the next three categories, we will clarify what we mean by the 'exact analytical solution' versus simply 'an analytical solution'. Generically, we will refer to any solution used to measure the correctness of numerical solutions as the reference solution. To evaluate the correctness of the code rigorously, we need to have a *reference solution* that is the exact analytical solution to the mathematical model that the code approximates (in our case, usually the weak statement of the underlying BIVP). This consistency is a key point, because if the analytical solution is for a mathematical model to a "near by problem"(*i.e.*, a *surrogate solution*) the verification is weaker.

The common case of adopting a surrogate solution, for solid mechanics, is the use of analytical solutions for linear elasticity problems. Obviously this is the class of solid mechanics problems for which many closed form solutions exist. In the case of Sierra/SM, an analytical solution for linear elasticity problems is not an exact solution to the underlying mathematical model, because the code inherently addresses finite deformations, yielding a nonlinear strain-displacement relationship and enforcement of equilibrium in the deformed configuration; linear elastic response can only be obtained in the limit (of infinitesimal displacements). Even when the goal is to examine how well the code performs for a problem governed by linear elasticity, the underlying nonlinearities can complicate the comparison, because there is the potential for these

nonlinearities to affect the perceived "error"[6] As such, this issue is most evident for a highly accurate solution where the first significant digit of the error corresponds to many digits into the floating-point word representing the response quantity. For convergence tests, the issue is more important (usually for finer the meshes) and will be discussed further below.

Of course, a challenge is that there are far fewer analytical solutions that are consistent with the underlying mathematical model of Sierra/SM, *i.e.*, for problems with finite deformations. If we consider problems that have other sources of nonlinearities (like contact and nonlinear constitutive behavior) and complexity (like integral operators, e.g., to characterize path dependence), the chance of obtaining an exact analytical solution is reduced further. While we have started to apply the manufacturing of solutions for problems with finite deformations, extension to problems with contact and material models defined in an incremental manner needs further development.

1.3.2.2. Discretization Error Tests

Note that our current verification test suite is dominated by discretization error tests. This is a rather natural state, since these tests can offer a good balance between verifying the code correctness (or quality) and the investment required to develop the test. These tests are also easy for an analyst to relate to since the comparison of two solutions is often limited to a tabular or graphical representation of quantities of interest or their "errors" (e.g., a patch test stress state or a load-deflection curve), and the problems are physically meaningful. The reference solution in this case, while analytical, may not be the exact solution. Unfortunately these tests address accuracy alone, and it is often difficult to assess if a level of accuracy is acceptable for a given discretization. As such, these tests can reveal major code errors but are less useful at revealing subtle code errors that error quantification tests can reveal.

1.3.2.3. Convergence Tests

These tests are the weaker of the tests that yield information on convergence. The source of their weakness is typically either that they: (1) adopt an inexact reference solution; or, (2) demonstrate a tendency to converge without reference to another solution. One type of convergence test that the verification test suite adopts is an asymptotic analysis to estimate the rate of convergence. This will be discussed more below, but it can be thought of as adopting an inexact reference solution, since the analysis follows the asymptotic approach of Richardson's extrapolation and obtains an estimate of the exact solution that is one order higher than the numerical analysis. While a test in this category may indicate a tendency to converge, and may even loosely demonstrate convergence at the proper order of convergence, it does not show that the convergence is to the exact solution; we can only say the approximation appears to be converging to a solution. Detail on the characteristics of convergence tests adopting a surrogate reference solution or using asymptotic analysis will be discussed more in separate sections that follow.

[6] Technically, any difference between an inexact reference solution and a numerical solution is not an **error**, but herein this reference is occasionally made, and it should be interpreted as a **difference**.

1.3.2.4. Error Quantification Tests

This category of tests contains the strongest tests for convergence. They adopt an exact analytical reference solution, consistent with the underlying mathematical model of the code and clearly demonstrate the ability of the code to exhibit the asymptotic rate of convergence (with mesh or time step refinement). While mathematical proofs of convergence generally address measures of errors in fields, we include both field and quantity of interest measures of convergence in this category.

1.3.3. Observed Convergence Rate

Verification has been referred to as being an inherently empirical process [4] in the sense that we seek, via numerical experiments, to determine if the code is "behaving correctly." In the context of order-of-accuracy tests, we seek to show that in the asymptotic range, the observed rate of convergence matches that theoretically predicted. If so, confidence is increased that the code is correctly approximating the underlying mathematical model. In the V&V literature, the rate inferred from multiple numerical analyses with different levels of discretization is referred to as the *observed order of accuracy* [4]. In this document, we will tend to use the more common FEM phrases, *observed convergence rate* or just *convergence rate* (in the latter case, the distinction between the theoretical rate and observed rate is determined by the context).

To estimate the convergence rate from multiple finite element analyses, we assume that Equation (1.3) is valid, and that the $O(h^{p+1})$ terms are not significant, *i.e.*, that we are obtaining the asymptotic rates. Taking the log of both sides of the asymptotic part of Equation (1.3) gives

$$\log(\|e_h\|) = \log(c) + p\log(h). \tag{1.4}$$

Thus on a log-log plot of error versus element size, the slope of the line gives the observed rate of convergence. Often the results for the coarser meshes are not in the asymptotic range, and then the slopes obtained by sequences of results from two meshes changes, giving more accurate convergence rates with finer meshes. For two results from a FEA where the exact solution is known, we can estimate this convergence rate by comparing the errors from these two meshes, and solving for p. For the problems that follow, most refinements involve halving the element size, h, which leads to the following relation for estimating the convergence rate:

$$p = \log(\|e_{h/2}\|/\|e_h\|)/\log(1/2) \tag{1.5}$$

where $e_{h/2}$ is the corresponding error for uniform (half-size) mesh refinement. When we have multiple levels of refinement, we could apply linear regression to all of the results on a log-log plot obtaining the rate of convergence over a larger range of meshes; however, obtaining rates of convergence from sequences of two results provides an indication of the extent to which the results are in the asymptotic range.

The above discussion of observed rate of convergence is based upon the assumption that we have the exact solution and is thus applicable to error quantification tests. For tests that fall in the

convergence test category, we usually do not have the exact solution, but we may still seek to estimate the rate of convergence. If so, we attempt to obtain the rate of convergence either by using a surrogate solution or by using asymptotic analysis. Both approaches can provide more confidence in the code correctness for some problems, but they also have limitations that will be discussed below.

1.3.4. Convergence Tests using a Surrogate Solution

As previously noted, a surrogate solution is not an exact solution to the underlying mathematical model that the code approximates, but rather is the solution of a "nearby problem". As such, the surrogate solution has *mathematical modeling errors* due to the differences in the problem it solves. The surrogate solution may be useful for estimating the rate of convergence when the numerical modeling errors are greater than the mathematical modeling errors, that is, for sufficiently coarse meshes it provides an accurate surrogate for the exact solution. When the converse is true (*i.e.*, the mesh is relatively fine), strictly speaking we are faced with the uncertainly of whether the difference in solutions is due entirely to the inexactness of surrogate solution, or due to a subtle error in the implementation that verification is designed to reveal. Unfortunately for coarser meshes where the surrogate solution is sufficiently close to the exact solution, the numerical solution may not be in the asymptotic range. As such, for a surrogate solution to be useful in estimating the rate of convergence, we need a range where the surrogate is sufficiently accurate and the numerical results are in the asymptotic range. For some sequences of solutions, this range will not even exist.

For the case of an exact reference solution, one expects the code to yield the asymptotic rate with increasing accuracy upon mesh refinement. For the case of a surrogate reference solution, if the FEM solution is approaching the exact solution, one would expect the difference to approach a constant value that quantifies the mathematical modeling error. That is, in the limit, the difference is an indicator of the error in the surrogate solution, not the FEM solution. Generally however, we do not know that the FEM solution is approaching the exact solution, so the constant difference that the FEM solution approaches could be a combination of code error and mathematical modeling error.

Another characteristic of using a surrogate reference solution is that the convergence to the constant difference is not necessarily monotonic. This is true for field quantities and quantities of interest, and can be illustrated in terms of a solution in a function space or on the real line, respectively. For simplicity, consider a description for a quantity of interest, the values of which are on the real line. Assume for example that our quantity of interest is a force response, and that the exact solution for the response is 1000. Also assume that the surrogate solution gives a force response of 1001. If the sequence of results from the FEM starts at 1400 (a 40 percent error) and monotonically decreases toward the exact solution, apparent non-monotonic convergence can be obtained relative to the surrogate solution. Note that at a load level of 1400, the surrogate solution provides a reasonable measure of the error (~39.8%). For example, consider a sequence of FEM force predictions having linear convergence given by {..., 1006, 1003, 1001.5, 1000.75, 1000.375,...} and that are converging to the exact solution. The actual sequence of errors are simply {..., 6, 3, 1.5, 0.75, 0.375,...} which exhibits monotonic convergence. However, the

perceived ‘errors’ (actually differences) obtained relative to the surrogate solution are {..., 5, 2, 0.5, 0.25, 0.625,...} – not monotonic; these differences approach a difference value of 1 – the mathematical modeling error. Note that if the surrogate solution gave a value of 999, the convergence would be monotonic, so the relative values of the exact and surrogate solutions can determine the nature of the convergence.

1.3.5. Convergence Tests using Asymptotic Analysis

Asymptotic analysis is used in convergence tests when we are seeking an estimate of the rate of convergence, sometimes when we have a surrogate solution and other times when we do not. We tend to use them when we have a surrogate solution, for cases where we do not have an obvious range for estimating the rate of convergence. As such, it can strengthen the convergence argument, though it is still weaker than having an error quantification test. When we do not have any surrogate solution, it provides an estimate of the rate of convergence when otherwise we could only observe a tendency of the results to converge to some value (hopefully the exact solution); in this later case, it is being applied identically as one does for solution verification.

The asymptotic analysis can be considered as consisting of two steps. First, the results from sequences of analyses based upon three mesh refinements, where each refinement halves the characteristic length of the element (e.g., each hex is approximately subdivided, into eight hex elements), are used to estimate the rate of convergence. Second, the convergence rate obtained from the finest sequence of meshes may be assumed to be accurate, and then is used with Richardson extrapolation to obtain a higher-order estimate of the exact solution. The Richardson’s extrapolated estimate is then often adopted as the reference solution to analyze the results, sometimes with log-log plots of a "difference measure" versus an element size measure as would be done with an analytical reference solution. For problems like contact, where we cannot define the expected rate of convergence exactly, we have chosen to use the rate obtained in the first step as the rate applied in the second step - essentially solving three equations for three unknowns (as we will outline below), but the rate is not an integer. The alternative is to use the rate obtained in the first step to provide an estimate of the rate and round it to the next integer. If the formal rate of convergence is known for the numerical method, that rate should be used in the Richardson extrapolation.

Consider an outline of the asymptotic analysis as two steps. For more detail, see references [4] or [5]. First consider a sequence of three scalar results for a quantity of interest or norm of a field that will be denoted as $\{S_i, S_{i+1}, S_{i+2}\}$, where S_i denotes the scalar value for the coarsest mesh, and S_{i+1} and S_{i+2} denote the scalar values for one and two uniform mesh refinements, respectively; consistently, these results correspond to meshes such that $h_{i+1} = h_i/\mathrm{r}$ where $r = 2$.[7] As with Equation (1.3), we assume that error can be expressed in a power series in h, as

$$S_i = S_{exact} + ch_i^p + O(h_i^{p+1}) \tag{1.6}$$

for a p^{th}-order method. Combining the higher order error terms with the exact solution gives

[7] Uniform mesh refinement as specified here is not a requirement of the methodology, but it is the approach that has been adopted for all problems in the manual to date.

$$S_{RE} = S_{exact} + O(h_i^{p+1}) \tag{1.7}$$

where S_{RE} denotes an approximation of the exact solution that, if the h_i^{p+1} term exists, is one order higher in accuracy than the original p^{th}-order method would give. The notation of the *RE* subscript denotes the Richardson Extrapolated value for S, which will be solved for in the second step of the analysis. Combining Equations 1.6 and 1.7 gives

$$S_i = S_{RE} + ch_i^p. \tag{1.8}$$

If we write this relationship for meshes i, $i+1$, and $i+2$, we have three equations and three unknowns. Eliminating S_{RE} and c from the three equations and solving for p gives

$$p = \frac{\ln(\frac{S_1 - S_2}{S_2 - S_3})}{\ln(r)}. \tag{1.9}$$

Note that the above analysis can be used for successive sequences of three meshes, as is done for many verification problems, and consistency in the results for p then gives an indication if the results are in the asymptotic range. If the results are not consistent, the asymptotic analysis is not conclusive, though the result from the finest set of meshes may suggest a tendency in the rate of convergence.

The second step of the analysis corresponds to the generalized Richardson extrapolation, where the extrapolated value is given by

$$S_{RE} = S_3 + \frac{S_s - S_2}{r^p - 1}. \tag{1.10}$$

As previously noted, this value can now be used as a reference solution. We have done that in many of the tests to give a graphical representation of the results from the asymptotic analysis. These types of graphical results must be interpreted carefully, because in most cases S_{RE} is considered to be more useful as an indicator of the uncertainly in the solution than as a proper surrogate solution. Use of this solution, when p is obtained directly from Equation (1.10), also tends to instill false confidence in the results; this is due to the fact that by definition the convergence plot will show the results for the finest three meshes as lying perfectly on a straight line. When this occurs with an exact solution, we infer that we are in the asymptotic range; when it occurs in this case it is simply a result of solving the corresponding three equations to make it occur. In this case, we need four values to lie along a line, which corresponds to getting consistent p estimates from two overlapping sequences (as previously mentioned). Another caveat in plotting the results for these tests is that when multiple tests are plotted on the same plot we have to keep in mind that they each have their own reference solution, so comparisons of relative accuracies can be questionable - though potentially meaningful if we know the extrapolated results are based upon data from the asymptotic range.

1.4. MANUAL ORGANIZATION

The remainder of the Sierra/SM Verification Tests Manual is divided into chapters that represent related capabilities. Each section of a chapter represents a distinct verification "test group." In some cases, the test group contains a single test, and in other cases it contains a group of related tests (e.g., a patch test applied all of the hex elements). The verification test groups listed in each chapter verify some aspect of that suite of capabilities. Some of these verification tests are run nightly by the development team to continually verify code solution quality. Other tests that are too computationally demanding to be run nightly are tested before each code release. The graphics and charts in this document are automatically generated by the test runs. The test files for these problems may be found in the Sierra regression test repository, in the sub-directory

```
adagio_rtest/VerificationTestManual
```

On many Sierra-supported platforms, the latest versions of these tests can be accessed at

```
/sierra/dev/nightly/Sierra.tests.master/adagio_rtest/VerificationTestManual
```

1.5. REFERENCES

1. F. G. Blottner, "Navier-Stokes results for the hypersonic flow over a spherical nosetip", *Journal of Spacecraft and Rockets*, 27(2), 113-122, 1990.
2. J. V. Cox and K. D. Mish, "Sierra Solid Mechanics Example Verification Problems to Highlight the use of Sierra Verification Tools", *SAND Report*, SAND2013-2390, Sandia National Laboratories, 2013.
3. B. W. Boehm, **Software Engineering Economics**, Prentice-Hall, 1981.
4. W. L. Oberkampf and C. J. Roy, **Verification and Validation in Scientific Computing**, New York, NY: Cambridge University Press, 2010.
5. P. J. Roache, **Verification and Validation in Computational Science and Engineering**, Albuquerque, NM: Hermosa Publishers, 1998.
6. I. Stakgold, **Green's functions and Boundary Value Problems**, New York, NY: John Wiley & Sons, 1979.
7. K. D. Copps and B. R. Carnes, "Encore User Guide", *SAND Report*, DRAFT, Sandia National Laboratories, 2009.

2. CONTACT VERIFICATION TESTS

The following are tests that verify aspects of our contact capabilities. These tests cover contact contact input definition, constraint creation, contact equation solution, and friction model behavior. These tests span the full Sierra/SM solution spaces of explicit transient dynamics, implicit transient dynamics, and implicit quasi-statics.

2.1. CONTACT FORCE BALANCE

Analysis Type	Quasi-statics (Adagio)
Element Type	Hex8
Strain Incrementation	Strongly Objective
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Contact Force
Number of Tests	1
Keywords	Force Balance

2.1.1. Problem Description

This test checks that the computed nodal forces are balanced (in equilibrium). It is composed of a unit cube sitting on top of a larger block. The unit cube block has an applied pressure on the top surface while the bottom block is held fixed.

2.1.1.1. Boundary Conditions

The applied pressure on the top surface of the unit cube is ramped in a sinusoidal manner to 1000 psi (a force of 1000 lbs) while the bottom block is held fixed in all directions.

2.1.1.2. Material Model

Each block uses an elastic material model. The parameters are shown in table 2-1. The parameters were simply chosen for convenience.

Table 2-1. Elastic Material Properties

Young's Modulus	E	$30e6$ Psi
Poisson's Ratio	ν	0.3
Density	ρ	100.0 lbf sec^2/in^4

2.1.1.3. Feature Tested

The balance of nodal contact forces.

2.1.2. Assumptions and notes

This problem assumes that the deformation does not significantly affect the loaded area and thus magnitude of the load.

2.1.3. Verification of Solution

The prescribed pressure force serves as the analytic value. The summed y nodal contact force on the top block should be equal and opposite in sign to the pressure force. The sum of the nodal contact forces in the y direction on the bottom block should be equal to the pressure force. Finally, the summed y reaction force on the bottom block should be equal and opposite in sign to the pressure force. These collected forces can be seen in the figure.

All values are balanced in the final time steps to 0.05% error.

For input deck see Appendix B.1.

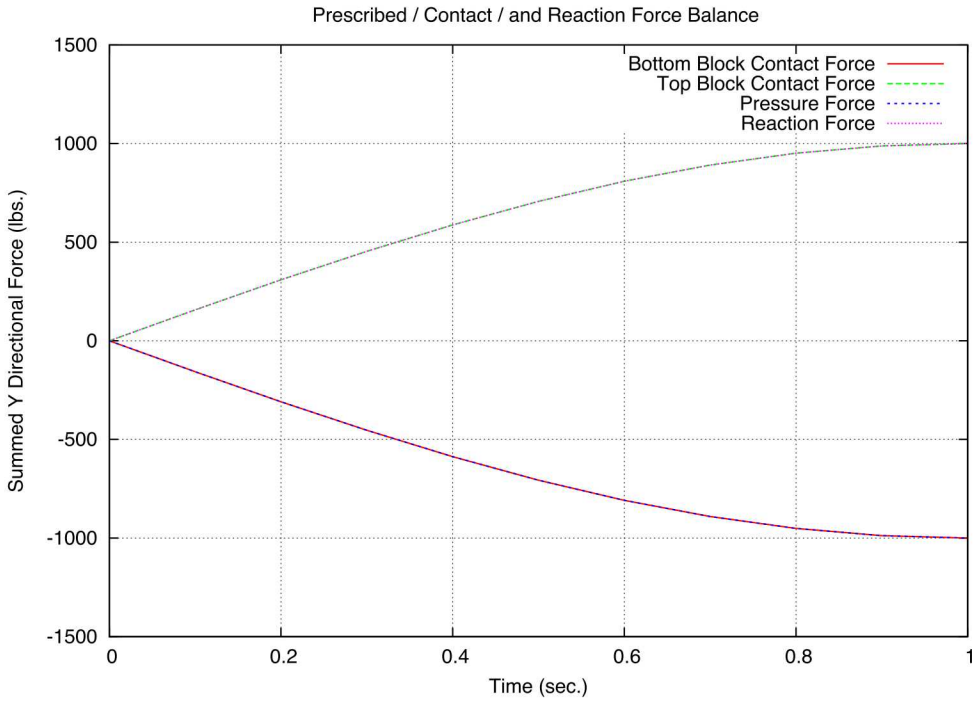


Figure 2-1. Force Balance

2.2. HERTZ SPHERE-SPHERE CONTACT

Analysis Type	Quasi-statics (Adagio)
Element Type	Hex8
Material Model	Elastic
Keywords	Contact

2.2.1. Problem Description

This problem presses an elastic sphere into a rigid plate and compares the resulting contact radius and the maximum sphere deformation to analytic predictions from Hertzian contact theory.

2.2.1.1. Boundary Conditions

The boundary conditions are illustrated in the Figure 2-2.

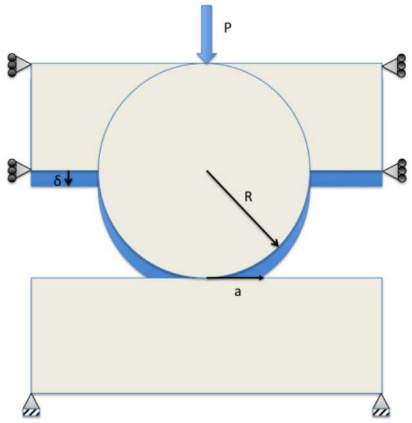


Figure 2-2. Elastic Sphere on Rigid Plate Problem Setup

2.2.1.2. Material

The sphere's elastic material parameters can be found in Table 2-2.

Young's Modulus	E	$68.9e+9$
Poisson's Ratio	ν	0.33
Density	ρ	$1.024e-6$

Table 2-2. Elastic Material Properties

2.2.1.3. Feature Tested

The augmented Lagrange node-Face contact algorithm for fricitionless contact is tested and compared to an analytic solution.

2.2.2. Assumptions and notes

The assumptions for this problem match those of Hertzian contact problems. The strains are assumed small and within the elastic limit. The radius of contact is much smaller than the characteristic radius of the body. The surfaces are frictionless, continuous, and non-conforming.

2.2.3. Verification of Solution

The analytic solution based on Hertzian contact for the contact radius (a) and the resulting deflection (δ) as illustrated in the Figure 2-2 are given by Equations 2.1 and 2.2.

This problem ran has a sphere of radius $R = 1.0$, an applied load of $P = 5.0e7$ and an elastic modulus of $E = 68.9e + 9$.

$$a = \left(\frac{3PR}{4E}\right)^{\frac{1}{3}} \tag{2.1}$$

$$\delta = \frac{a^2}{R} \tag{2.2}$$

The percent error is computed by Equation 2.3 for the contact radius and the deflection.

$$\%Error = \frac{|Analytic - Computed|}{|Analytic|} * 100.0 \tag{2.3}$$

The contact radius is within 0.5% of 4.5% error and the deflection is within 1% of 9.75% error.

Figure 2-3 shows the contact pressure in the compressed region of the sphere where the contact radius is computed.

For input deck see Appendix B.2.

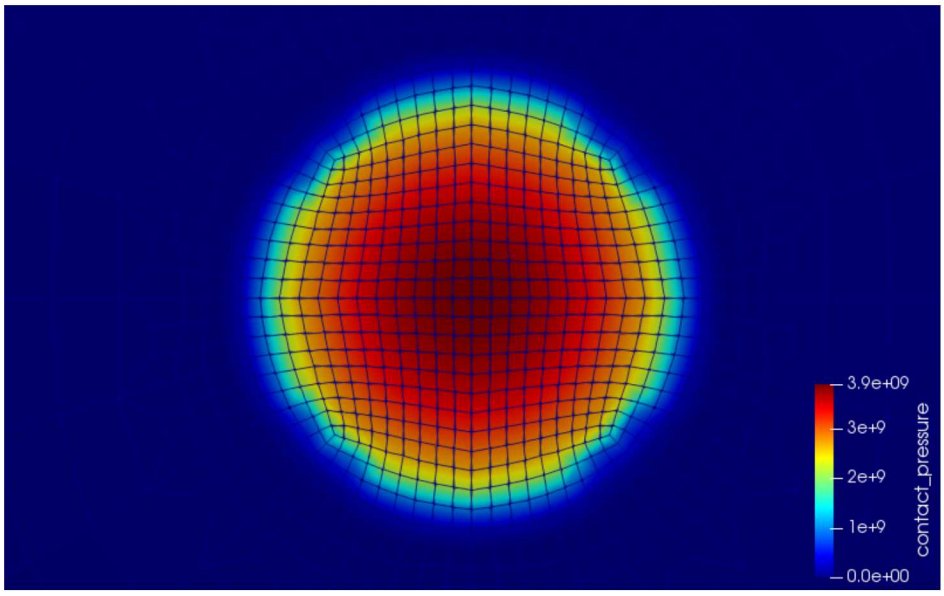


Figure 2-3. Contact Pressure on Compressed Region of Sphere

2.3. DERESIEWICZ SPHERE-SPHERE CONTACT

Analysis Type	Quasi-statics (Adagio)
Element Type	Hex8
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Torque
Number of Tests	4
Keywords	Contact

2.3.1. Problem Description

This problem presses two elastic spheres together, and then applies a rotational motion to them; see Figure 2-4 and references [1, 2]. Since the spheres are in frictional contact, the rotation requires the application of a twisting moment. Due to symmetry considerations, only one-half of one sphere need be simulated; see Figure 2-5.

2.3.2. Exact Solution

Denote by M the twisting moment applied, N the contact normal force, a the contact radius, μ the coefficient of friction, G the shear modulus, and β the angle of twist. Define the non-dimensional angle

$$\theta := \frac{\beta G a^2}{\mu N} , \tag{2.4}$$

and the non-dimensional torque

$$T := \frac{M}{\mu N a} . \tag{2.5}$$

There exists an exact solution, in terms of elliptic integrals, relating the dimensionless parameters T and θ [1, 2]. In reference [3] the authors approximate the exact solution with the rational function

$$T(\theta) = \left[\sum_{k=0}^{4} a_k \theta^k \right] \left[\sum_{k=0}^{4} b_k \theta^k \right]^{-1} . \tag{2.6}$$

The parameters are given in Table 2-3. Equation (2.6) is convenient for numerical evaluation and can be used for verification purposes.

2.3.3. Numerical Solution

This problem can be simulated using Adagio. The code can output the applied twisting moment M and normal force N as a function of time. The twisting angle β and contact radius a as a function of time can also be computed with user defined functions.

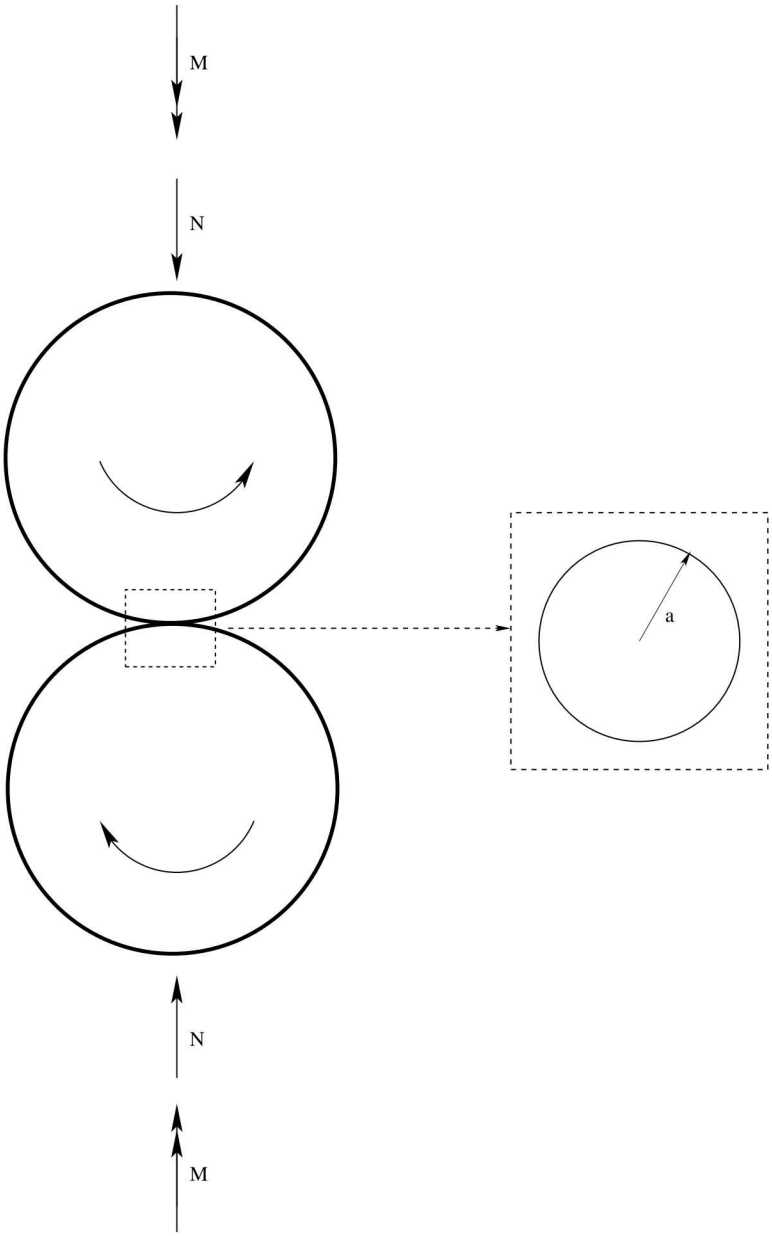


Figure 2-4. Two spheres pressed together and subjected to a torsional couple. N is the normal force, M is the applied moment, and a is the radius of contact.

2.3.4. Verification

The results of non-dimensional torque versus time can be compared for the exact (2.6) and numerical solutions. One can compute an L^1 integrated in time error, if desired. Typical results are shown, for example, in Figures 2-6 and 2-7.

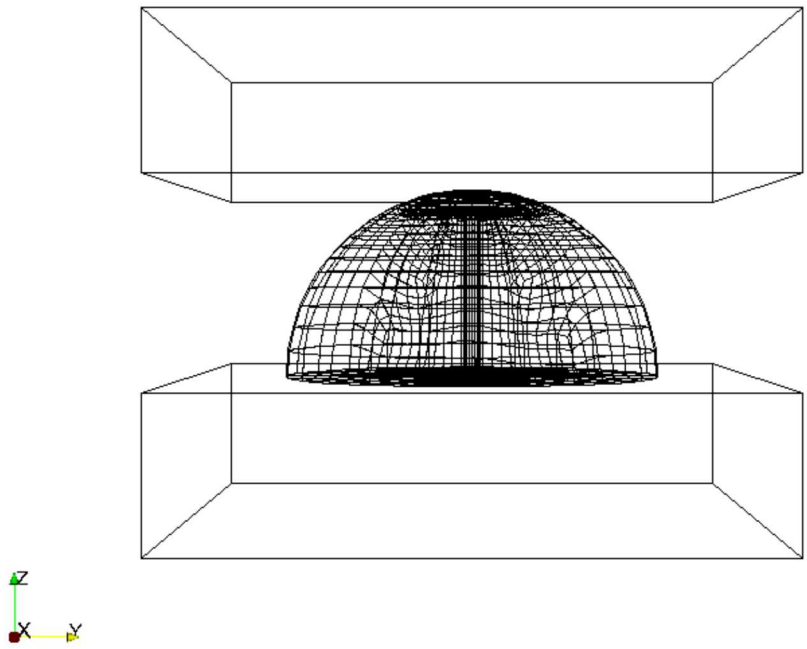


Figure 2-5. One-half sphere used for computational simulation.

Table 2-3. Padé approximation data

a_0	0	b_0	1
a_1	16/3	b_1	5.1193
a_2	6.0327	b_2	15.6833
a_3	19.6951	b_3	30.8099
a_4	42.5359	b_4	72.2111

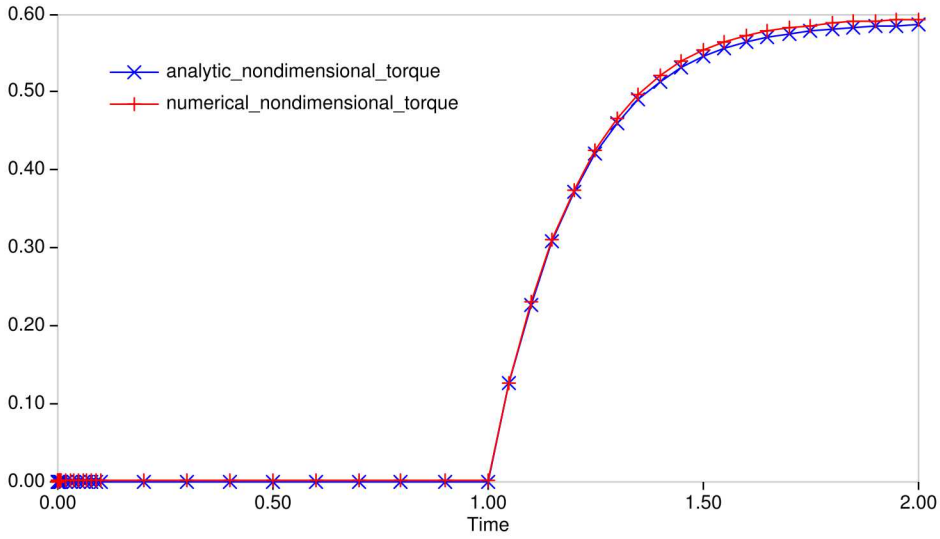


Figure 2-6. Non-dimensional torque versus time.

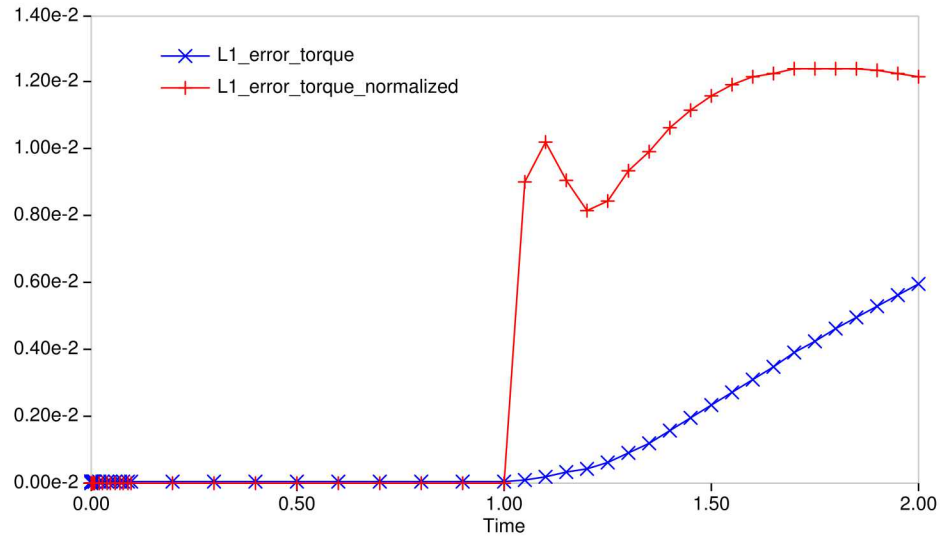


Figure 2-7. Non-dimensional torque error versus time.

2.3.5. References

1. H. Deresiewicz *Contact of Elastic Spheres Under an Oscillating Torsional Couple*, *ASME Journal of Applied Mechanics*, **21**, 1954, pp. 52-56.
2. L. Lubkin, *The Torsion of Elastic Spheres in Contact*, *ASME Journal of Applied Mechanics*, **18**, 1951, pp. 183-187.
3. Daniel J. Segalman, Michael J. Starr and Martin W. Heinstein, *New Approximations for Elastic Spheres Under an Oscillating Torsional Couple*, *ASME Journal of Applied Mechanics*, **72**, September 2005, pp. 705-710.

For input deck see Appendix B.3.

2.4. HERTZ CYLINDER-CYLINDER CONTACT – CONVERGENCE TEST

Analysis Type	Quasi-statics (Adagio)
Element Types	Hex8
Element Formulations	Mean Quadrature, Fully Integrated
Strain Incrementation	Strongly Objective
Material Models	Elastic
Verification Category	Convergence
Verification Quantities	Boundary Displacement (δ), Contact Force (P)
Number of Tests	4
Keywords	Hertz, Contact, Convergence

2.4.1. Brief Description

This series of analyses demonstrates the convergence of contact for a classical Hertz problem. This problem is a quasistatic version of the (inactive) Sierra/SM heavy test examining the dynamic impact of two cylinders. Dash contact using both the face/face and node/face formulations is tested. Two types of 8-noded, hexahedral elements are examined, namely (1) uniform gradient (mean quadrature) elements, and (2) fully-integrated elements both with a strongly objective strain incrementation. The first element is the most commonly used element and the second one (loosely speaking) provides a bound on the element formulations (in terms of integration).

2.4.1.1. Functionality Tested

Primary capabilities:

- Dash contact face-face and node-face formulations

Secondary capabilities:

- The following element formulations:

  (1) eight-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation.

  (2) eight-node hexahedron with the mean quadrature formulation and strongly objective strain incrementation.

- prescribed displacement boundary conditions

2.4.1.2. Mechanics of Test

The in-plane geometry of the cylinder-cylinder contact problem is depicted in Figure 2-8. SI units are adopted for this problem, and thus the radius of the cylinders is 4 meters. The half-cylinders, as shown in the figure, have equal radii, but also have equal thicknesses (lengths). The thicknesses are defined for each mesh such that the elements in the contact region are approximately cubes. The problem is defined as a quasistatic problem under displacement controlled deformation.

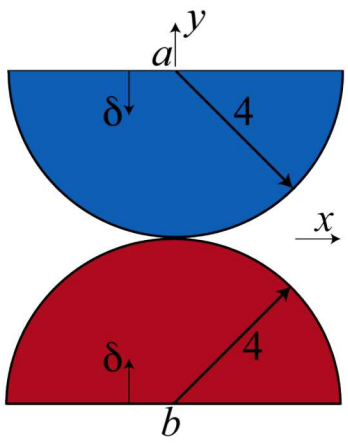


Figure 2-8. Hertz cylinder-cylinder contact problem.

2.4.1.3. Material Model

The primary material model used for this problem is the elastic model implemented in Lame [1].
The selected properties were given as follows.

Young's Modulus	E	1.0×10^5 Pa
Poisson's Ratio	ν	0.2

To examine the effect upon the convergence of the temporal integration of the elastic model (a hypoelastic model), limited analyses using a hyperelastic model were conducted as well, but these results are "document static", i.e., are not updated automatically.

2.4.1.4. Boundary Conditions

The boundary conditions for this problem, as depicted in Figure 2-8, show the horizontal surfaces (symmetry cuts) of the two half-cylinders have prescribed vertical displacements, denoted as δ. The maximum value of δ, which is the state at which the response is measured, is 2 cm. The half-cylinders geometrically thus look more like half-disks, but plane strain boundary conditions are applied to both "z-faces." The horizontal symmetry cuts of the cylinders allow us to define these surfaces as displacement reference planes; physically this corresponds to a unit cell out of a

stack of cylinders. If the objective were to reduce the problem size, it could be reduced further (in this case) to a cylinder-plane contact problem.

2.4.1.5. Meshes

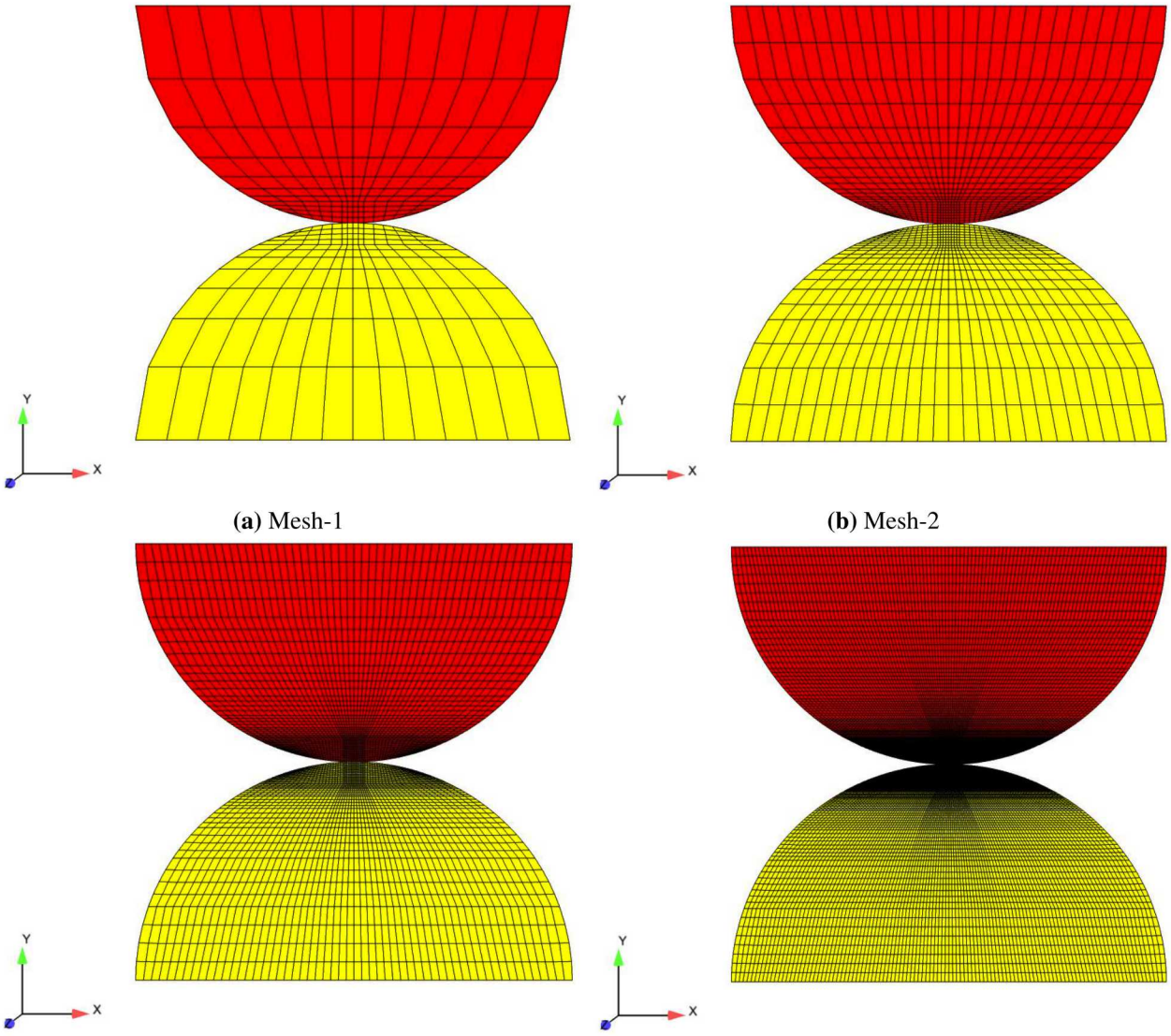
Four of the five meshes used in this study are shown in Figure 2-9. Each mesh contains four times as many elements (in the plane) as the coarser mesh that it is refined from, since $h_i = h_{i-1}/2$, where h_i denotes the characteristic in-plane element size for mesh i. The mesh refinements conform to the defining geometry, not the coarser mesh, and as such the solution space for the coarser mesh is not a proper subspace of the solution space for the finer mesh. Through the thickness, we examined mesh refinements that (1) maintained a constant thickness with one element through the thickness and mesh refinements that (2) varied the thickness with one element through the thickness. Approach (1) varies the element aspect ratios (at a given point in space) with mesh refinement, while approach (2) approximately maintained the aspect ratios (at a give point in space) by varying the element thickness to give approximately cube elements in the contact region. The apparent rates of convergence differed for the different mesh cases, but the value of "error" they converged to was essentially the same. Since approach (1) varies the mesh quality with mesh refinement it is not used here to examine the rates of convergence.

The table below contains the number of elements for each of the meshes.

Mesh name	Number of Elements
Mesh-1	308
Mesh-2	1232
Mesh-3	4928
Mesh-4	19712
Mesh-5	78848

2.4.2. Expected Results

For this problem we have evaluated the results in two ways: (1) using an analytical reference solution based upon the Hertz approach, and (2) using asymptotic estimates of the rate of convergence based upon results from sequences of three meshes. The analytical reference solution is briefly discussed below. The asymptotic analysis leading to the rate of convergence based upon a sequence of approximate results (like the development of Richardson's extrapolation) is based upon the assumption that the form of the dominant error term for each mesh is as ch_i^p. Once the observed rate of convergence is obtained, it can be used in the generalized Richardson extrapolation to give a higher order estimate of the exact solution. The motivation for using the analysis to determine the observed rate of convergence first is two fold: (1) it provides an indicator that the approximate solutions are in the asymptotic range, and (2) for quantities of interest like the reaction force we are treating the rate of convergence as an unknown since in general we do not expect the contact algorithm to maintain the optimal rates of convergence that are observed for simpler continuum problems. A detailed description of the analysis that provides



(a) Mesh-1 **(b)** Mesh-2

(c) Mesh-3 **(d)** Mesh-4

Figure 2-9. Four of the five meshes used in this study

the estimated rate of convergence is presented in the text by Oberkampf and Roy [2]. Roach [3] indicates that the analysis leading to the rate of convergence is from G. de Vahl Davis [4].

The analytical reference solution used in this study is taken from the *ContactMechanics* text of K.L. Johnson [5]. The relation for the displacement on the flat surface of the cylinder for this problem can be obtained as

$$\delta = \frac{(1-\nu^2)P\left(-1+2\ln\left(\frac{2\sqrt{\pi}R}{\sqrt{\frac{(1-\nu^2)PR}{E}}}\right)\right)}{E\pi} \tag{2.7}$$

where $R \sim$ radius of the cylinder, $E \sim$ Young's modulus, $\nu \sim$ Poisson's ratio, and $P \sim$ contact force. Note that the form of the equation does not lend itself to the algebraic solution for P. As

such, we apply the equation in its current form in the following manner: (1) the FE model applies displacements of magnitude δ to the two horizontal cuts of the cylinders, (2) the reaction force (equivalent in magnitude to the contact force, P) is calculated in the FE analysis, and (3) this value is used in the analytical expression above to determine the theoretical value for δ that should have caused this level of force. The difference between the values of δ applied to the model, and that obtained from the analytical expression are the quantity of interest type "error measure" used in this study.

For readers that have additional interest in the source of the above equation, the results from symbolic calculations within Mathematica are included in this test file's directory for reference. In particular, the above form reflects the particular data used to specify this problem: identical cylinders with respect to both geometry and material. Note that the Mathematica results also present the graphical relationships δ *vs.* P and a *vs.* P, where $a \sim$ contact width.

The analytical solution for this problem is not exact not only because it is based upon linear elasticity, but also because it is based upon the simplifying approximations presented by Hertz. These approximations include: (1) a representation of the contact surfaces by quadratic surfaces, (2) a component of the deformation response of each body can be approximated by the solution of a loaded half-space, and (3) relative displacement between the center and edge points of contact are small compared to the contact radius. These approximations require both the geometric dimensions of the body and the radii of curvature in the contact region (one in the same for this problem) to be much larger than the contact radius. Thus the ideal, in terms of using these approximations, is to adopt an extremely small contact area, but then that makes it more difficult to define a mesh that efficiently uses small elements near the contact but transitions to larger elements away from this region (for the sake of numerical efficiency). In defining this problem, we initially sought to find a balance between test run times and sufficient accuracy to obtain a measure of convergence, but admittedly pushed the upper limit of the contact size. Figure 2-10 depicts how localized the contact response is even with the selected contact area.

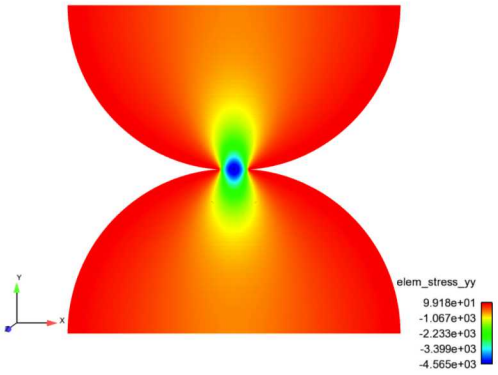


Figure 2-10. Concentrated stress response for the cylinder-cylinder contact problem.

Since the reference solution is not exact the difference in the solutions is not really the error, though it may be close to the actual error for coarser meshes. The "error" value that the solution levels off to (in the limit) is a measure of the error in the reference solution, assuming that the finite element solution is actually converging to the exact solution. The convergence to a fixed

difference between the analytical reference solution and the finite element solutions, can occur from above or below and is not necessarily monotonic in nature. Because of this convergence behavior for finer meshes, it can be difficult to find a range of discretization for which the approximate reference solution is sufficiently accurate to serve as a "surrogate" for the exact solution and yet the meshes are sufficiently fine to be in the asymptotic range. As we will see in this case, we did not obtain a region where the inexact reference solution allowed us to estimate the rate of convergence, but we will observe it converging to a fixed difference. To strengthen the argument that it is converging and to address the question of rate, we will estimate the rate of convergence using the approach discussed above and apply Richardson Extrapolation to estimate the exact solution.

2.4.3. Verification Results

As noted above, the quantity of interest in this test (for the analytical reference solution) is the boundary displacement, δ. The slopes of the relative error curves between the data points (corresponding to two meshes, on the log-log plots) yield observed rates of convergence. For an exact reference solution, the observed rate of convergence approaches the asymptotic rate with mesh refinement, assuming other sources of numerical error (e.g., solver accuracy) do not corrupt the results. For this problem we are not using an exact solution, so an improvement in the convergence estimate is not guaranteed. As previously noted, typically for problems without an exact solution there is (or we hope for) a "sweet range" where the approximations are in the asymptotic range but not refined enough to measure the inexactness of the references solution. Of course the size of this "sweet range" is problem dependent, e.g., in this problem we have not only the approximations associated with linear elasticity but also those associated with the Hertz solution.

Initially we will examine the observed rates of convergence based upon the approximate reference solution.

2.4.3.1. Results based on Hertz reference solution

The following tables give the observed rates of convergence for the two variations of the Dash contact algorithm and the two Hex8 element formulations between each sequential pair of meshes, where h_{fine} denotes the relative element size of the finer mesh of the pair (i.e., where 1 denotes the coarsest mesh - mesh 1). The following plots show the corresponding graphical representations of the error data as a function of the element size.

Table 2-4. Observed convergence rates based upon the Hertz reference solution.

Face/face

Mean quadrature

h_{fine}	$\lvert\delta_{error}\rvert/\lvert\delta_{analyt}\rvert$
0.5000	2.6467
0.2500	0.8739
0.1250	-0.8244
0.0625	-0.1643

Fully integrated

h_{fine}	$\lvert\delta_{error}\rvert/\lvert\delta_{analyt}\rvert$
0.5000	3.0267
0.2500	-0.0444
0.1250	-0.6605
0.0625	-0.1594

Node/face

Mean quadrature

h_{fine}	$\lvert\delta_{error}\rvert/\lvert\delta_{analyt}\rvert$
0.5000	2.6140
0.2500	0.8709
0.1250	-0.8363
0.0625	-0.1671

Fully integrated

h_{fine}	$\lvert\delta_{error}\rvert/\lvert\delta_{analyt}\rvert$
0.5000	3.0102
0.2500	-0.2828
0.1250	-0.6494
0.0625	-0.1605

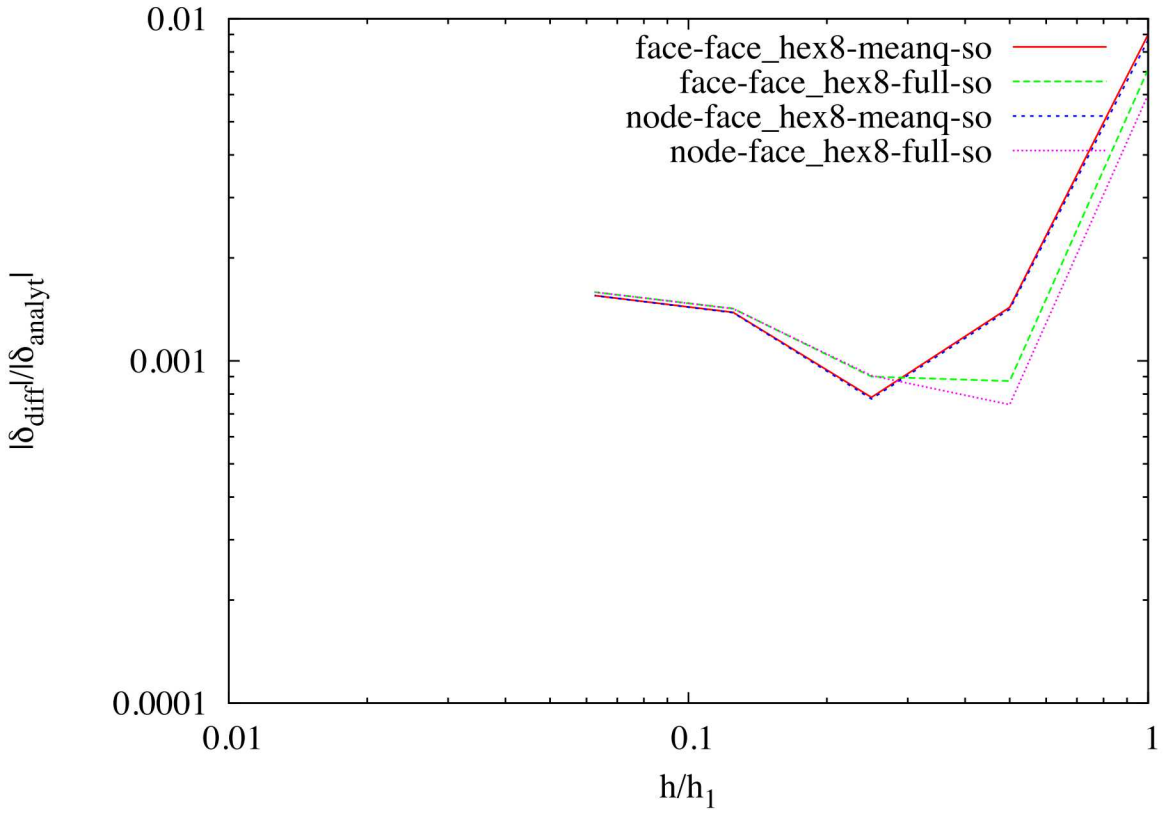


Figure 2-11. Convergence of the displacement boundary condition versus element size.

While the results suggest that each test case is converging (just not to the analytical reference solution), as previously noted, the inexactness of the reference solution makes an estimate of the rate of convergence intractable for the selected models. To examine the convergence rate we resort to asymptotic analyses of the numerical results alone (i.e., without assuming a reference solution) in the following section.

2.4.3.2. Results based on asymptotic analysis

The asymptotic analysis applied in this verification problem can be considered as consisting of two steps. First, the results from sequences of analyses based upon three mesh refinements, where each refinement halves the characteristic length of the element (i.e., each hex is approximately subdivided, into eight hex elements), are used to estimate the rate of convergence. (Note that only four of the elements are present in the finer mesh since the thickness is halved in the refinement.) Second, the convergence rate obtained from the finest sequence of meshes is assumed to be accurate, and then is used with Richardson extrapolation to obtain a higher order estimate of the exact solution. The Richardson extrapolated estimate is then adopted as the reference solution to analyze the results, as the analytical reference solution was used in the previous section.

Using sequences of three numerical results one can solve for the observed rate of convergence. Two values are presented in the table, one for the normal force (P), and one for the contact radius (a) calculated from P . Calculating the contact radius from P, in a sense just makes it a measure of P, and both quantities yield nearly the same rates of convergence. The rates of convergence are nearly quadratic for the reaction force with the mean quadrature element formulation. Also note that the relative consistency of the convergence rates (more so for the mean quadrature results with the finer two sequences of three meshes) suggests the results are in the asymptotic range.

Table 2-5. Observed convergence rates based upon asymptotic analysis.

Face/face

Mean quadrature

h_{fine}	P	a
0.2500	1.80	1.79
0.1250	1.89	1.89
0.0625	1.85	1.84

Fully integrated

h_{fine}	P	a
0.2500	1.84	1.83
0.1250	1.77	1.77
0.0625	1.65	1.63

Node/face

Mean quadrature

h_{fine}	P	a
0.2500	1.76	1.75
0.1250	1.86	1.86
0.0625	1.84	1.82

Fully integrated

h_{fine}	P	a
0.2500	1.69	1.69
0.1250	1.69	1.68
0.0625	1.62	1.62

Since we use a sequence of three numerical results in the asymptotic analysis (giving us three

equations), we can solve for the two remaining unknowns: the constant (c) and the estimate of the exact solution (which is one order more accurate than that given by the finite element solution, assuming the next term in the error expansion is one order higher); this part of the asymptotic analysis corresponds to Richardson extrapolation. We then use the higher order estimate of the exact solution (labeled by *RE*) as our reference solution. Admittedly, this higher order solution estimate is better suited for uncertainty quantification [2,3], but we will still use it here as a reference solution to show that it yields the desired linear relationship between error and discretization on a log-log plot (for P). Following the same order as we did above for the analytical solution, first consider the convergence rates obtained using P_{RE} and a_{RE} as the reference solutions, in tabular form. These results are obtained from pairs of meshes, and by definition approach the same values obtained from the asymptotic analyses with mesh refinement.

Table 2-6. Observed convergence rates based upon the Richardson extrapolation references, P_{RE} and a_{RE}.

Face/face

	Mean quadrature			Fully integrated	
h_{fine}	P	a	h_{fine}	P	a
0.5000	1.8204	1.8133	0.5000	1.8091	1.8043
0.2500	1.8773	1.8775	0.2500	1.7327	1.7306
0.1250	1.8544	1.8439	0.1250	1.6507	1.6335
0.0625	1.8544	1.8439	0.0625	1.6507	1.6335

Node/face

	Mean quadrature			Fully integrated	
h_{fine}	P	a	h_{fine}	P	a
0.5000	1.7847	1.7771	0.5000	1.6847	1.6812
0.2500	1.8532	1.8505	0.2500	1.6680	1.6625
0.1250	1.8446	1.8176	0.1250	1.6246	1.6215
0.0625	1.8446	1.8176	0.0625	1.6246	1.6215

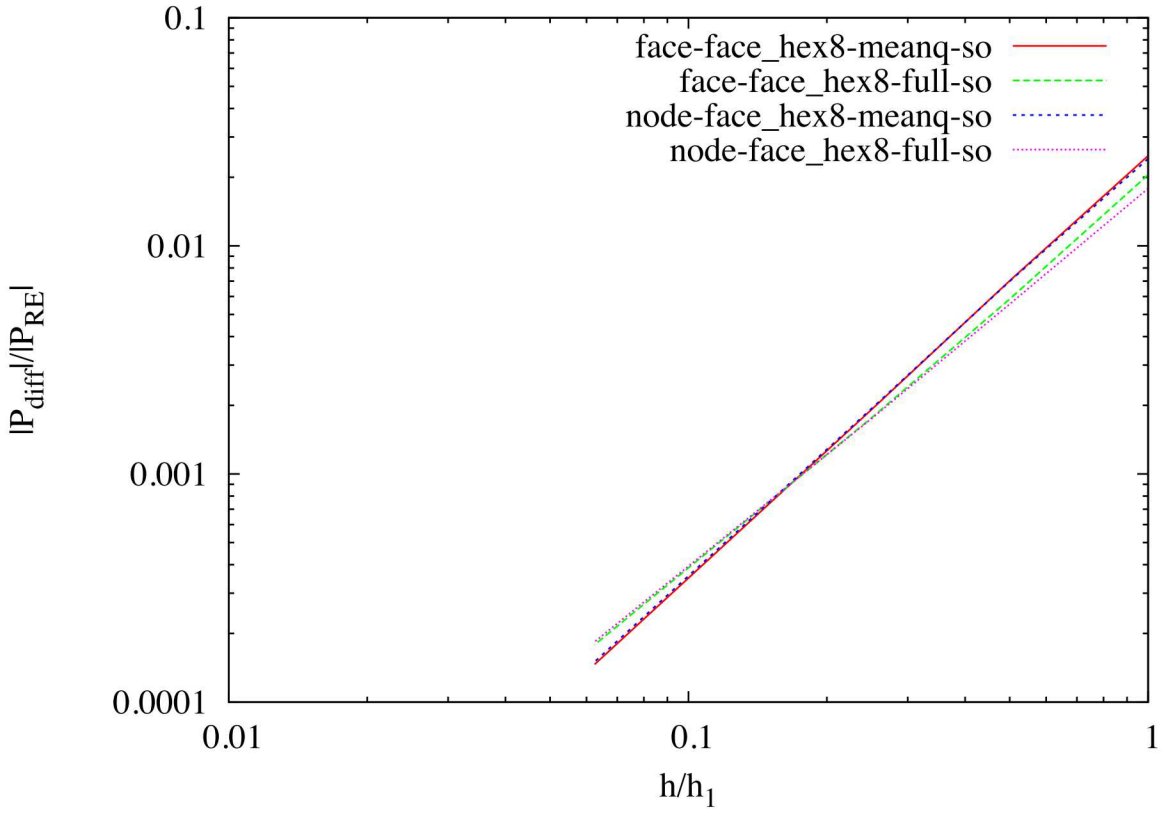


Figure 2-12. Convergence of the normal force, P, versus element size.

The above results suggest that reasonable accuracy is obtained (less than 1 percent difference) for the contact force, except for the coarsest meshes. The asymptotic results also enforce the interpretation that the convergence to a constant difference when using the analytical reference solution was an indication of the weaknesses in the analytical solution not the contact algorithm.

While the above results indicate that the algorithm is giving nearly quadratic convergence in the response, it does beg the questions of whether these results are as close as the algorithm can come to producing quadratic convergence, whether there is an error in the algorithm producing a reduced rate of convergence, or whether other aspects of the numerical simulation are polluting the observed rates of convergence. Frankly, we do not expect the algorithm to maintain the optimal rate of convergence associated with the elements, but it is still worth considering the other factors that can reduce the observed rate of convergence; among the other factors are relaxed solver tolerances that reduce the accuracy of the solution, and a mixture of the order of the algorithms that has not been accounted for in the convergence study. The solver tolerances were adjusted to be as tight as possible while still yielding a converged solution. The second issue however was purposefully not completely addressed in the above results to keep the analysis times smaller; specifically, the elastic material model is a hypoelastic model and thus is numerically integrated in time. At best we would expect quadratic convergence in time, and thus for the asymptotic terms associated with both space and time to be consistently reduced (assuming quadratic convergence in time) we should have reduced the time step by a factor of one half with each mesh refinement. We assumed this effect would be relatively small – though not necessarily negligible, but used the elastic model because it is the underlying elastic model for several commonly used models in Lame [1].

To indirectly examine the effect that the elastic model may have had on the accuracy, let's consider some results obtained with the neo-Hookean model (a hyperelastic model which thus does not require temporal integration). The table below presents the convergence results for the two tests based upon node/face contact.

Table 2-7. Observed convergence rates based upon asymptotic analysis.

Cases: neo-Hookean material model, and node/face contact.

Mean quadrature			Fully integrated		
h_{fine}	P	a	h_{fine}	P	a
0.2500	1.13	1.12	0.2500	1.60	1.60
0.1250	1.88	1.87	0.1250	1.71	1.70
0.0625	1.90	1.91	0.0625	1.64	1.66

For the finest mesh sequence and the mean quadrature element, the convergence rate for P increased from 1.84 to 1.90. This change is not negligible and would be important if we expected to obtain quadratic convergence in the limit. The improvement for the fully integrated element is less significant.

Summary of results: the contact algorithm appears to converge for this classical contact problem, and the difference between the Hertz reference solution and the FEM solutions for the finer meshes is less than one percent. The difference results (referencing the Hertz solution) do not lend themselves to directly evaluating the rate of convergence of the contact algorithm, as there are not sufficient data that exhibit asymptotic behavior without being tainted by the inaccuracy of the reference solution. Using the Hertz solution the numerical results approach a constant difference which we interpret in the limit as representing the error in the analytical solution. To enforce this interpretation, we estimated the rate of convergence for the reaction force using asymptotic analysis which "approached quadratic convergence." We interpret these results as positive verification results; however, these results must be weighted with the facts that the analytical reference solution is not exact and the use of asymptotic analysis does not provide as strong of verification as having an exact reference solution [2,3,6].

2.4.4. References

1. Scherzinger, W.M. and Hammerand, D.C. *Constitutive Models in Lame*. Sandia Report SAND2007-5873, September 2007.
2. Oberkampf. W.L. and Roy, C.J. *Verification and Validation in Scientific Computing*. Cambridge University press, 2010.
3. Roach, P.J. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.
4. de Vahl Davis, G. "Natural Convection of Air in a Square Cavity: A Bench Mark Numerical Solution," *International Journal for Numerical Methods in Fluids*," vol 3, pp 249-264.
5. Johnson K.L. *Contact Mechanics*. Cambridge University press, 1985.
6. Cox, J.V. and Mish, K.D. *Sierra Solid Mechanics Example Verification Problems to Highlight the use of Sierra Verification Tools*, Sandia Report SAND2013-2390, 2013.

For input deck see Appendix B.4.

2.5. MINDLIN CYLINDER-CYLINDER CONTACT – CONVERGENCE TEST

Analysis Type	Quasi-statics
Element Types	Hex8
Element Formulations	Mean Quadrature, Fully Integrated
Strain Incrementation	Strongly Objective
Material Models	Elastic
Verification Category	Convergence
Verification Quantities	Boundary Displacement (δ), Contact Shear (Q)
Number of Tests	4
Keywords	Mindlin, Hertz, Contact, Friction, Convergence

2.5.1. Brief Description

This series of analyses demonstrates the convergence of contact for the classical Mindlin problem [6]. This problem builds on the Hertz problem (cylinder on cylinder) to develop the normal preload, and then follows that with a lateral shear applied to the flat surfaces of both half-cylinders. Dash contact using both the face/face and node/face formulations is tested. Two types of 8-noded, hexahedral elements are examined, namely (1) uniform gradient (mean quadrature) elements, and (2) fully-integrated elements both with a strongly objective strain incrementation. The first element is the most commonly used element and the second one (loosely speaking) provides a bound on the element formulations (in terms of integration).

2.5.1.1. Functionality Tested

Primary capabilities:

- Dash contact face-face and node-face formulations

Secondary capabilities:

- The following element formulations:

  (1) eight-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation.

  (2) eight-node hexahedron with the mean quadrature formulation and strongly objective strain incrementation.

- Prescribed displacement boundary conditions

2.5.1.2. Mechanics of Test

The geometry consists of two half-cylinders in contact, as depicted in Figure 2-13. SI units are adopted for this problem, and thus the radius of the cylinder is 4 meters. The half-cylinders, as shown in the figure, have equal radii. The problem is defined as a quasistatic problem under displacement controlled deformation. Note that because we have posed this in terms of displacement boundary conditions, the normal and shear forces will change with mesh refinement, as the cylinders change in compliance. The problem consists of two loading periods. The first period corresponds to the Hertz problem with normal displacements applied to the cylinder-halves (flat surfaces) to establish a normal force. The second period applies lateral displacements to the flat surfaces of the cylinder halves, developing shear loads on both the reaction faces and the contact surfaces.

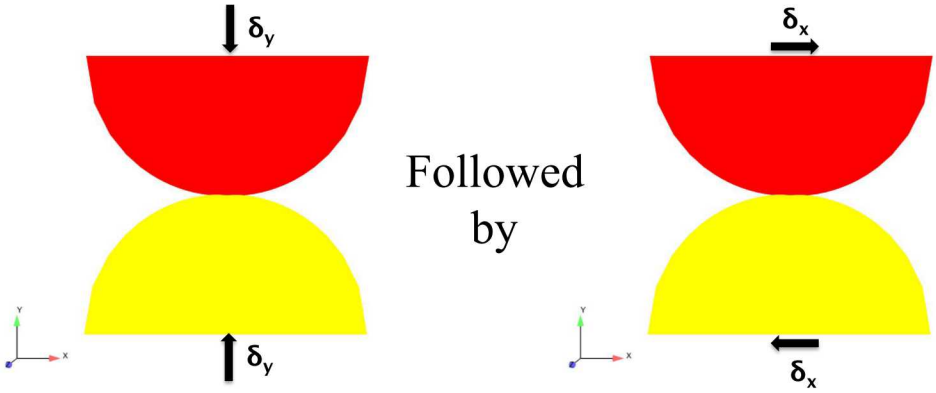


Figure 2-13. Mindlin cylinder-cylinder contact problem.

2.5.1.3. Material Model

The primary material model used for this problem is the elastic model implemented in Lame [2].
The selected properties were given as follows.

Table 2-8. Material model properties.

Young's Modulus	E	1.0×10^5 Pa
Poisson's Ratio	ν	0.2

2.5.1.4. Boundary Conditions

The boundary conditions for this problem, as depicted in Figure 2-13, show the horizontal surfaces (symmetry cuts) of the two half-cylinders have prescribed vertical displacements, denoted as δ_y followed by prescribed lateral displacements, denoted as δ_x. The half-cylinders geometrically thus look more like half-disks, but plane strain boundary conditions are applied to

both z-faces. In the second time period, the vertical displacements on the boundaries are held constant, and the horizontal displacements are varied linearly in time. The prescribed displacements end at maximum magnitudes of 1 cm.

2.5.1.5. Meshes

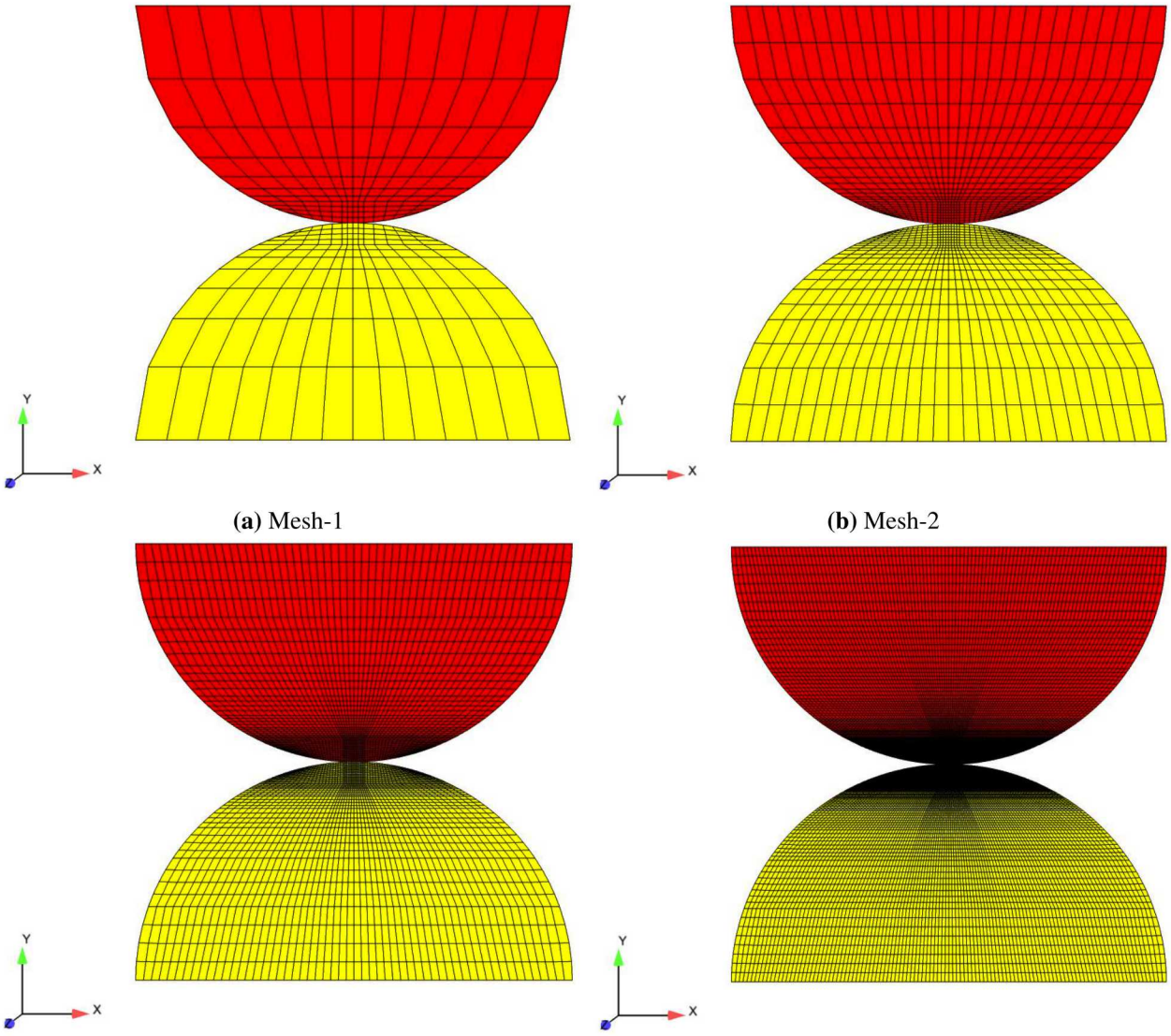
Four of the five meshes used in this study are shown in Figure 2-14. Each mesh contains four times as many elements (in the plane) as the coarser mesh that it is refined from, since $h_i = h_{i-1}/2$, where h_i denotes the characteristic in-plane element size for mesh i. The mesh refinements conform to the defining geometry, not the coarser mesh, and as such the solution space for the coarser mesh is not a proper subspace of the solution space for the finer mesh. We varied the thickness, with one element through the thickness, to approximately maintain the element aspect ratios (at a give point in space); this approximately gave cube elements in the contact region.

Table 2-9. Mesh characteristics.

Mesh label	h/h_1	Number of Elements
Mesh-1	1	308
Mesh-2	1/2	1232
Mesh-3	1/4	4928
Mesh-4	1/8	19712
Mesh-5	1/16	78848

2.5.2. Expected Results

For this problem we have evaluated the results using an analytical estimation of the rate of convergence based upon results from sequences of three meshes. The analysis leading to the rate of convergence based upon a sequence of approximate results (like the development of Richardson's Extrapolation) is based upon the assumption that the form of the error for each mesh is as ch_i^p. The three equations for the shear loads in terms of the higher order estimate of the exact solution and the error term are solved for the rate of convergence (eliminating the unknowns c and the higher order estimate of the exact solution). Once the observed rate of convergence is obtained, it can be used in the generalized Richardson extrapolation to obtain a higher order estimate of the exact solution. The motivation for using the analysis to determine the observed rate of convergence first is two fold: (1) it provides an indicator that the approximate solutions are in the asymptotic range, and (2) for quantities of interest like the reaction force we are treating the rate of convergence as an unknown since in general we do not expect the contact algorithm to maintain the optimal rates of convergence that are observed for simpler continuum problems. For the results to indicate that the sequences of results are in the asymptotic range, we expect the predicted rates of convergence from sequential sets of three meshes (e.g., meshes 2,3,4 and meshes 3,4,5) to give nearly the same rates of convergence. A detailed description of the analysis that provides the estimated rate of convergence is presented in the text by Oberkampf and Roy



(a) Mesh-1 **(b)** Mesh-2

(c) Mesh-3 **(d)** Mesh-4

Figure 2-14. Four of the five meshes used in this study

[3]. Roach [4] indicates that the analysis leading to the rate of convergence is from G. de Vahl Davis [5].

2.5.3. Verification Results

As noted above, the quantity of interest in this test (for the analytical reference solution) is the shear load, Q. The slopes of the relative error curves between the data points (corresponding to two meshes, on the log-log plots) yield observed rates of convergence; for the convergence plots shown here, these slopes correspond to the asymptotic rate of convergence for the two finest sets of meshes. This result is by definition since the three meshes were used to solve for the rate. The slopes corresponding to coarser meshes will match those of the finer meshes if they are in the

asymptotic range.

Because the analyses associated with the finest mesh can be very time consuming, a different approach is being taken in presenting the results. The nightly analyses only use the finest three meshes, and as formulated here will obtain a different solution for the Richardson extrapolation than results based upon finer meshes (extended results). As such nightly and extended results are presented separately.

2.5.3.1. Results based on asymptotic analysis

The asymptotic analysis applied in this verification problem can be considered as consisting of two steps: one to obtain the rate of convergence, and one to obtain a higher order estimate of the solution from Richardson's Extrapolation. This is discussed in more detail in the introduction of this manual. The tables below present the results from the first step: the estimated convergence rates from sequences of three meshes.

The shear load (Q) is again treated as the quantity of interest from which that rate of convergence is estimated. Note that for this problem we do not have sufficient consistency between the results (for successive sequences of three meshes) to definitely claim that we are in the asymptotic range of convergence.

Table 2-10. Observed convergence rates based upon asymptotic analysis – Extended results.

Face/face

Mean quadrature

h_{fine}	P
0.2500	1.74
0.1250	1.67
0.0625	2.94

Fully integrated

h_{fine}	P
0.2500	1.80
0.1250	1.61
0.0625	1.83

Node/face

Mean quadrature

h_{fine}	P
0.2500	1.12
0.1250	1.61
0.0625	1.84

Fully integrated

h_{fine}	P
0.2500	1.16
0.1250	1.43
0.0625	1.14

Table 2-11. Observed convergence rates based upon asymptotic analysis – Nightly results.

Face/face

Mean quadrature

h_{fine}	P
0.2500	1.73

Fully integrated

h_{fine}	P
0.2500	1.79

Node/face

Mean quadrature

h_{fine}	P
0.2500	1.13

Fully integrated

h_{fine}	P
0.2500	1.16

Given the higher order estimate of the exact solution from Richardson's extrapolation we now use this result (labeled by *RE*) as our reference solution. Admittedly, this higher order solution estimate is better suited for uncertainty quantification [3,4], but we will still use it here as a reference solution to show that by design it yields the desired linear relationship between error and discretization on a log-log plot. First we present the convergence rates obtained using Q_{RE} as the reference solution, in tabular form. These results are obtained from pairs of meshes, and by definition approach the same values obtained from the asymptotic analyses with mesh refinement. As previously presented, extended results are followed by nightly results. For this set of results, the extended and nightly results do not match because the extrapolated reference solution is not based upon the same sets of meshes.

Table 2-12. Observed convergence rates based upon the Richardson extrapolation references, Q_{RE} – Extended results.

Face/face

Mean quadrature

h_{fine}	P
0.5000	1.79
0.2500	1.91
0.1250	2.94
0.0625	2.94

Fully integrated

h_{fine}	P
0.5000	1.76
0.2500	1.68
0.1250	1.83
0.0625	1.83

Node/face

Mean quadrature

h_{fine}	P
0.5000	1.32
0.2500	1.68
0.1250	1.84
0.0625	1.84

Fully integrated

h_{fine}	P
0.5000	1.22
0.2500	1.31
0.1250	1.14
0.0625	1.14

Table 2-13. Observed convergence rates based upon the Richardson extrapolation references, Q_{RE} – Nightly results.

Face/face

Mean quadrature

h_{fine}	P
0.5000	1.7329
0.2500	1.7329

Fully integrated

h_{fine}	P
0.5000	1.7929
0.2500	1.7929

Node/face

Mean quadrature

h_{fine}	P
0.5000	1.1328
0.2500	1.1328

Fully integrated

h_{fine}	P
0.5000	1.1568
0.2500	1.1568

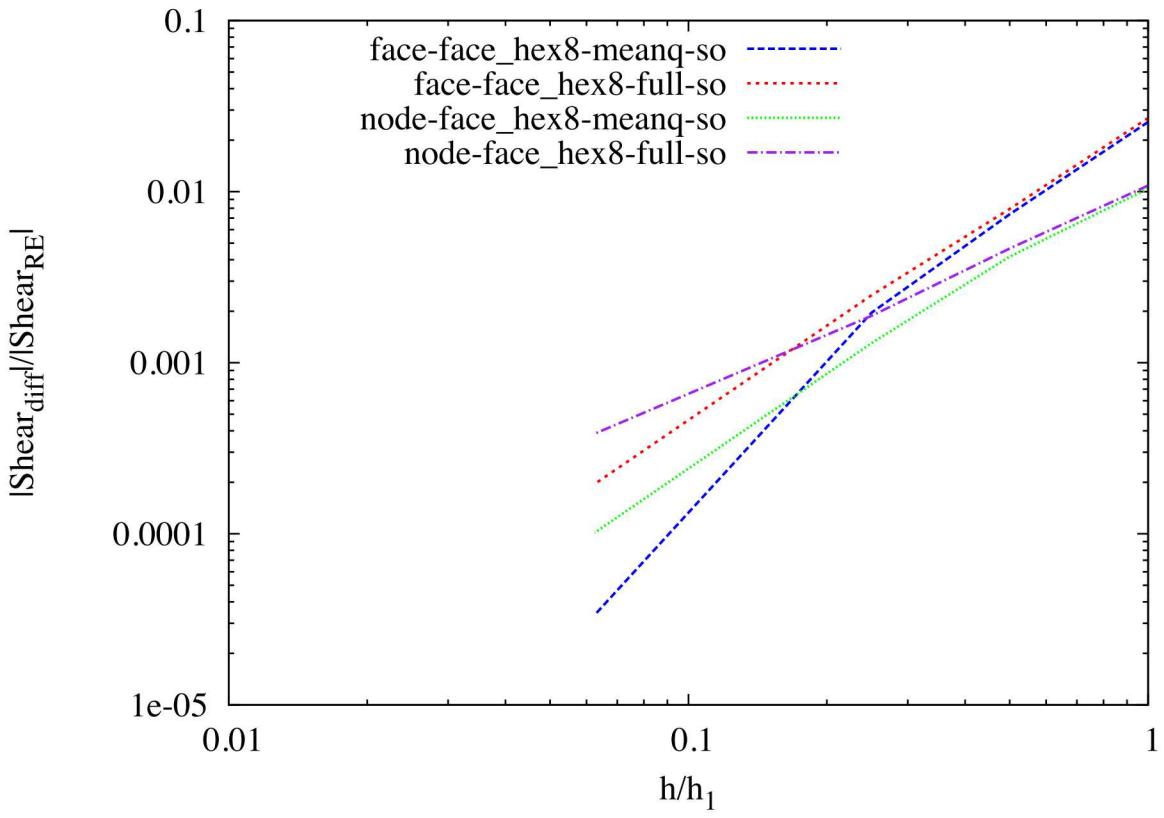


Figure 2-15. Convergence of the shear force, Q, versus element size, Richardson extrapolation reference solution using extended results.

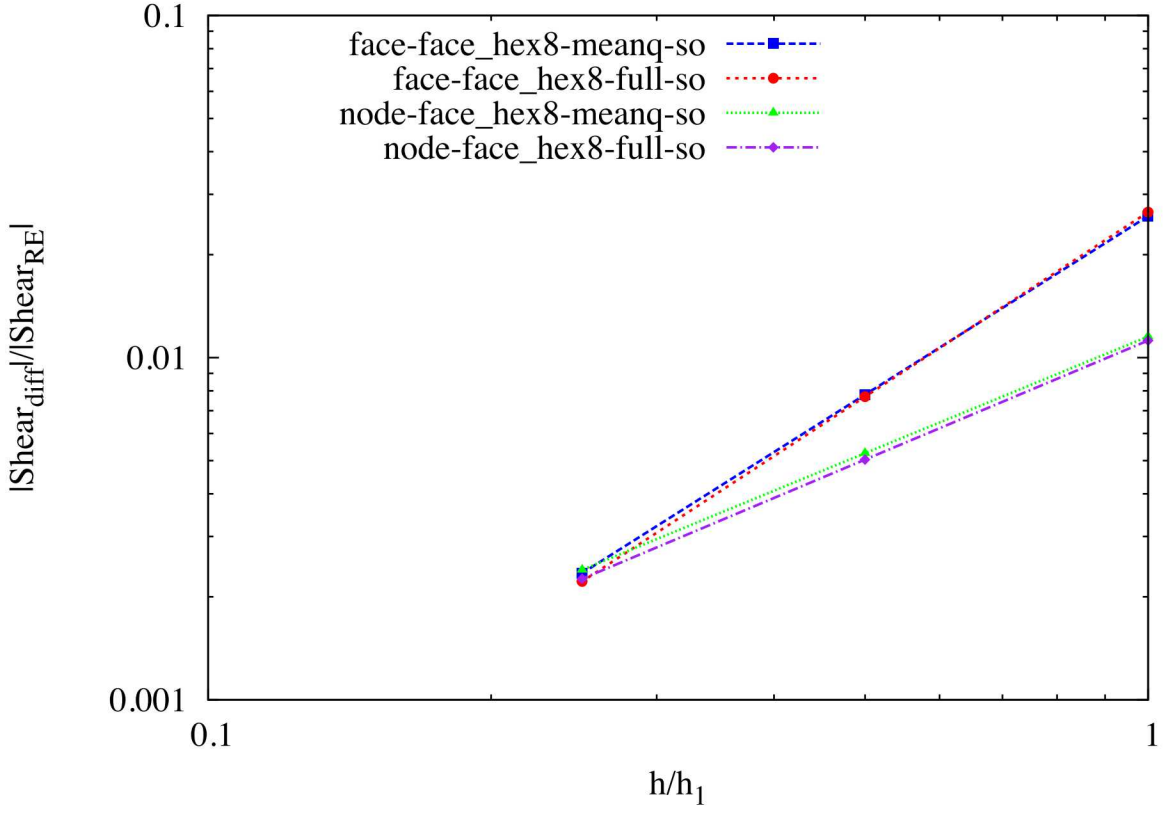


Figure 2-16. Convergence of the shear force, Q, versus element size, Richardson extrapolation reference solution using nightly results.

The inconsistency in the convergence rates makes it difficult to definitively assess the rate of convergence, but two test cases ($face/face$ + *fully integrated* and *node*/$face$ + *mean quadrature*) gave closer agreement and indicated convergence closer to quadratic than linear. The other two test cases showed greater variation in the rates of convergence and for the finest meshes gave rates of convergence that were closer to linear and cubic.
The plot of the asymptotic results graphically depicts the differences in the rates of convergence. Each test has its own extrapolated reference solution, so using the asymptotic results to compare the relative accuracies can be misleading without examining the agreement of the extrapolated reference solutions. If we take the results at face value the $face/face$ + *meanquadrature* test yields both the highest rate of convergence and accuracy. Furthermore all the test cases yield better than one percent difference for all but the coarsest mesh.

All of the asymptotic results indicate convergent behaviors and some give nearly quadratic convergence, but the results beg the questions of whether these results are as close as the algorithm can come to producing quadratic convergence, whether there is an error in the algorithm producing a reduced rate of convergence, or whether other aspects of the numerical simulation are polluting the observed rates of convergence. Frankly, we do not expect the algorithm to maintain the optimal rate of convergence associated with the elements, but it is still worth considering the other factors that can reduce the observed rate of convergence; among the other factors are relaxed solver tolerances that reduce the accuracy of the solution, and a mixture of the order of the algorithms that has not not been accounted for in the convergence study. The solver tolerances were adjusted to be as tight as possible while still yielding a converged solution.

The second issue however was purposefully not addressed in the above results to keep the analysis times smaller; specifically, the elastic material model is a hypoelastic model and thus is numerically integrated in time. At best we would expect quadratic convergence in time, and thus for the asymptotic terms associated with both space and time to be consistently reduced (assuming quadratic convergence in time and space) we should have reduced the time step by a factor of one half with each mesh refinement. We assumed this effect would be relatively small – though not necessarily negligible, but used the *elastic* model because it is the underlying elastic model for several commonly used models in Lame [2]. To examine the effect of using a model that does not require temporal integration, in the Hertz cylinder-cylinder contact test we examined how the results differed when using a hyperelastic model; in summary, the effect was second order relative to our deviations from second order. In order to investigate the effect of time step size, another series of tests (3 meshes) with smaller time steps (within the shear part) were ran. The results revealed a change in the convergence rates. Table 2-14 shows the percentage change in the convergence rates.

Table 2-14. Effect of time step size on convergence rate.

Case	Percentage change in convergence rates
Face-Face-Meanq	1.14
Face-Face-Full	1.11
Node-Face-Meanq	-5.36
Node-Face-Full	-6.90

Summary of results: the contact algorithm appears to converge for this classical contact problem, and the differences between the Richardson extrapolation solutions and the FEM solutions for for all but the coarsest mesh were less than one percent. The asymptotic estimates for the rates of convergence were not sufficiently consistent to definitively identify the actual convergence rates of the tests, but two of the results indicated convergence rates closer to quadratic than to linear convergence. We interpret these results as positive verification results; however, these results must be weighted with the fact that the use of asymptotic analysis does not provide as strong of verification as having an exact reference solution [3,4,7]; it merely indicates that the FEM solution for the shear load is converging to some value.

2.5.4. References

1. Mindlin, R. D. *Compliance of elastic bodies in contact*. Trans. ASME, Series E, Journal of Applied Mechanics, vol 16, pp 259-268, 1949.
2. Scherzinger, W.M. and Hammerand, D.C. *Constitutive Models in Lame*. Sandia Report SAND2007-5873, September 2007.
3. Oberkampf. W.L. and Roy, C.J. *Verification and Validation in Scientific Computing*. Cambridge University press, 2010.
4. Roach, P.J. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.
5. de Vahl Davis, G. "Natural Convection of Air in a Square Cavity: A Bench Mark Numerical Solution," *International Journal for Numerical Methods in Fluids*, vol 3, pp 249-264.
6. Johnson K.L. *Contact Mechanics*. Cambridge University press, 1985.
7. Cox, J.V. and Mish, K.D. *Sierra Solid Mechanics Example Verification Problems to Highlight the use of Sierra Verification Tools*, Sandia Report SAND2013-2390, 2013.

For input deck see Appendix B.5.

2.6. HERTZ SPHERE-SPHERE CONTACT – CONVERGENCE TEST

Analysis Type	Quasi Statics
Element Types	Hex8
Element Formulations	Mean Quadrature, Fully Integrated
Strain Incrementation	Strongly Objective
Material Models	Elastic
Verification Category	Convergence
Verification Quantities	Boundary Displacement (δ)
Number of Tests	4
Keywords	Hertz, Contact, Convergence

2.6.1. Brief Description

This series of analyses demonstrates the convergence of contact for a classical Hertz solution. This problem is a quasistatic version of the Sierra/SM heavy test examining the quasistatic compression of two hemispheres. Two types of 8-noded, hexahedral elements are examined, namely (1) fully-integrated elements, and (2) uniform gradient elements with a strongly objective strain formulation.

2.6.1.1. Functionality Tested

Primary capabilities:

- Dash contact face-face and node-face formulations

Secondary capabilities:

- The following element formulations:

  (1) eight-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation.

  (2) eight-node hexahedron with the mean quadrature formulation and strongly objective strain incrementation.

- Prescribed displacement boundary conditions

2.6.1.2. Mechanics of Test

The side-view geometry of the sphere-sphere contact problem is depicted in Figure 2-17. SI units are adopted for this problem, and thus the radii of the hemispheres are 4 meters. The problem is defined as a quasistatic problem under displacement controlled deformation.

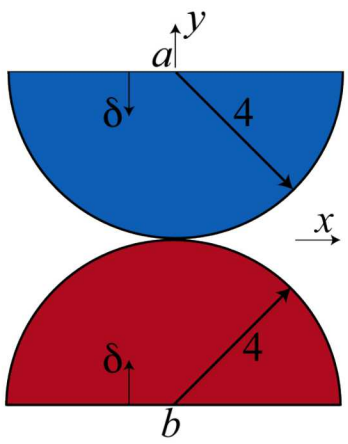


Figure 2-17. Hertz sphere-sphere contact problem.

2.6.1.3. Material Model

The material model used for this problem is the elastic model implemented in Lame [1]. The selected properties were given as follows.

Young's Modulus	E	1.0×10^5 Pa
Poisson's Ratio	ν	0.2

2.6.1.4. Boundary Conditions

The boundary conditions for this problem, as depicted in Figure 2-17, show the horizontal surfaces (symmetry cuts) of the hemispheres have prescribed vertical displacements, denoted as δ. The horizontal symmetry cuts of the hemispheres allow us to define these surfaces as displacement reference planes. If the objective were to reduce the problem size, it could be reduced further to a sphere-plane contact problem. Table 2-15 contains the mesh label, relative element size, and number of elements for each of the meshes.

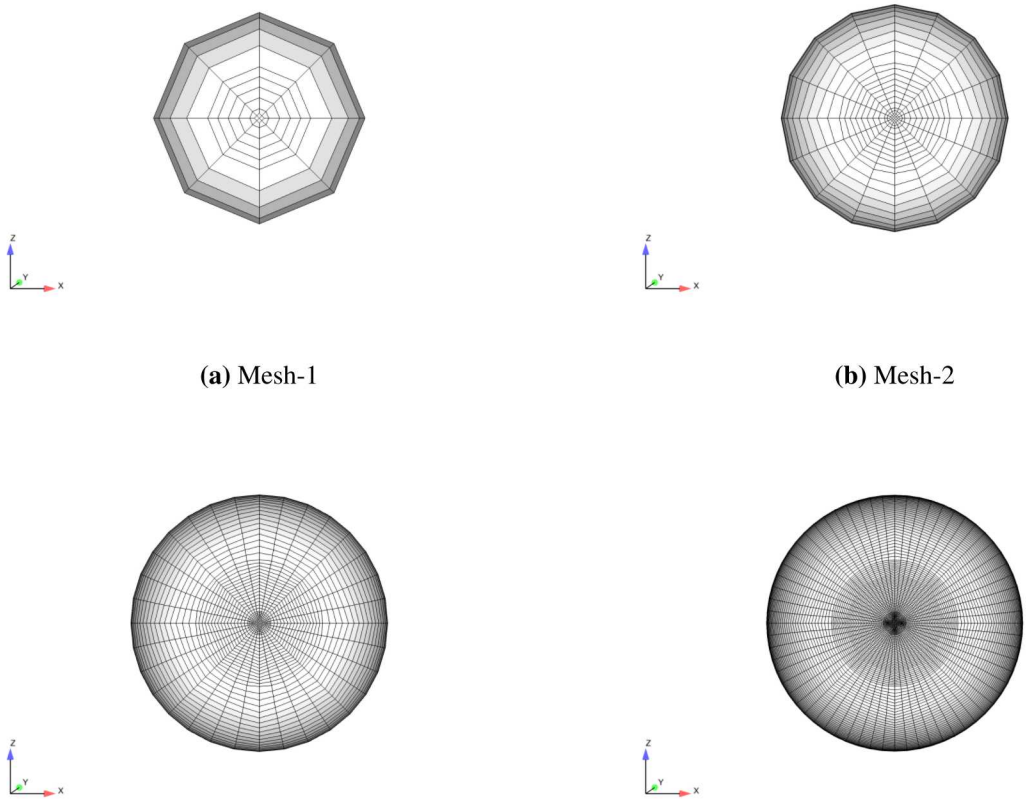
Table 2-15. Mesh characteristics.

Mesh label	h/h_1	Number of Elements
Mesh-1	1	392
Mesh-2	1/2	3136
Mesh-3	1/4	25088
Mesh-4	1/8	200704

2.6.1.5. Meshes

The meshes used in this study are shown in Figure 2-18. Each mesh contains eight times as many elements as the coarser mesh that it is refined from, since $h_i = h_{i-1}/2$, where h_i denotes the

characteristic element size for mesh i. The mesh refinements conform to the defining geometry, not the coarser mesh, and as such the solution space for the coarser mesh is not a proper subspace of the solution space for the finer mesh.



(a) Mesh-1

(b) Mesh-2

(c) Mesh-3

(d) Mesh-4

Figure 2-18. Meshes used in this study

2.6.2. Expected Results

The analytical reference solution used in this study is taken from the *Contact Mechanics* text of K.L. Johnson [5]. The relation for the mutual displacement of the hemispheres is represented by

$$\delta = \left(\frac{9(-1+\nu^2)^2 P^2}{2E^2 R} \right)^{1/3} \tag{2.8}$$

where R denotes the radii of the hemispheres, E denotes Young's modulus, ν denotes Poisson's ratio, and P denotes contact force. Consistent with the approach used for Hertz cylinder-cylinder

tests, we apply the equation in its current form in the following manner: (1) the FE model applies displacements of magnitude δ to the two horizontal cuts of the hemispheres, (2) the reaction force (equivalent in magnitude to the contact force, P) is calculated in the FE analysis, and (3) this value is used in the analytical expression above to determine the theoretical value for δ that should have caused this level of force. The difference between the values of δ applied to the model, and that obtained from the analytical expression are the quantity of interest type "error measure" used in this study.

The analytical solution for this problem is not exact (and is thus a *surrogate solution*) not only because it is based upon linear elasticity, but also because it is based upon the simplifying approximations presented by Hertz. These approximations include: (1) a representation of the contact surfaces by quadratic surfaces, (2) the deformation response of each body can be approximated by the solution of a loaded half-space, and (3) relative displacement between the center and edge points of contact are small compared to the contact radius. These approximations require both the geometric dimensions of the body and the radii of curvature in the contact region (one in the same for this problem) to be much larger than the contact radius. Thus the ideal, in terms of using these approximations, is to adopt an extremely small contact area, but then that requires a mesh that efficiently uses small elements near the contact but transitions to larger elements away from this region for the sake of numerical efficiency. In defining this problem, we sought to find a balance between test run times and sufficient accuracy to obtain a measure of convergence.

Again note that since the reference solution is not exact the difference in the solutions is not really the error, though it may be close to the actual error for coarser meshes. The difference value that the solution levels off to, actually is a measure of the error in the reference solution, assuming that the finite element solution is actually converging to the exact solution.

2.6.3. Verification Results

As noted above, the quantity of interest in this test (for the analytical reference solution) is the boundary displacement, δ. The slopes of the relative error curves between the data points (corresponding to two meshes, on the log-log plots) yield observed rates of convergence. For an exact reference solution, the observed rate of convergence approaches the asymptotic rate with mesh refinement, assuming other sources of numerical error (e.g., solver accuracy) do not corrupt the results. For this problem we are not using an exact solution, so an improvement in the convergence estimate is not guaranteed. As previously noted, typically for problems without an exact solution there is (or we hope for) a "sweet range" where the approximations are in the asymptotic range but not refined enough to measure the inexactness of the references solution. Of course the size of this "sweet range" is problem dependent, e.g., in this problem we have not only the approximations associated with linear elasticity but also those associated with the Hertz solution.

Initially we will examine the observed rates of convergence based upon the approximate reference solution.

2.6.3.1. Results based on Hertz reference solution

In this section, we are showing results that are labeled as extended and nightly, the former of which have longer run times. Extended results have four meshes, and the nightly results have two meshes. The following tables give the observed rates of convergence (nightly) for the node-face variation of the Dash contact algorithm and the two Hex8 element formulations between each sequential pair of meshes, where h_{fine} denotes the relative element size of the finer mesh of the pair. The face-face variation of the Dash contact algorithm is not shown for the nightly results because it takes longer to run than currently allowed in the nightly testing process. The following plot shows the corresponding graphical representations of the difference data as a function of the element size. The results appear to be converging to a difference, that is on the order of two percent.

Table 2-16. Observed convergence rates based upon the Hertz reference solution–Nightly.

Node/face

Mean quadrature		Fully integrated	
h_{fine}	$\|\|\delta_{error}\|\|_2/\|\|\delta_{analyt}\|\|_2$	h_{fine}	$\|\|\delta_{error}\|\|_2/\|\|\delta_{analyt}\|\|_2$
0.5000	4.6057	0.5000	1.3373
0.2500	-2.9255	0.2500	0.9518

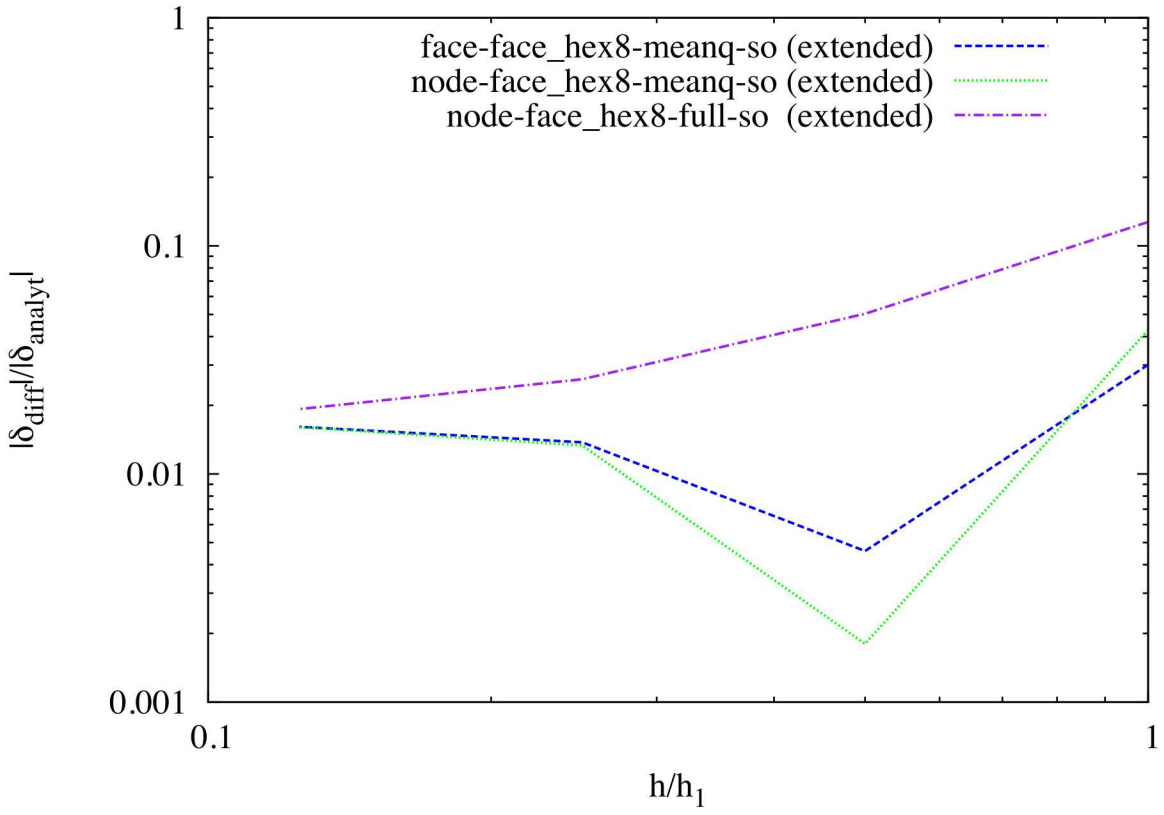


Figure 2-19. Convergence of the displacement boundary condition versus element size.

2.6.3.2. Results based on asymptotic analysis

The asymptotic analysis applied in this verification problem can be considered as consisting of two steps: one to obtain the rate of convergence, and one to obtain a higher order estimate of the solution from Richardson's Extrapolation. This is discussed in more detail in the introduction of this manual. The tables below present the results from the first step: the estimated convergence rates from sequences of three meshes. Two values are presented in the table, one for the normal force (P), and one for the contact radius (a) calculated from P . Calculating the contact radius from P, in a sense just makes it a measure of P, and both quantities yield nearly the same rates of convergence. The rates of convergence are nearly quadratic for the reaction force for all test cases. The consistency of the convergence rates (for a given quantity of interest but different sets of three meshes) suggests the results are approaching the asymptotic range, but the ideal is to have rates that are much closer.

Table 2-17. Observed convergence rates based upon asymptotic analysis.

Face/face

Mean quadrature

h_{fine}	P	a
0.2500	1.84	1.87
0.1250	1.94	1.95

Node/face

Mean quadrature

h_{fine}	P	a
0.2500	1.85	1.89
0.1250	2.06	2.08

Fully integrated

h_{fine}	P	a
0.2500	1.86	1.78
0.1250	1.90	1.87

Since we use a sequence of three numerical results in the asymptotic analysis (giving us three equations), we can solve for the two remaining unknowns: the constant (c) and the estimate of the exact solution (which is one order more accurate than that given by the finite element solution, assuming the next term in the error expansion is one order higher); this part of the asymptotic analysis corresponds to Richardson extrapolation. We then use the higher order estimate of the exact solution (labeled by *RE*) as our reference solution. Admittedly, this higher order solution estimate is better suited for uncertainty quantification [2,3], but we will still use it here as a reference solution to show that it yields the desired linear relationship between error and discretization on a log-log plot (for P). Following the same order as we did above for the analytical solution, first consider the convergence rates obtained using P_{RE} and a_{RE} as the reference solutions, in tabular form. These results are obtained from pairs of meshes, and by definition approach the same values obtained from the asymptotic analyses with mesh refinement.

Table 2-18. Observed convergence rates based upon the Richardson extrapolation references, P_{RE} and a_{RE}.

Face/face

Mean quadrature

h_{fine}	P	a
0.5000	1.87	1.89
0.2500	1.94	1.95
0.1250	1.94	1.95

Node/face

Mean quadrature

h_{fine}	P	a
0.5000	1.90	1.94
0.2500	2.06	2.08
0.1250	2.06	2.08

Fully integrated

h_{fine}	P	a
0.5000	1.87	1.80
0.2500	1.90	1.87
0.1250	1.90	1.87

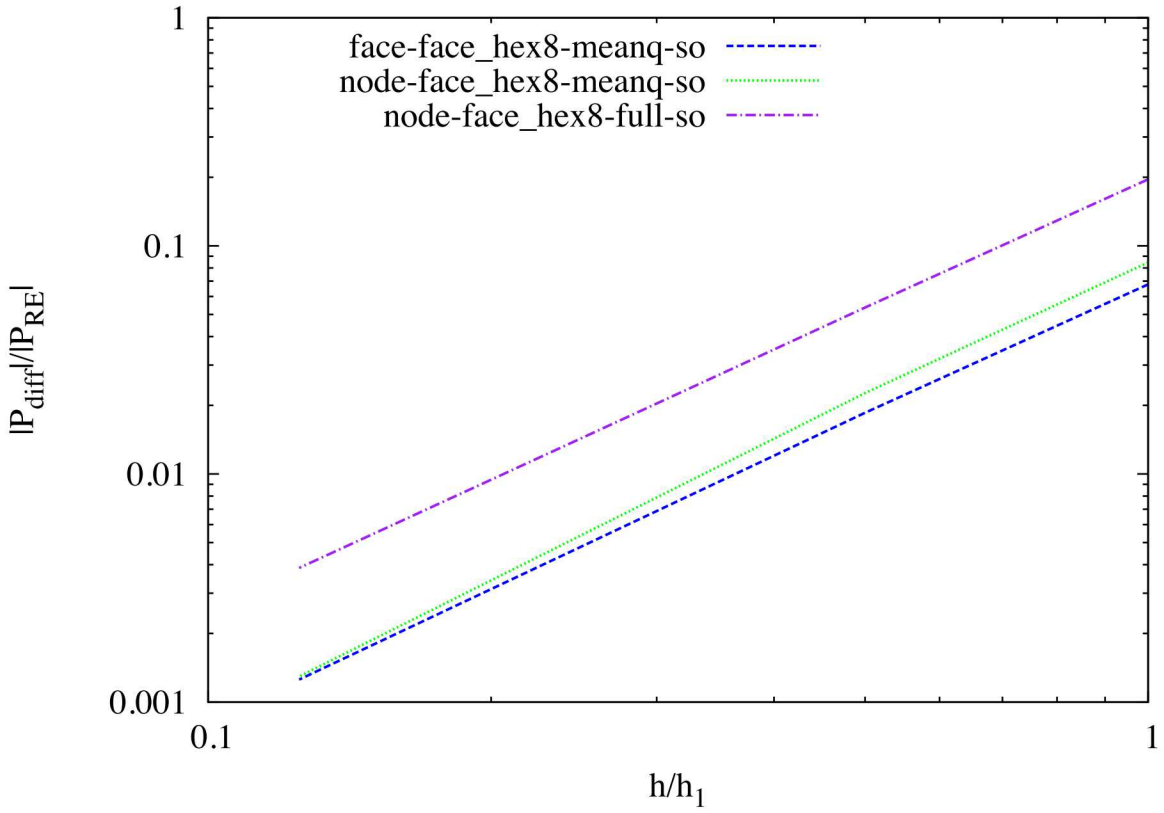


Figure 2-20. Convergence of the normal force, P, versus element size, Richardson extrapolation reference solution.

The above results suggest that reasonable accuracy is obtained (about 2 percent difference) for the contact force, using the finest meshes. The asymptotic results also enforce the interpretation that the convergence to a constant difference when using the analytical reference solution was an indication of the weaknesses in the analytical solution not the contact algorithm.

While the above results indicate that the algorithm is giving nearly quadratic convergence in the response, it does beg the questions of whether these results are as close as the algorithm can come to producing quadratic convergence, whether there is an error in the algorithm producing a reduced rate of convergence, or whether other aspects of the numerical simulation are polluting the observed rates of convergence. Frankly, we do not expect the algorithm to maintain the optimal rate of convergence associated with the elements, but it is still worth considering the other factors that can reduce the observed rate of convergence; among the other factors are relaxed solver tolerances that reduce the accuracy of the solution, and a mixture of the order of the algorithms that has not been accounted for in the convergence study. The solver tolerances were adjusted to be as tight as possible while still yielding a converged solution. The second issue however was purposefully not completely addressed in the above results to keep the analysis times smaller; specifically, the elastic material model is a hypoelastic model and thus is numerically integrated in time. At best we would expect quadratic convergence in time, and thus for the asymptotic terms associated with both space and time to be consistently reduced (assuming quadratic convergence in time) we should have reduced the time step by a factor of one half with each mesh refinement. We assumed this effect would be relatively small – though not necessarily negligible, but used the elastic model because it is the underlying elastic model for several commonly used models in Lame [1].

Summary of results: the contact algorithm appears to converge for this classical contact problem, and the difference between the Hertz reference solution and the FEM solutions for the finest meshes is about two percent. The difference results (referencing the Hertz solution) do not lend themselves to directly evaluating the rate of convergence of the contact algorithm, as there are not sufficient data that exhibit asymptotic behavior without being tainted by the inaccuracy of the reference solution. Using the Hertz solution the numerical results approach a constant difference which we interpret in the limit as representing the error in the analytical solution. To enforce this interpretation, we estimated the rate of convergence for the reaction force using asymptotic analysis which "approached quadratic convergence." We interpret these results as positive verification results; however, these results must be weighted with the facts that the analytical reference solution is not exact and the use of asymptotic analysis does not provide as strong of verification as having an exact reference solution [2,3,6].

2.6.4. References

1. Scherzinger, W.M. and Hammerand, D.C. *Constitutive Models in Lame*. Sandia Report SAND2007-5873, September 2007.
2. Oberkampf. W.L. and Roy, C.J. **Verification and Validation in Scientific Computing**. Cambridge University press, 2010.
3. Roach, P.J. **Verification and Validation in Computational Science and Engineering**. Hermosa Publishers, 1998.
4. de Vahl Davis, G. "Natural Convection of Air in a Square Cavity: A Bench Mark Numerical Solution," *International Journal for Numerical Methods in Fluids*, vol 3, pp 249-264.
5. Johnson K.L. **Contact Mechanics**. Cambridge University press, 1985.
6. Cox, J.V. and Mish, K.D. *Sierra Solid Mechanics Example Verification Problems to Highlight the use of Sierra Verification Tools*, Sandia Report SAND2013-2390, 2013.

For input deck see Appendix B.6.

2.7. LUBKIN SPHERE-SPHERE CONTACT – CONVERGENCE TEST

Analysis Type	Quasi-statics
Element Types	Hex8
Element Formulations	Mean Quadrature, Fully Integrated
Strain Incrementation	Strongly Objective
Material Models	Elastic
Verification Category	Convergence
Verification Quantities	Non-dimensional Contact Torque (T)
Number of Tests	4
Keywords	Lubkin, Hertz, Contact, Friction, Convergence

2.7.1. Brief Description

This series of analyses demonstrates the convergence of contact for the classical Lubkin [1] problem. This problem builds on the Hertz problem (sphere on sphere) to develop the normal preload, and then follows that with a torque applied about an axis connecting the center of both hemispheres. Dash contact using both the face/face and node/face formulations is tested. Two types of 8-noded, hexahedral elements are examined, namely (1) uniform gradient (mean quadrature) elements, and (2) fully-integrated elements both with a strongly objective strain incrementation. The first element is the most commonly used element and the second one (loosely speaking) provides a bound on the element formulations (in terms of integration). Note that there are two other closely related verification problems in the manual: Elastic Spheres in Frictional Torsional Contact (sphere on plate load-deflection test), and Elastic Spheres in Frictional Torsional Contact (sphere on plate convergence test).

2.7.1.1. Functionality Tested

Primary capabilities:

- Dash contact face-face and node-face formulations

Secondary capabilities:

- The following element formulations:

  (1) eight-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation.

  (2) eight-node hexahedron with the mean quadrature formulation and strongly objective strain incrementation.

- Prescribed displacement boundary conditions

2.7.1.2. Mechanics of Test

The geometry consists of two hemispheres in contact, as depicted in Figure 2-21. SI units are adopted for this problem, and thus the radius of the spheres is 4 meters. The hemispheres, as shown in the figure, have equal radii. The problem is defined as a quasistatic problem under displacement controlled deformation. The problem consist of two loading periods. The first, corresponds to the Hertz problem with normal displacements applied to the hemispheres' flat surfaces to establish a normal force. Note that because we have posed this in terms of displacement boundary conditions, the normal force will change with mesh refinement, as the hemispheres change in compliance. Though symmetry does not necessitate it for this problem, the contact surfaces are frictionless during this first period, so that no tangent frictional forces exist at the start of the second time period. The second period of loading, applies a torque to each of the hemispheres' flat surfaces. The torque loading is also applied by a displacement boundary condition. During this period of loading contact friction is turned on, with a coefficient of friction of 0.3. (For the current version of the input, the means used to change the coefficient of friction – or the friction model in this case, is by applying each friction model in a separate procedure block.)

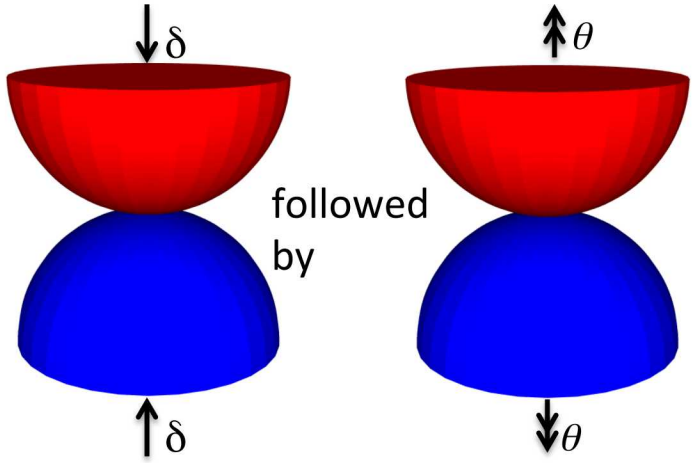


Figure 2-21. Lubkin sphere-sphere contact problem.

2.7.1.3. Material Model

The primary material model used for this problem is the elastic model implemented in Lame [4].
The selected properties were given as follows.

Table 2-19. Material model properties.

Young's Modulus	E	1.0×10^5 Pa
Poisson's Ratio	ν	0.2

2.7.1.4. Boundary Conditions

The boundary conditions for this problem, as depicted in Figure 2-21, show the horizontal surfaces (symmetry cuts) of the two hemispheres have prescribed vertical displacements, denoted as δ. The maximum value of δ, which is the state at which the response is measured, is 2 cm. The horizontal symmetry cuts of the spheres allow us to define these surfaces as displacement reference planes; physically this corresponds to a unit cell out of a stack of spheres. If the objective were to reduce the problem size, it could be reduced further (in this case) to a sphere-plane contact problem; a sphere-plane version of this test exists too. In the second time period, the vertical displacements on the boundaries are held constant, and the in-plane displacement components consist of prescribed displacements in the azimuthal direction. As such the free degrees of freedom correspond to the radial displacements. The prescribed displacements about the cylindrical axis correspond to a maximum rotation of 0.1 radians.

2.7.1.5. Meshes

Figure 2-21 depicts two hemispheres in contact. A contact surface view of the sequence of meshes used in this study (of one hemisphere) are shown in Figure 2-22. Each mesh contains eight times as many elements as the coarser mesh that it is refined from, since $h_i = h_{i-1}/2$, where h_i denotes the characteristic element size for mesh i. The number of elements in each mesh is presented in the table below. The mesh refinements conform to the defining geometry, not the coarser mesh, and as such the solution space for the coarser mesh is not a proper subspace of the solution space for the finer mesh. The table below contains the mesh label, relative element size, and number of elements for each of the meshes.

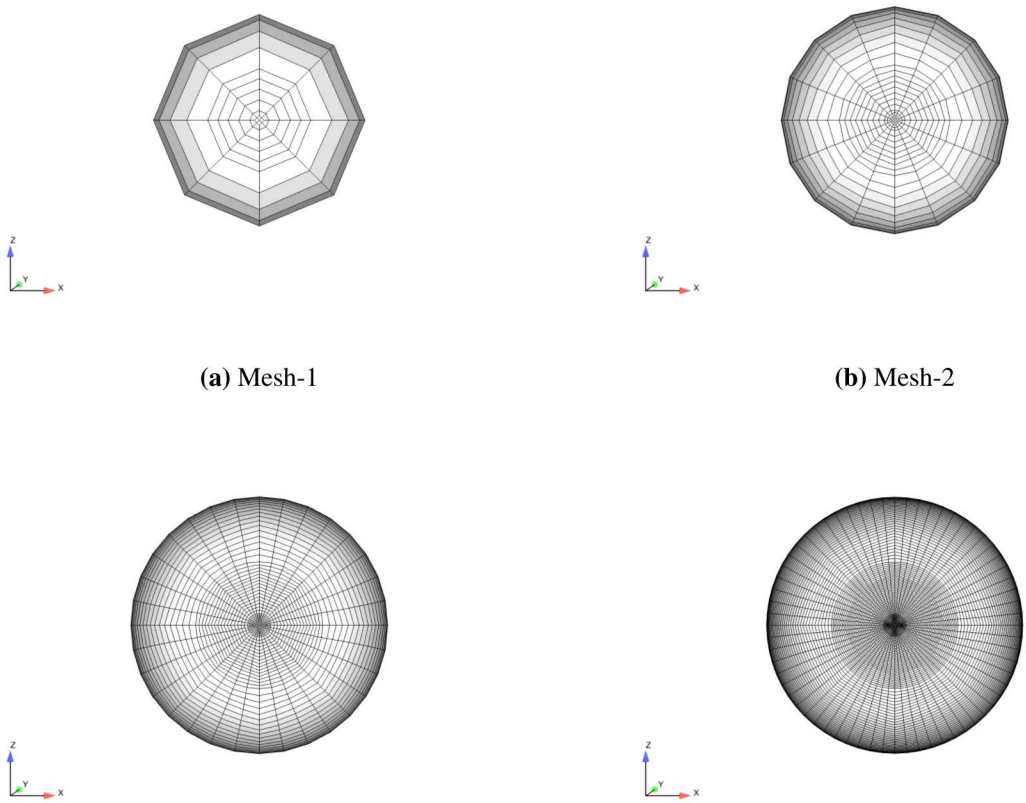
Table 2-20. Mesh characteristics.

Mesh label	h/h_1	Number of Elements
Mesh-1	1	392
Mesh-2	1/2	3136
Mesh-3	1/4	25088
Mesh-4	1/8	200704

2.7.2. Expected Results

For this problem we have evaluated the results in two ways: (1) using an analytical reference solution, and (2) using asymptotic estimates of the rate of convergence based upon results from sequences of three meshes. The analytical reference solution used in this study is taken from the work of Segalman, Starr, and Heinstein [3] and is briefly discussed below. For additional discussion of the asymptotic analysis, please refer to the expected results section of the Hertz cylinder-cylinder contact tests.

The analytical reference solution adopted here [3] is a fourth-order Padé rational function approximation to the analytical solution given by Lubkin [1]. Lubkin's original solution is



(a) Mesh-1

(b) Mesh-2

(c) Mesh-3

(d) Mesh-4

Figure 2-22. Meshes used in this study

expressed in terms of complete elliptic integrals, while the approximate form given in reference [3] is readily amenable to verification and is reported to agree with numerical evaluation of the original solution by Lubkin to within $2\text{x}10^{-5}$ over the full range. The Padé approximation expresses a non-dimensional torque (T) as a function of a non-dimensional twisting angle (θ), defined respectively as

$$\theta \equiv \frac{\beta G a^2}{\mu N}, \tag{2.9}$$

and

$$T \equiv \frac{M}{\mu N a}, \tag{2.10}$$

where $M \sim$ the twisting moment applied, $N \sim$ the contact normal force, $a \sim$ the contact radius, $\mu \sim$ the coefficient of friction, $G \sim$ the shear modulus, and $\beta \sim$ the angle of twist.

$$T(\theta) = \left[\sum_{k=0}^{4} a_k \theta^k\right] \left[\sum_{k=0}^{4} b_k \theta^k\right]^{-1}. \tag{2.11}$$

where the parameters are given in Table 2-21.

Table 2-21. Padé approximation data

a_0	0	b_0	1
a_1	16/3	b_1	5.1193
a_2	6.0327	b_2	15.6833
a_3	19.6951	b_3	30.8099
a_4	42.5359	b_4	72.2111

We apply the this equation in the following manner: (1) the hemispheres are preloaded with a prescribed normal displacement δ_y, (2) the hemispheres are then displaced laterally to a prescribed displacement δ_x, (2) the vertical force (equivalent in magnitude to the contact force, N) is calculated in the FE analysis, (3) the contact radius is calculated from the Hertz solution for the given preload (N), (3) N and δ_x are used in the analytical expression above to determine the non-dimensional rotation θ, and then (4) equation 2.11 is used to calculate the theoretical value for the non-dimensional torque. The model value for the torque (M) is obtained from the cross product of all the contact forces (on a single hemisphere) with their respective in-plane position vectors from the vertical axis (connecting the sphere centers). The quantity of interest, for which errors are calculated, is the non-dimensional torque.

The analytical solution for this problem is not exact not only because it is based upon linear elasticity, but also because it is based upon the simplifying approximations presented by Hertz. These approximations include: (1) a representation of the contact surfaces by quadratic surfaces, (2) a component of the deformation response of each body can be approximated by the solution of a loaded half-space, and (3) relative displacement between the center and edge points of contact are small compared to the contact radius. These approximations require both the geometric dimensions of the body and the radii of curvature in the contact region (one in the same for this problem) to be much larger than the contact radius. Thus the ideal, in terms of using these approximations, is to adopt an extremely small contact area, but then that makes it more difficult to define a mesh that efficiently uses small elements near the contact but transitions to larger elements away from this region (for the sake of numerical efficiency). In defining this problem, we initially sought to find a balance between test run times and sufficient accuracy to obtain a measure of convergence, but admittedly pushed the upper limit of the contact size.

Since the reference solution is not exact the difference in the solutions is not really the error, though it may be close to the actual error for coarser meshes. The error value that the solution levels off to (in the limit) is a measure of the error in the reference solution, assuming that the finite element solution is actually converging to the exact solution. The convergence to a fixed difference between the analytical reference solution and the finite element solutions, is not necessarily monotonic in nature. Because of this convergence behavior for finer meshes, it can be difficult to find a range of discretization for which the approximate reference solution is

sufficiently accurate to serve as a surrogate for the exact solution and yet the meshes are sufficiently fine to be in the asymptotic range. As we will see in this case, we did not obtain a region where the inexact reference solution allowed us to estimate the rate of convergence, but we will observe it converging to a fixed difference. To strengthen the argument that it is converging and to further examine the question of rate, we will estimate the rate of convergence using the approach discussed above and apply Richardson's Extrapolation to estimate the exact solution; unfortunately even the asymptotic solutions for this problem, and corresponding models, do not provide a good estimate of the rate of convergence. Finer meshes would be needed to establish that several of the approximate analyses were in the asymptotic range.

2.7.3. Verification Results

As noted above, the quantity of interest in this test is the non-dimensional torque, T. The slopes of the relative error curves between the data points (corresponding to two meshes, on the log-log plots) yield observed rates of convergence. For an exact reference solution, the observed rate of convergence approaches the asymptotic rate with mesh refinement, assuming other sources of numerical error (e.g., solver accuracy) do not corrupt the results. For this problem we are not using an exact solution, so an improvement in the convergence estimate is not guaranteed. As previously noted, typically for problems without an exact solution there is (or we hope for) a sweet range where the approximations are in the asymptotic range but not refined enough to measure the inexactness of the references solution. Of course the size of this sweet range is problem dependent, e.g., in this problem we have not only the approximations associated with linear elasticity but also those associated with the Hertz solution.

Because the analyses associated with the finest mesh can be very time consuming, a different approach is being taken in presenting the results. The nightly analyses only use the coarsest three meshes, but they are plotted (in one case) with the results for finer meshes too for graphical comparison. Likewise, tables are presented based upon nightly results and those obtained from analyses that include the finest mesh (extended results). As such passing of the tests is not based upon results from the finest mesh, but rather upon change in the convergence rates for the nightly tests. Note that the tabular results can differ between the extended and nightly analyses; currently the extended and nightly analyses both use 30 time steps for the compression preload (Hertz part) but use 20 and 30 time steps for the torsional loading, respectively. (The intent is to obtain extended analyses results that use 30 steps for the torsional loading as well.) Also, due to time constraints of nightly testing, the nightly results are only for the node-face contact formulation.

Initially we will examine the observed rates of convergence based upon the approximate reference solution.

2.7.3.1. Results based on Hertz reference solution

The following tables give the observed rates of convergence for the two variations of the Dash contact algorithm and the two Hex8 element formulations between each sequential pair of meshes, where h_{fine} denotes the relative element size of the finer mesh of the pair. The following plots show the corresponding graphical representations of the error data as a function of the

element size. The first set of tables present the extended results, and the second set present the nightly results.

Table 2-22. Observed convergence rates based upon the Lubkin reference solution – Extended results.

Face/face

Mean quadrature

h_{fine}	*Torque convergence rate*
0.5000	1.25
0.2500	0.82
0.1250	0.36

Fully integrated

h_{fine}	*Torque convergence rate*
0.5000	1.00
0.2500	0.52
0.1250	0.21

Node/face

Mean quadrature

h_{fine}	*Torque convergence rate*
0.5000	1.29
0.2500	1.32
0.1250	0.43

Fully integrated

h_{fine}	*Torque convergence rate*
0.5000	0.41
0.2500	1.09
0.1250	0.40

Table 2-23. Observed convergence rates based upon the Lubkin reference solution – Nightly results.

Node/face

Mean quadrature

h_{fine}	*Torque convergence rate*
0.5000	1.3194
0.2500	1.3240

Fully integrated

h_{fine}	*Torque convergence rate*
0.5000	0.6168
0.2500	1.0738

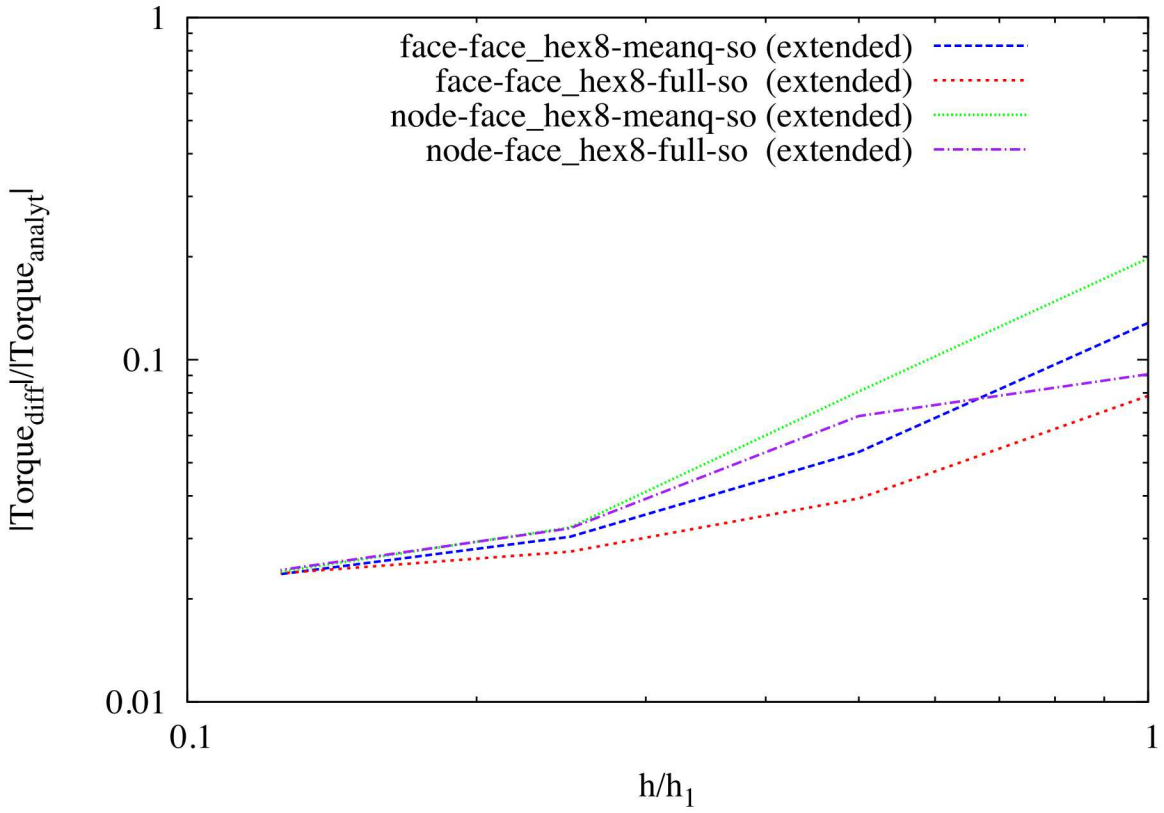


Figure 2-23. Convergence of the non-dimensional Torque versus element size (analytical reference solution).

The difference in the solutions ($Torque_{diff}$) for the finer two meshes is about 3 percent or less, thus giving reasonably good agreement with the analytical reference solution. While the results suggest that each test case is converging (just not to the analytical reference solution), as previously noted, the inexactness of the reference solution makes an estimate of the rate of convergence intractable for the selected models. To examine the convergence rate further we resort to asymptotic analyses of the numerical results alone (i.e., without assuming a reference solution) in the following section.

2.7.3.2. Results based on asymptotic analysis

The asymptotic analysis applied in this verification problem can be considered as consisting of two steps. First, the results from sequences of analyses based upon three mesh refinements, where each refinement halves the characteristic length of the element (i.e., each hex is approximately subdivided, into eight hex elements), are used to estimate the rate of convergence. Second, the convergence rate obtained from the finest sequence of meshes is assumed to be accurate, and then is used with Richardson extrapolation to obtain a higher order estimate of the exact solution. The Richardson's extrapolated estimate is then adopted as the reference solution to analyze the results, as the analytical reference solution was used in the previous section.

Using sequences of three numerical results one can solve for the observed rate of convergence. The non-dimensional torque (T) is again treated as the quantity of interest from which that rate of convergence is estimated. Note that for this problem we do not have sufficient consistency between the results (for successive sequences of three meshes) to definitely claim that we are in the asymptotic range of convergence. As done previously for with the analytical reference solution, we again include results using finer meshes (extended results) and results based upon nightly results.

Table 2-24. Observed convergence rates based upon asymptotic analysis – Extended results.

Face/face

Mean quadrature		Fully integrated	
h_{fine}	*Torque convergence rate*	h_{fine}	*Torque convergence rate*
0.2500	1.68	0.2500	3.16
0.1250	1.84	0.1250	2.02

Node/face

Mean quadrature		Fully integrated	
h_{fine}	*Torque convergence rate*	h_{fine}	*Torque convergence rate*
0.2500	1.37	0.2500	4.80
0.1250	2.36	0.1250	0.27

Table 2-25. Observed convergence rates based upon asymptotic analysis – Nightly results.

Node/face

Mean quadrature		Fully integrated	
h_{fine}	*Torque convergence rate*	h_{fine}	*Torque convergence rate*
0.2500	1.39	0.2500	4.32

Since we use a sequence of three numerical results in the asymptotic analysis (giving us three equations), we can solve for the two remaining unknowns: the constant (c) and the estimate of the exact solution (which is one order more accurate than that given by the finite element solution, assuming the next term in the error expansion is one order higher); this part of the asymptotic analysis corresponds to Richardson's extrapolation. We then use the higher order estimate of the exact solution (labeled by *RE*) as our reference solution. Admittedly, this higher order solution estimate is better suited for uncertainty quantification [5, 6], but we will still use it here as a reference solution to show that by design it yields the desired linear relationship between error and discretization on a log-log plot (for the finest meshes). Following the same order as we did above for the analytical solution, first consider the convergence rates obtained using T_{RE} as the reference solution, in tabular form. These results are obtained from pairs of meshes, and by definition yield the same values obtained from the asymptotic analyses with mesh refinement. As previously presented, extended results are followed by nightly results. For this set of results, the extended and nightly results do not match because the extrapolated reference solution is not based upon the same sets of meshes. In each case the Richardson's extrapolation for the exact solution is obtained from the finest meshes (meshes 2, 3, and 4 for most of the extended analyses and meshes 1, 2, and 3 for the nightly analyses).

Table 2-26. Observed convergence rates based upon the Richardson extrapolation references, T_{RE} – Extended results.

Face/face

Mean quadrature

h_{fine}	$Torque\ convergence\ rate$
0.5000	1.73
0.2500	1.84
0.1250	1.84

Fully integrated

h_{fine}	$Torque\ convergence\ rate$
0.5000	2.95
0.2500	2.02
0.1250	2.02

Node/face

Mean quadrature

h_{fine}	$Torque\ convergence\ rate$
0.5000	1.62
0.2500	2.36
0.1250	2.36

Fully integrated

h_{fine}	$Torque\ convergence\ rate$
0.5000	2.50
0.2500	0.27
0.1250	0.27

Table 2-27. Observed convergence rates based upon the Richardson extrapolation references, T_{RE} – Nightly results.

Node/face

Mean quadrature

h_{fine}	$Torque\ convergence\ rate$
0.5000	1.3932
0.2500	1.3932

Fully integrated

h_{fine}	$Torque\ convergence\ rate$
0.5000	4.3217
0.2500	4.3217

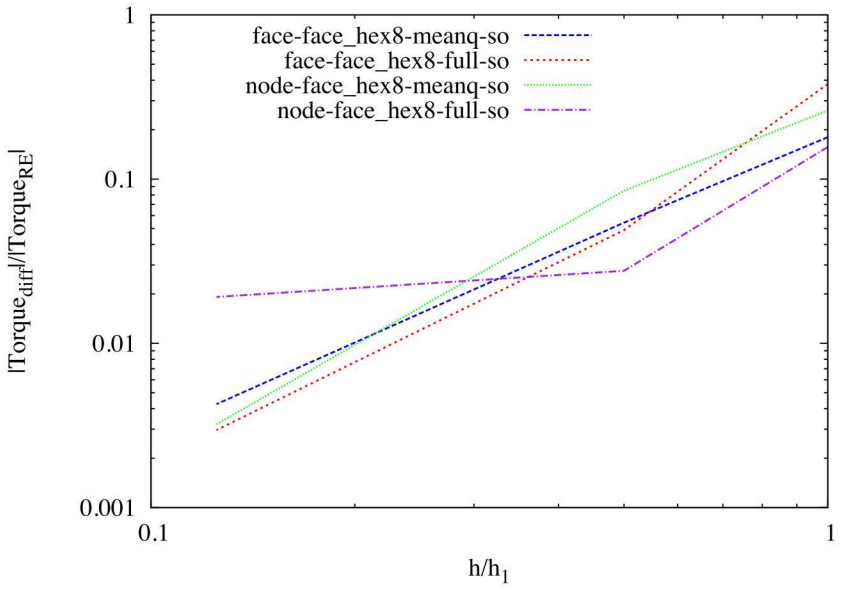


Figure 2-24. Convergence of the non-dimensional Torque versus element size, Richardson extrapolation reference solution using extended results.

For this problem it is difficult to make a strong statement about the rate of convergence even from the results of the asymptotic analysis. The lack of consistency (for a given test, but different sequences of 3 meshes) in the convergence rates (Table 2-22) generally does not indicate that the convergence rate is obtained from results within the asymptotic range. This is reflected in the scatter of the apparent rates of convergence in Tables 2-22 and 2-26 and Figure 2-24. The results suggest that the accuracy of the finite element solutions are within about 2 percent for meshes 3 and 4. The most consistent asymptotic rates fortunately occur for the test case that corresponds to features commonly used by analyst: the face/face contact algorithm and the mean-quadrature element formulation. For this test case the asymptotic rates are much closer (Table 2-22) and indicate that the rate of convergence is much closer to quadratic than linear.

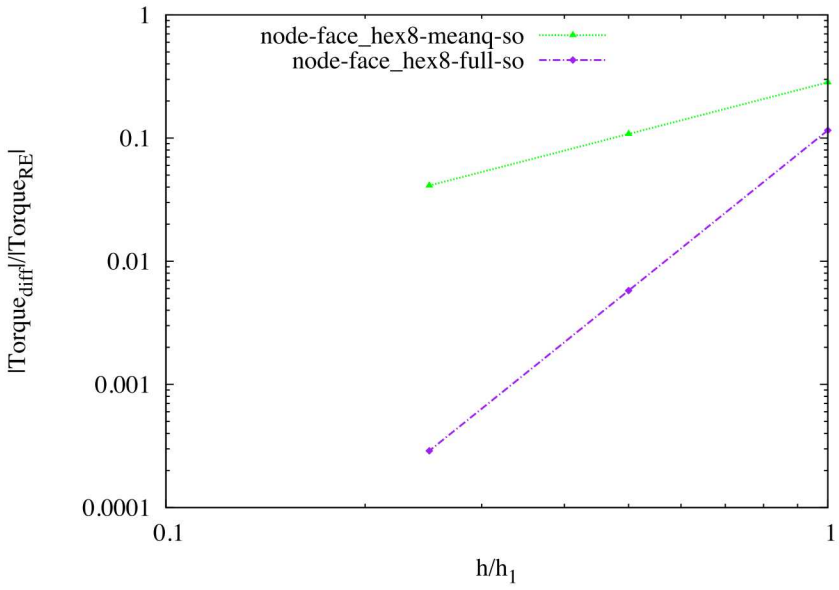


Figure 2-25. Convergence of the non-dimensional Torque versus element size, Richardson extrapolation reference solution using nightly results.

Summary of results: the contact algorithm appears to converge for this classical contact problem, and the difference between the reference solution and the FEM solutions for the finer meshes is less than three percent. The difference results, referencing the Lubkin solution and the Richardson extrapolation approximation of the exact solution, do not lend themselves to directly evaluating the rate of convergence of the contact algorithm with certainty. Nonetheless it is encouraging to note that the test case adopting face/face contact and the mean quadrature element formulation indicate a convergence rate that is closer to quadratic than linear.

We interpret these results as positive verification results, from the view point that all the results show a convergent behavior and yield reasonable agreement with the analytical solution. While, these results must be weighted with the facts that neither an inexact analytical reference solution nor use of asymptotic analysis provide as strong of verification as having an exact reference solution [5, 6, and 9], for sliding friction it appears to be the best we can do. We have a manufactured solution approach that can potentially be applied to frictionless contact, but extension of the approach to frictional contact has yet to be considered and promises at best to be extremely challenging. Further optimization of the mesh, reduction of the Hertz loading, and extension of the analyses to five meshes are among the candidates that might help us obtain more results in the asymptotic range and thus more convincing data on the convergence rates.

2.7.4. References

1. L. Lubkin, *The Torsion of Elastic Spheres in Contact*, *ASME Journal of Applied Mechanics*, **18**, 1951, pp. 183-187.
2. H. Deresiewicz, *Contact of Elastic Spheres Under an Oscillating Torsional Couple*, *ASME Journal of Applied Mechanics*, **21**, 1954, pp. 52-56.
3. Daniel J. Segalman, Michael J. Starr and Martin W. Heinstein, *New Approximations for Elastic Spheres Under an Oscillating Torsional Couple*, *ASME Journal of Applied Mechanics*, **72**, September 2005, pp. 705-710.
4. W.M. Scherzinger, and D.C. Hammerand, *Constitutive Models in Lame*. Sandia Report SAND2007-5873, September 2007.
5. W.L. Oberkampf and C.J. Roy, C.J, *Verification and Validation in Scientific Computing*. Cambridge University press, 2010.
6. P.J. Roach, *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.
7. G. de Vahl Davis, "Natural Convection of Air in a Square Cavity: A Bench Mark Numerical Solution," *International Journal for Numerical Methods in Fluids*, **3**, pp 249-264.
8. K.L. Johnson, *Contact Mechanics*. Cambridge University press, 1985.
9. J.V. Cox and K.D. Mish, *Sierra Solid Mechanics Example Verification Problems to Highlight the use of Sierra Verification Tools*, Sandia Report SAND2013-2390, 2013.

For input deck see Appendix B.7.

2.8. STICKING-SLIPPING BLOCK AND SPRING - EXPLICIT DYNAMICS

Analysis Type	ExplicitDynamics
Element Type	Hex8, Spring
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Tangent Reaction Force
Number of Tests	1
Keywords	Coulomb Friction, Contact

2.8.1. Problem Description

This test checks the computed frictional force after slipping occurs for a Coulomb friction test. The test has one block placed on top of a larger block. A known vertical force and an increasing horizontal force are applied to the top block such that the maximum frictional force is eventually exceeded. The bottom block is fixed on one side and the top block is held in place by a spring with one end equivalenced to a node at the center of one of that block's faces. The frictional force is measured through the reaction on the fixed end of the spring.

There are three versions of this test that share the bulk of this documentation: quasi-statics, implicit dynamics, and explicit dynamics. These tests solve essentially the same problem, with minor differences due to the differing solution techniques (i.e., implicit versus explicit).

2.8.1.1. Boundary Conditions

The applied vertical force on the top block is a gravity load sinusoidally ramped and then held constant. The horizontal force is sinusoidally ramped after the vertical force has reached its maximum, and the horizontal force is applied at the interaction surface to avoid creating moments around the interaction surface. The bottom block is held fixed in all directions on the side away from the interaction.

2.8.1.2. Material Model

Each block uses an elastic material model where the values were picked for convenience.

Young's Modulus	E	$1e8$ Pa (blocks), $1e7$ Pa (spring)
Poisson's Ratio	ν	0.0
Density	ρ	$1.0e3$ kg/m^3

2.8.1.3. Contact Interaction Model

The two blocks interact through a constant coefficient Coulomb friction model. This model provides no resistance to surface separation (though none is induced here) and a maximum tangential contact force directly proportional to the normal contact force.

Coefficient of Friction	μ	0.5

2.8.1.4. Feature Tested

Sliding frictional contact force calculations.

2.8.2. Assumptions and notes

2.8.3. Verification of Solution

There is an analytic solution since we use a standard Coulomb friction model, which says

$$F_{\text{tang}} \leq \mu F_{\text{norm}}, \tag{2.12}$$

where F_{tang} is the tangential contact force, μ is the coefficient of friction, and F_{norm} is the normal contact force. After the sinusoidal ramp (used to minimize dynamic effects) the applied external vertical force is held constant, which implies that the normal contact force is held constant. While the normal contact force is constant, the applied tangential force is ramped up. At time $t_{\text{slip}} = 13.33$ (determined from the applied force functions in the input file) this tangential applied force exceeds the maximum of the tangential frictional force (given by the above equation). At that point the spring will load and support the applied force that is in excess of the maximum tangential contact force. Thus, the analytic solution for the spring reaction is

$$\begin{aligned} &\text{For} \quad t \leq t_{\text{slip}}, \qquad R_{\text{spring}} = 0; \\ &\text{For} \quad t > t_{\text{slip}}, \qquad R_{\text{spring}} = F_{\text{app\_tang}} - \mu F_{\text{norm}}. \end{aligned} \tag{2.13}$$

In the following figure the spring reaction is plotted with an analytic curve that is the solution for the spring reaction for $t > t_{\text{slip}}$. As you can see, the spring reaction is zero for $t \leq t_{\text{slip}}$ and matches the analytic solution for $t > t_{\text{slip}}$. For the quasistatic, implicit dynamics, and explicit dynamics cases these results are checked to within 1%, 2%, and 5%, respectively, of the maximum tangential contact force.

For input deck see Appendix B.8.

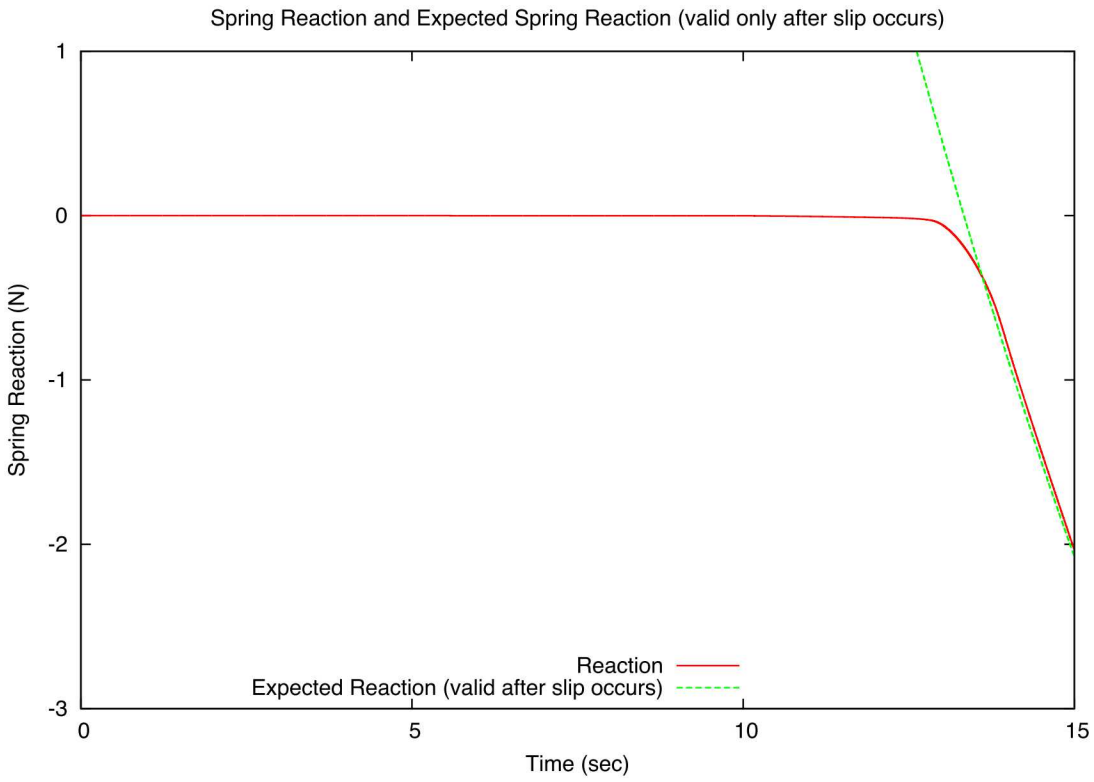


Figure 2-26. Spring Reaction Comparison

2.9. STICKING-SLIPPING BLOCK AND SPRING - IMPLICIT DYNAMICS

Analysis Type	ImplicitDynamics
Element Type	Hex8, Spring
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Tangent Reaction Force
Number of Tests	1
Keywords	Coulomb Friction, Contact

2.9.1. Problem Description

This test checks the computed frictional force after slipping occurs for a Coulomb friction test. The test has one block placed on top of a larger block. A known vertical force and an increasing horizontal force are applied to the top block such that the maximum frictional force is eventually exceeded. The bottom block is fixed on one side and the top block is held in place by a spring with one end equivalenced to a node at the center of one of that block's faces. The frictional force is measured through the reaction on the fixed end of the spring.

There are three versions of this test that share the bulk of this documentation: quasi-statics, implicit dynamics, and explicit dynamics. These tests solve essentially the same problem, with minor differences due to the differing solution techniques (i.e., implicit versus explicit).

2.9.1.1. Boundary Conditions

The applied vertical force on the top block is a gravity load sinusoidally ramped and then held constant. The horizontal force is sinusoidally ramped after the vertical force has reached its maximum, and the horizontal force is applied at the interaction surface to avoid creating moments around the interaction surface. The bottom block is held fixed in all directions on the side away from the interaction.

2.9.1.2. Material Model

Each block uses an elastic material model where the values were picked for convenience.

Young's Modulus	E	$1e8$ Pa (blocks), $1e7$ Pa (spring)
Poisson's Ratio	ν	0.0
Density	ρ	$1.0e3$ kg/m^3

2.9.1.3. Contact Interaction Model

The two blocks interact through a constant coefficient Coulomb friction model. This model provides no resistance to surface separation (though none is induced here) and a maximum tangential contact force directly proportional to the normal contact force.

Coefficient of Friction	μ	0.5

2.9.1.4. Feature Tested

Sliding frictional contact force calculations.

2.9.2. Assumptions and notes

2.9.3. Verification of Solution

There is an analytic solution since we use a standard Coulomb friction model, which says

$$F_{\text{tang}} \leq \mu F_{\text{norm}}, \tag{2.14}$$

where F_{tang} is the tangential contact force, μ is the coefficient of friction, and F_{norm} is the normal contact force. After the sinusoidal ramp (used to minimize dynamic effects) the applied external vertical force is held constant, which implies that the normal contact force is held constant. While the normal contact force is constant, the applied tangential force is ramped up. At time $t_{\text{slip}} = 13.33$ (determined from the applied force functions in the input file) this tangential applied force exceeds the maximum of the tangential frictional force (given by the above equation). At that point the spring will load and support the applied force that is in excess of the maximum tangential contact force. Thus, the analytic solution for the spring reaction is

$$\begin{aligned} &\text{For} \quad t \leq t_{\text{slip}}, \qquad R_{\text{spring}} = 0; \\ &\text{For} \quad t > t_{\text{slip}}, \qquad R_{\text{spring}} = F_{\text{app\_tang}} - \mu F_{\text{norm}}. \end{aligned} \tag{2.15}$$

In the following figure the spring reaction is plotted with an analytic curve that is the solution for the spring reaction for $t > t_{\text{slip}}$. As you can see, the spring reaction is zero for $t \leq t_{\text{slip}}$ and matches the analytic solution for $t > t_{\text{slip}}$. For the quasistatic, implicit dynamics, and explicit dynamics cases these results are checked to within 1%, 2%, and 5%, respectively, of the maximum tangential contact force.

For input deck see Appendix B.9.

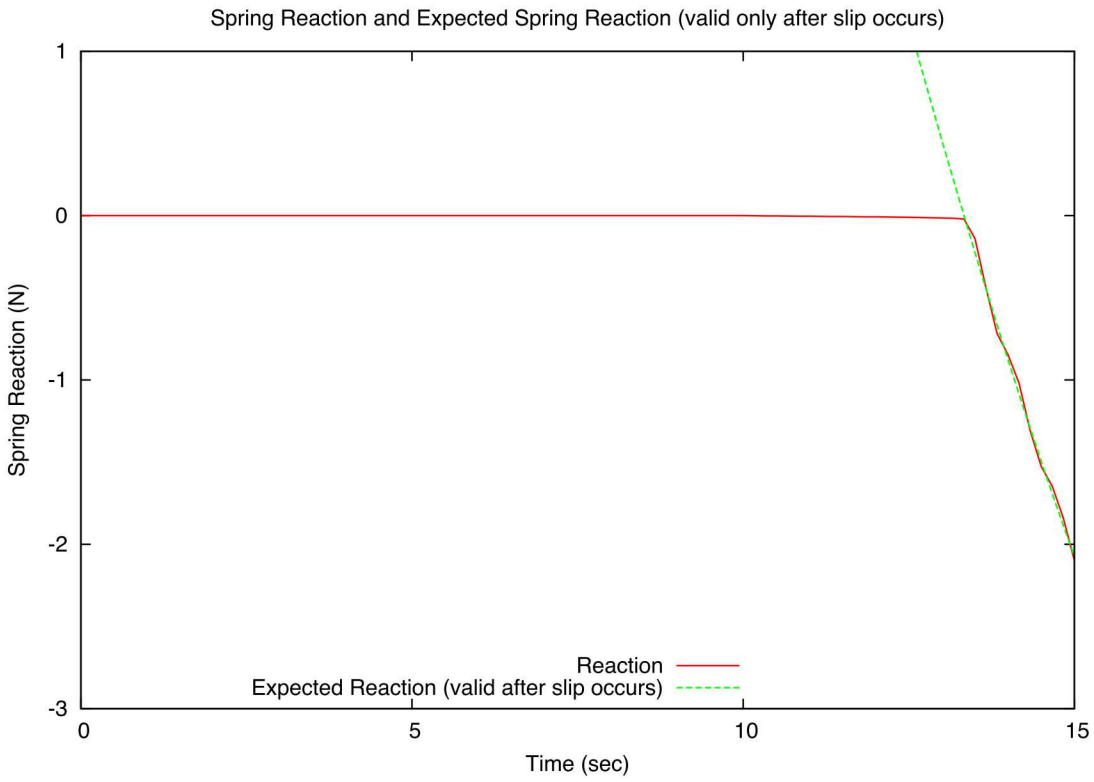


Figure 2-27. Spring Reaction Comparison

2.10. STICKING-SLIPPING BLOCK AND SPRING - IMPLICIT QUASI-STATICS

Analysis Type	Quasistatics
Element Type	Hex8, Spring
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Tangent Reaction Force
Number of Tests	1
Keywords	Coulomb Friction, Contact

2.10.1. Problem Description

This test checks the computed frictional force after slipping occurs for a Coulomb friction test. The test has one block placed on top of a larger block. A known vertical force and an increasing horizontal force are applied to the top block such that the maximum frictional force is eventually exceeded. The bottom block is fixed on one side and the top block is held in place by a spring with one end equivalenced to a node at the center of one of that block's faces. The frictional force is measured through the reaction on the fixed end of the spring.

There are three versions of this test that share the bulk of this documentation: quasi-statics, implicit dynamics, and explicit dynamics. These tests solve essentially the same problem, with minor differences due to the differing solution techniques (i.e., implicit versus explicit).

2.10.1.1. Boundary Conditions

The applied vertical force on the top block is a gravity load sinusoidally ramped and then held constant. The horizontal force is sinusoidally ramped after the vertical force has reached its maximum, and the horizontal force is applied at the interaction surface to avoid creating moments around the interaction surface. The bottom block is held fixed in all directions on the side away from the interaction.

2.10.1.2. Material Model

Each block uses an elastic material model where the values were picked for convenience.

Young's Modulus	E	$1e8$ Pa (blocks), $1e7$ Pa (spring)
Poisson's Ratio	ν	0.0
Density	ρ	$1.0e3$ kg/m^3

2.10.1.3. Contact Interaction Model

The two blocks interact through a constant coefficient Coulomb friction model. This model provides no resistance to surface separation (though none is induced here) and a maximum tangential contact force directly proportional to the normal contact force.

Coefficient of Friction	μ	0.5

2.10.1.4. Feature Tested

Sliding frictional contact force calculations.

2.10.2. Assumptions and notes

2.10.3. Verification of Solution

There is an analytic solution since we use a standard Coulomb friction model, which says

$$F_{\text{tang}} \leq \mu F_{\text{norm}}, \tag{2.16}$$

where F_{tang} is the tangential contact force, μ is the coefficient of friction, and F_{norm} is the normal contact force. After the sinusoidal ramp (used to minimize dynamic effects) the applied external vertical force is held constant, which implies that the normal contact force is held constant. While the normal contact force is constant, the applied tangential force is ramped up. At time $t_{\text{slip}} = 13.33$ (determined from the applied force functions in the input file) this tangential applied force exceeds the maximum of the tangential frictional force (given by the above equation). At that point the spring will load and support the applied force that is in excess of the maximum tangential contact force. Thus, the analytic solution for the spring reaction is

$$\begin{aligned} &\text{For} \quad t \leq t_{\text{slip}}, \qquad R_{\text{spring}} = 0; \\ &\text{For} \quad t > t_{\text{slip}}, \qquad R_{\text{spring}} = F_{\text{app\_tang}} - \mu F_{\text{norm}}. \end{aligned} \tag{2.17}$$

In the following figure the spring reaction is plotted with an analytic curve that is the solution for the spring reaction for $t > t_{\text{slip}}$. As you can see, the spring reaction is zero for $t \leq t_{\text{slip}}$ and matches the analytic solution for $t > t_{\text{slip}}$. For the quasistatic, implicit dynamics, and explicit dynamics cases these results are checked to within 1%, 2%, and 5%, respectively, of the maximum tangential contact force.

For input deck see Appendix B.10.

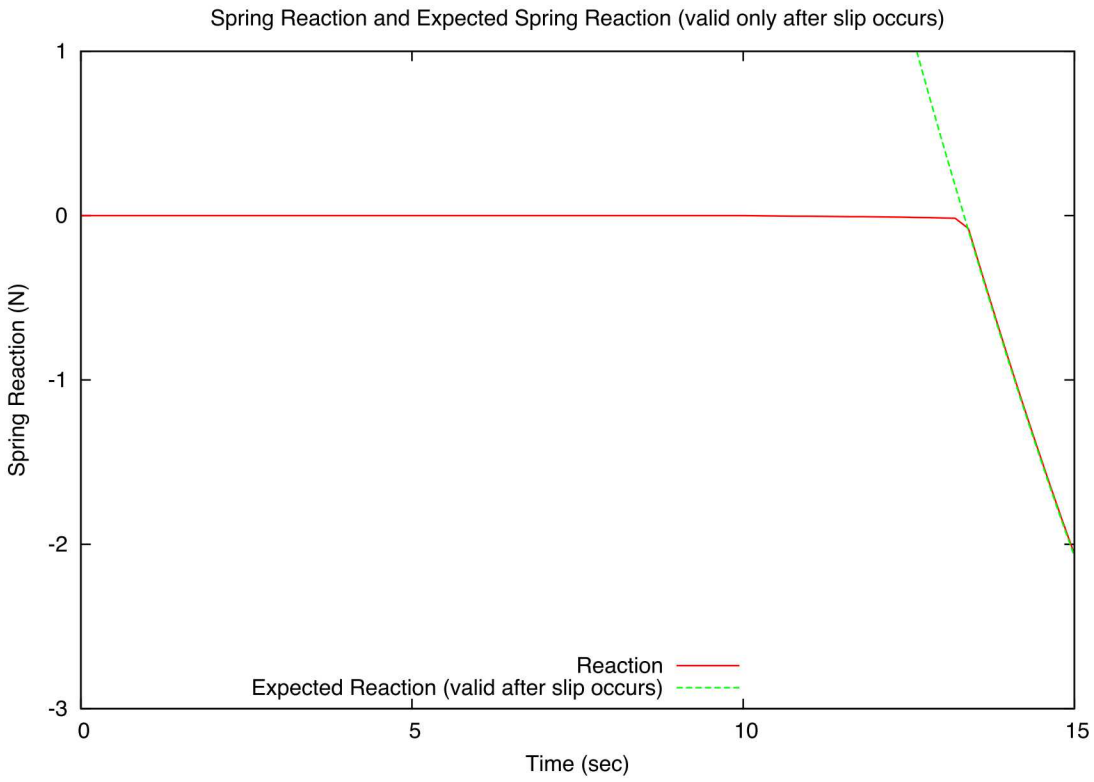


Figure 2-28. Spring Reaction Comparison

2.11. COULOMB FRICTION WITH SLIDING

Analysis Type	Explicit/Implicit Dynamics, Quasi-statics
Element Type	Hex8, Tet4, Rigid Body
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Contact Force, Displacement
Number of Tests	10
Keywords	Force Balance, Contact

2.11.1. Problem Description

This problem puts a scrubbing bubble geometry through a loading history that exercises all the regimes of the coulomb friction law. Figure 2-29 shows the computational mesh. The green block is a rigid body to which normal prescribed forces and displacements are applied to drive the problem. Contact occurs between the yellow tet4 block and the red hex8 block. The four blue cubes are used in the analytic rigid surface contact test cases to define an analytic rigid plane.

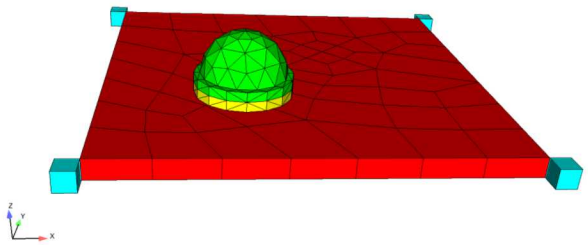


Figure 2-29. Mesh View

2.11.1.1. Boundary Conditions

The loading is accomplished through a combination of prescribed normal forces and kinematically prescribed sliding. The loading conditions are covered using Figure 2-30 and Table 2-28.

2.11.1.2. Material Model

2.11.1.3. Feature Tested

This test exercises the contact enforcement algorithms in explicit dynamics, implicit dynamics, and quasi-statics. This includes the different constraint formulations; node-face augmented

Table 2-28. Loading history

Load Period	Loading Time	Process
1	0.000	Initial condition, pad is located directly above surface, no contact forces produced
2	0.000-0.005	Normal loading force is ramped up
3	0.005-0.010	Block slides a small amount, total x displacement 0.0002
4	0.010-0.015	Block is held
5	0.015-0.020	Normal force released
6	0.020-0.025	Normal force re-applied
7	0.025-0.030	Slide block diagonally a small amount. Total x displacement -0.0002. Total y displacement 0.0002.
8	0.030-0.035	Block is held
9	0.035-0.040	Normal force released
10	0.040-0.045	Normal force re-applied
11	0.045-0.050	Slide block diagonally quickly. Total x displacement 0.05. Total y displacement 0.05.

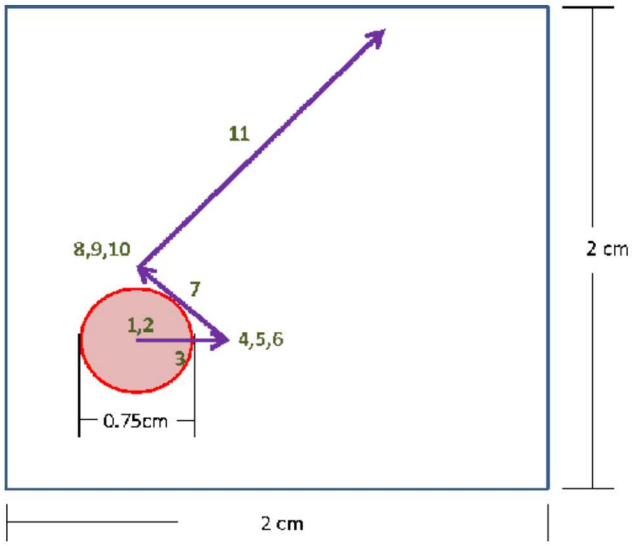


Figure 2-30. Loading History

Young's Modulus	E	1000
Shear Modulus	G	500
Density	ρ	$4.0e-4$

Lagrange, node-face kinematic, face-face augmented Lagrange, and analytic rigid surface contact. The test includes loading histories that test normal gap enforcement, stick/slip transition of coulomb friction law, constant velocity sliding, and variable direction and velocity sliding with a

coulomb friction law.

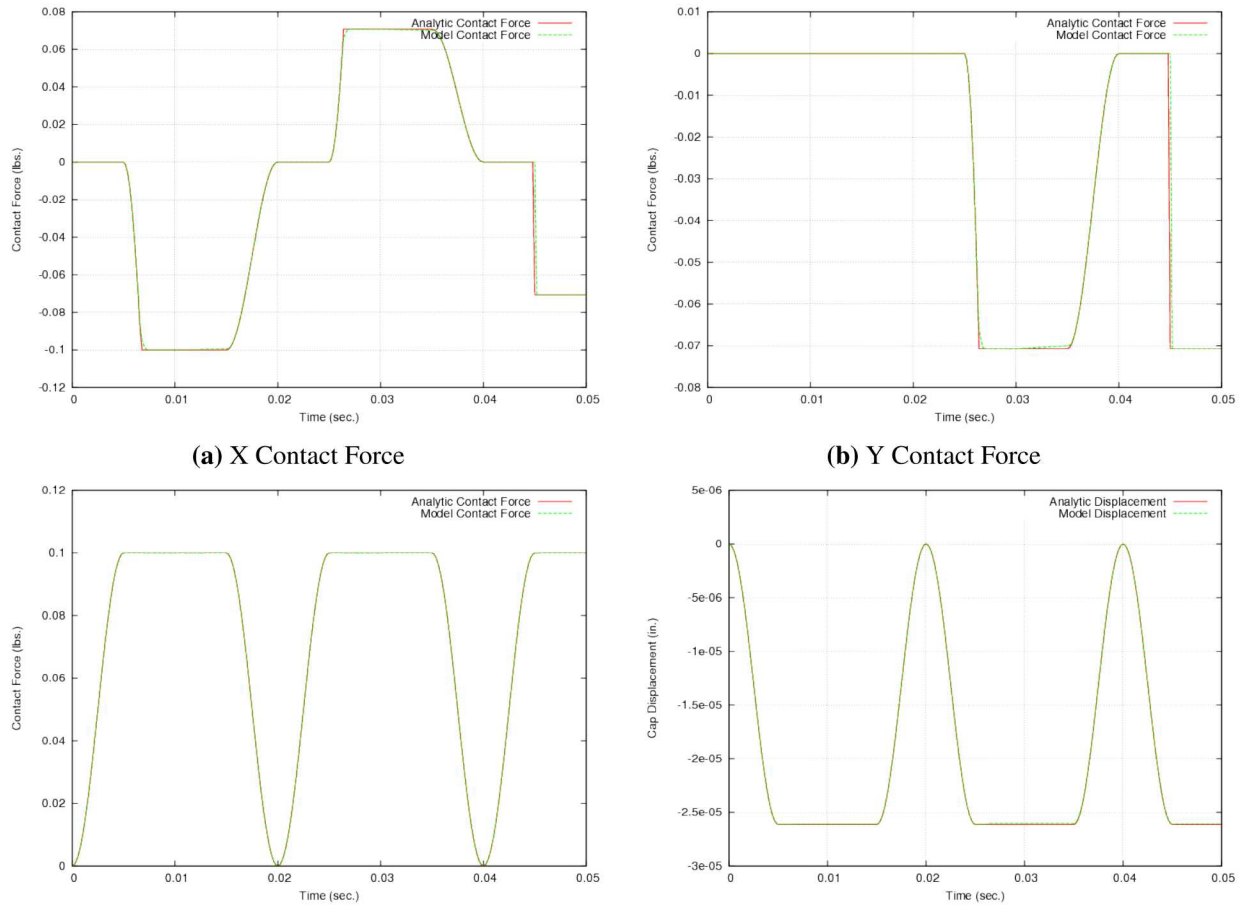
Also tested is total iteration counts and solver efficiency to solve contact systems in implicit dynamics and quasi statics.

2.11.2. Assumptions and notes

Analytic solution assumes quasi-statics. Material densities are set low enough so that the dynamic loading approximates a quasi-static solution. However, in the high slip rate regime dynamics calculations will see some deviation away from a static result.

2.11.3. Verification of Solution

Figure 2-31 shows results of augmented Lagrange static node-face contact (the dash quasistatic enforcement method). For this method all results nearly exactly match the analytic static solutions.



(a) X Contact Force

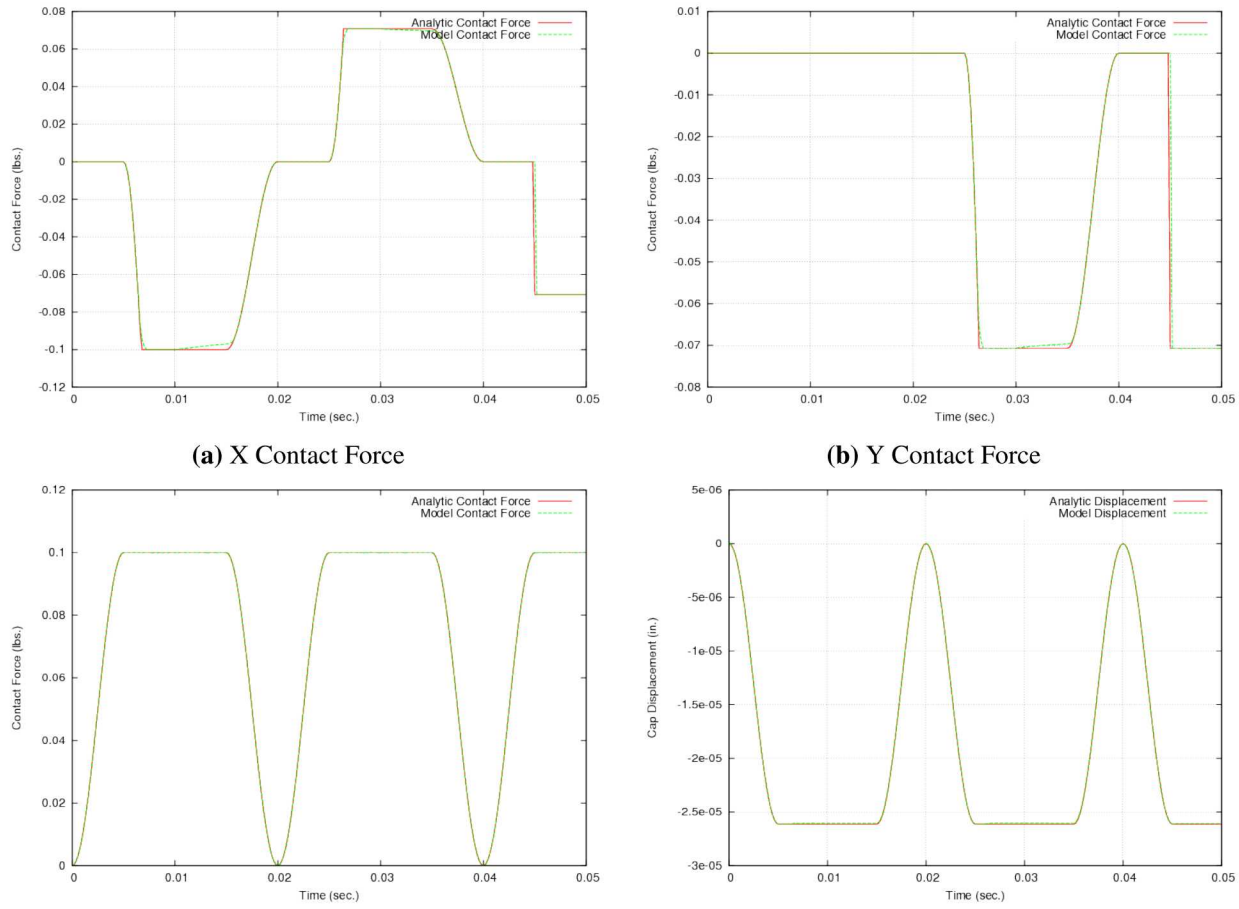
(b) Y Contact Force

(c) Z Contact Force

(d) Z Displacement

Figure 2-31. Static Node-Face

Figure 2-32 shows results of augmented Lagrange static face-face contact (the dash quasistatic enforcement method). For this method all results nearly exactly match the analytic static solutions.



(a) X Contact Force

(b) Y Contact Force

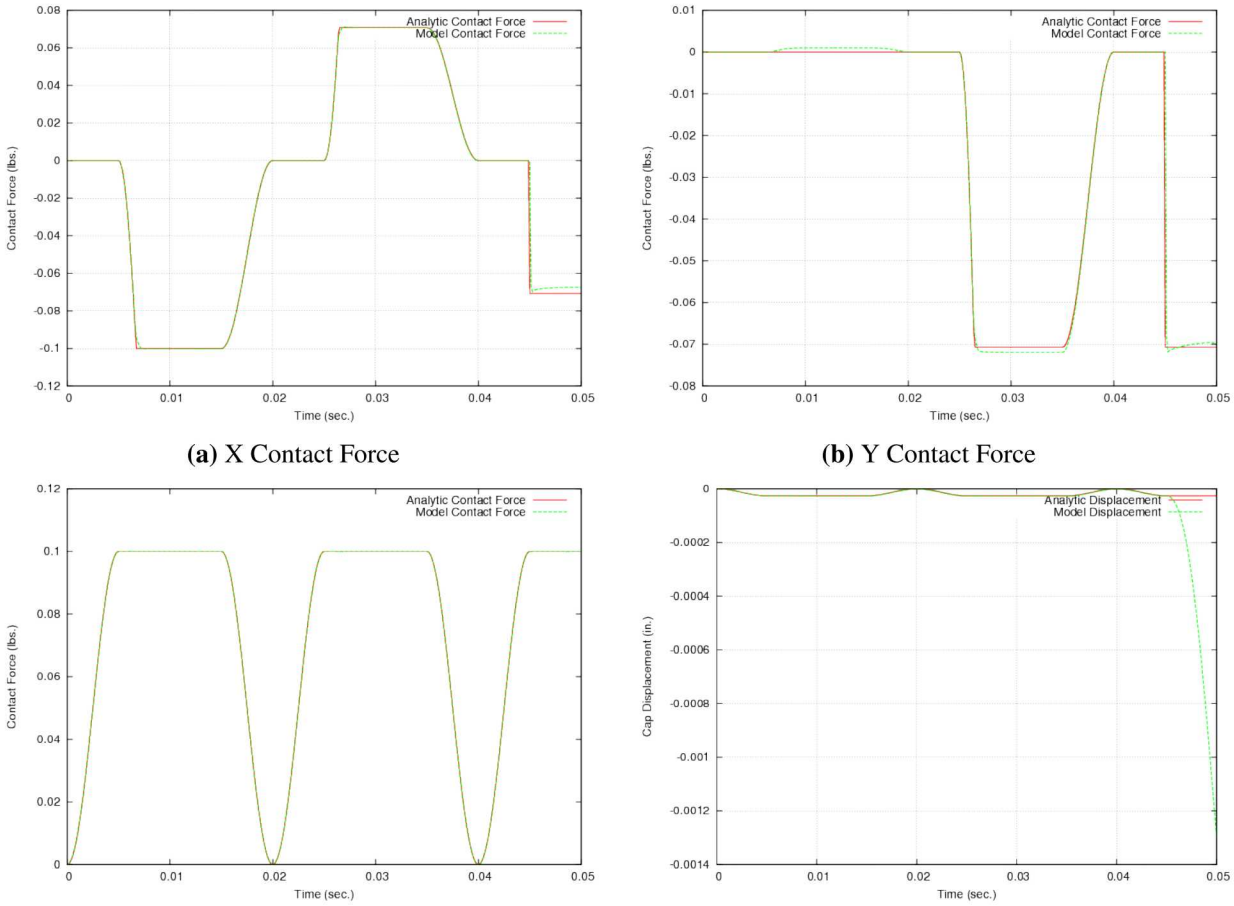
(c) Z Contact Force

(d) Z Displacement

Figure 2-32. Static Face-Face

Figure 2-33 shows results for analytic rigid surface contact (the ARS quasistatic enforcement method). For this case the bubble is sliding on an analytic plane, the corners of which are defined by the four blue cubes. This method gives good agreement to analytic results for loading period 1 to 10. However, in loading period 11, rapid and large motion sliding, the rigid surface enforcement method shows substantial deviations from the correct analytic solution. The reasons for these deviations is currently unknown. Because of these deviations analytic rigid surface contact has only been verified to give the correct results in small sliding regimes.

Figure 2-34 shows results for kinematic node-face contact (the ACME quasistatic enforcement method). For this case the contact forces are computed accurately. However, as seen in plot (d), Z displacement, the displacements may have errors. Kinematic enforcement is computed via a velocity constraint. Any inconsistency in the velocity constraint tend accumulate and after a significant run time may lead to a noticeable error in the displacement. This is an undesirable, but known and expected, limitation of the kinematic contact algorithm.



(a) X Contact Force

(b) Y Contact Force

(c) Z Contact Force

(d) Z Displacement

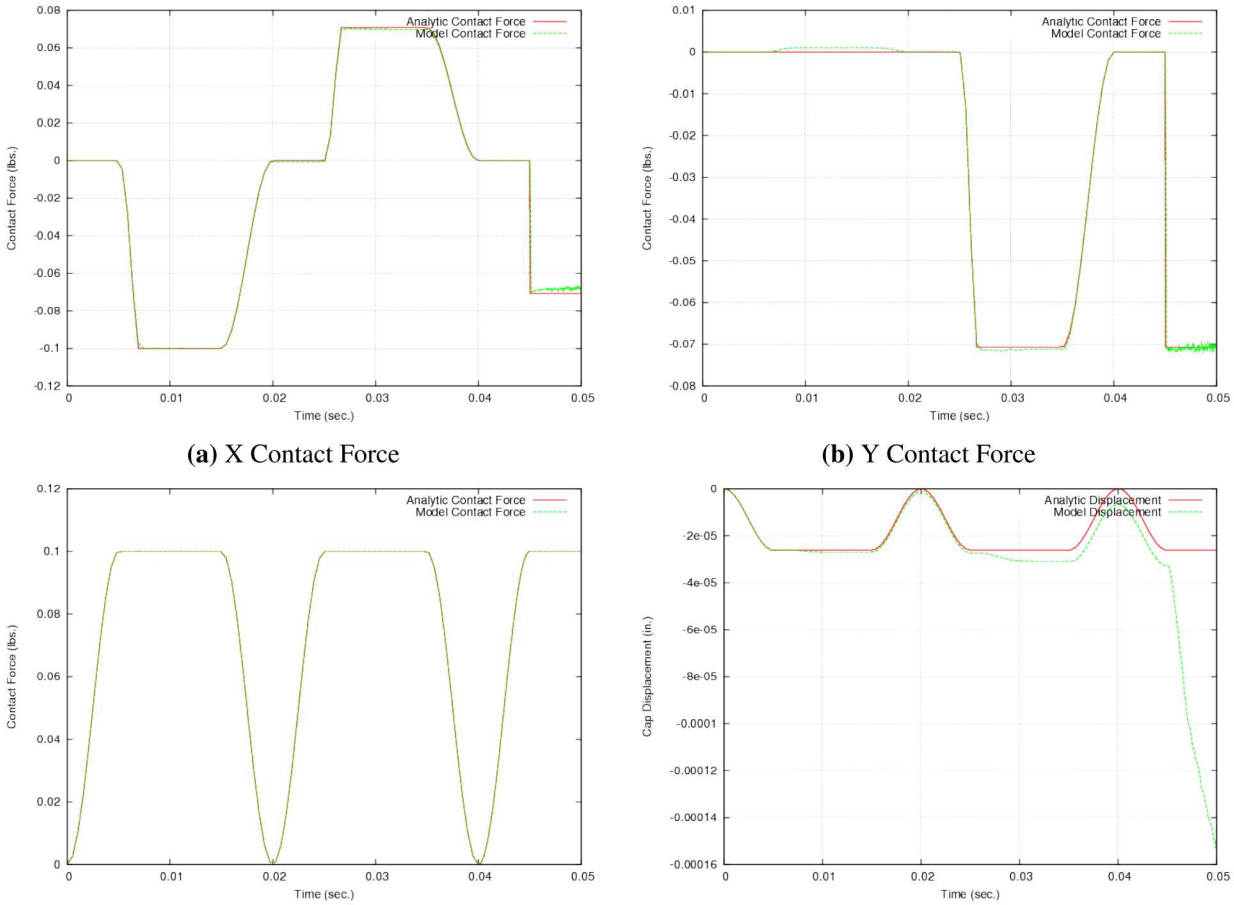
Figure 2-33. Static ARS

Figure 2-35 shows results of augmented Lagrange implicit dynamic node-face contact (the dash implicit enforcement method). For this method all results nearly exactly match the analytic static solutions.

Figure 2-36 shows results of augmented Lagrange implicit dynamic face-face contact (the dash implicit enforcement method). For this method all results nearly exactly match the quasistatic solution for small sliding rates (loading periods 1-10). Period 11 is a high sliding velocity period. In period 11 some high frequency oscillation is introduced into the sliding block. The analytic solution assumes quasistatics, for this case the actual problem does include dynamics thus some deviation from the analytic solution is expected. In the dynamic regime the code results should and do match the analytic results in the average sense once the high frequency noise is filtered out.

Figure 2-37 shows results of analytic rigid surface implicit contact (the ARS implicit enforcement method). As with static ARS contact this method shows good agreement to the analytic solution for small sliding, but large and unexpected deviations from the analytic solution during large magnitude and large velocity sliding.

Figure 2-38 shows results of augmented Lagrange explicit dynamic node-face contact (the dash



(a) X Contact Force

(b) Y Contact Force

(c) Z Contact Force

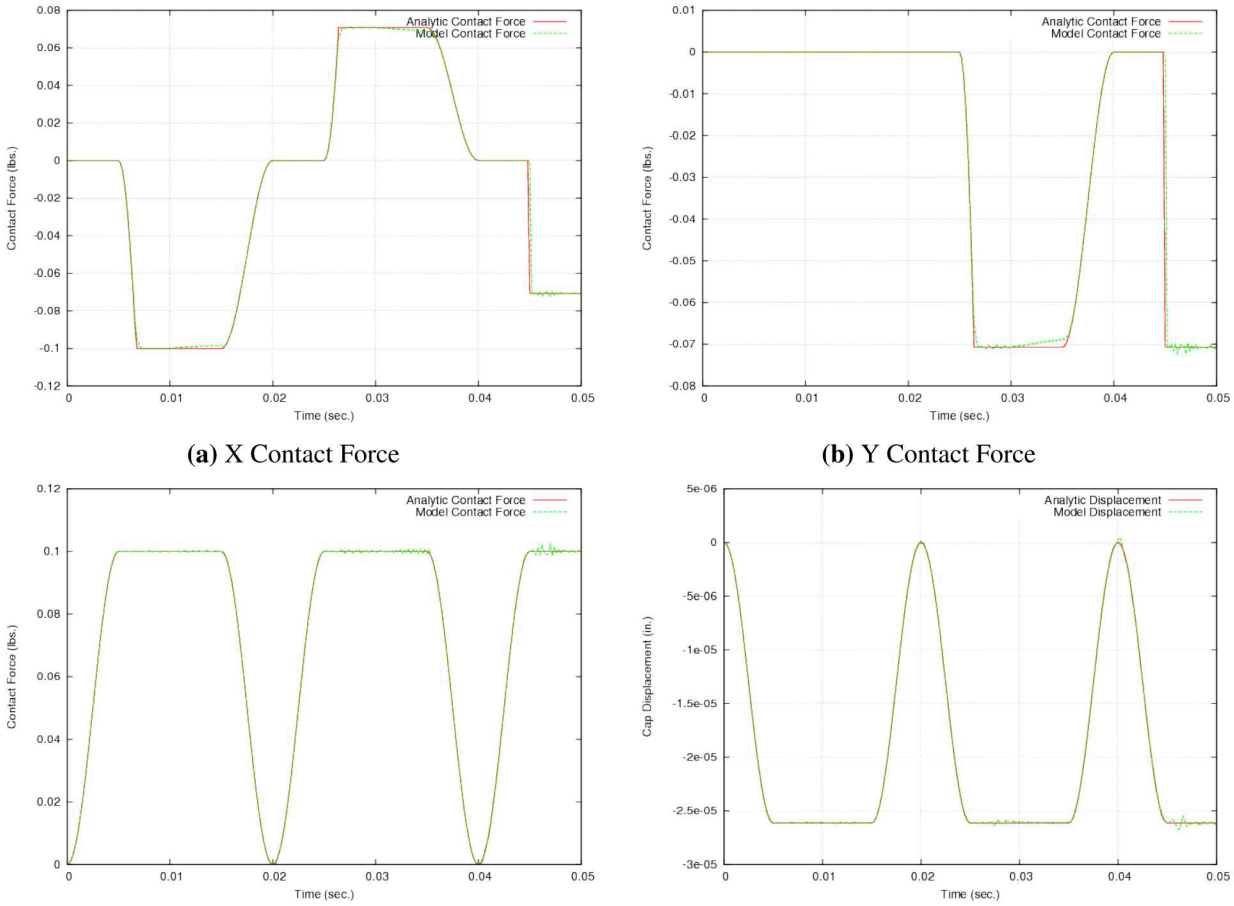
(d) Z Displacement

Figure 2-34. Static Kinematic

explicit enforcement method). This method shows good agreement with the static analytic results for small sliding. Period 11 is a high sliding velocity period. In period 11 some high frequency oscillation is introduced into the sliding block. The analytic solution is modeled quasistatics, for this case the actual problem does include dynamics thus some deviation from the analytic solution is expected. In the dynamic regime the code results should and do match the analytic static results in the average sense once the high frequency dynamic vibration noise is filtered out.

Figure 2-39 shows results of augmented Lagrange explicit dynamic face-face contact (the dash explicit enforcement method). This method has the same character as the node-face method. Good agreement to the analytic static results is obtained in the regime where the problem is loaded statically and some expected high frequency dynamic based noise shows up in the high sliding rate period 11. Thus in period 11 the code should and do match the analytic static results in the average sense once the high frequency dynamic vibration noise is filtered out.

Figure 2-40 shows results of analytic rigid surface explicit contact (the ARS explicit enforcement method). As with static and implicit dynamic ARS contact this method shows good agreement to the analytic solution for small sliding. In the large sliding rate period 11 there are unexpected deviations from the analytic solution. The high frequency noise in this loading period is expected



(a) X Contact Force

(b) Y Contact Force

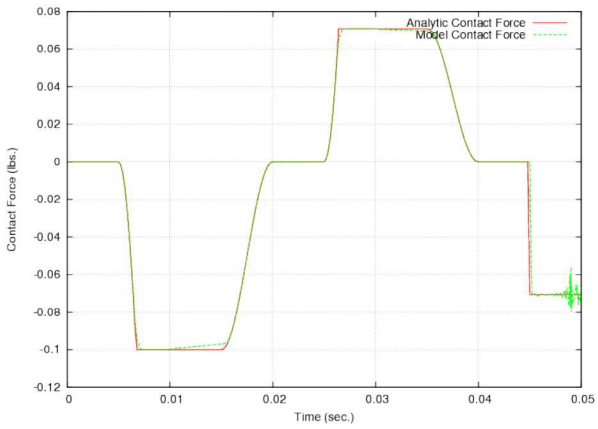
(c) Z Contact Force

(d) Z Displacement

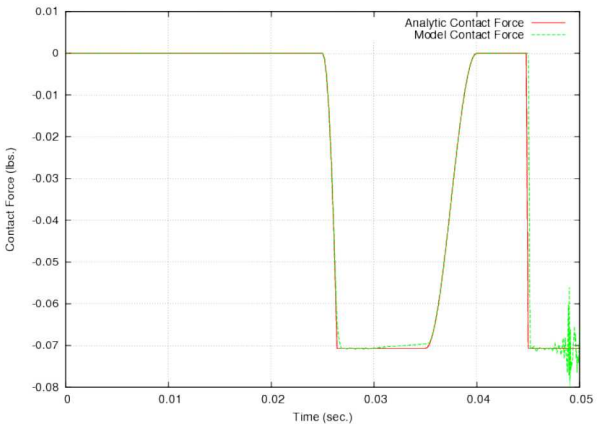
Figure 2-35. Implicit Dynamics Node-Face

and is seen in other explicit dynamics enforcement methods. However, unlike the node-face and face-face enforcement method the ARS contact method does not match the analytic solution in the average sense. Thus as with the quasistatic and implicit dynamics ARS enforcement methods, explicit ARS enforcement is only verified to give correct answers for small magnitudes and rates of sliding.

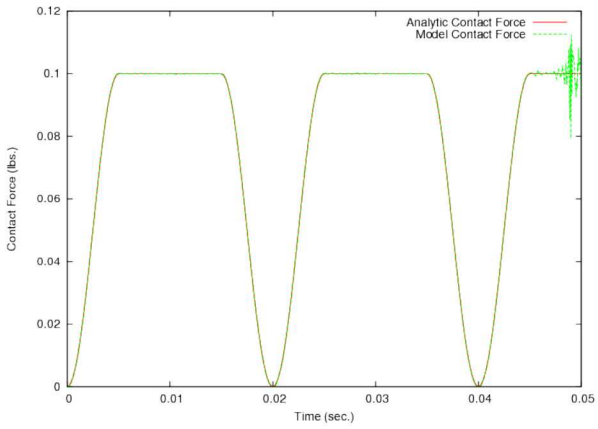
For input deck example see Appendix B.11.



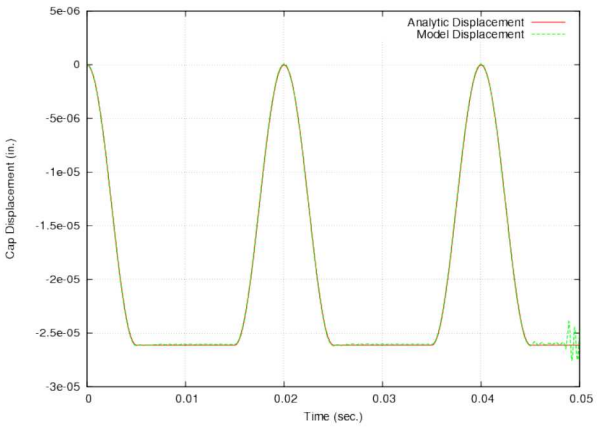
(a) X Contact Force



(b) Y Contact Force

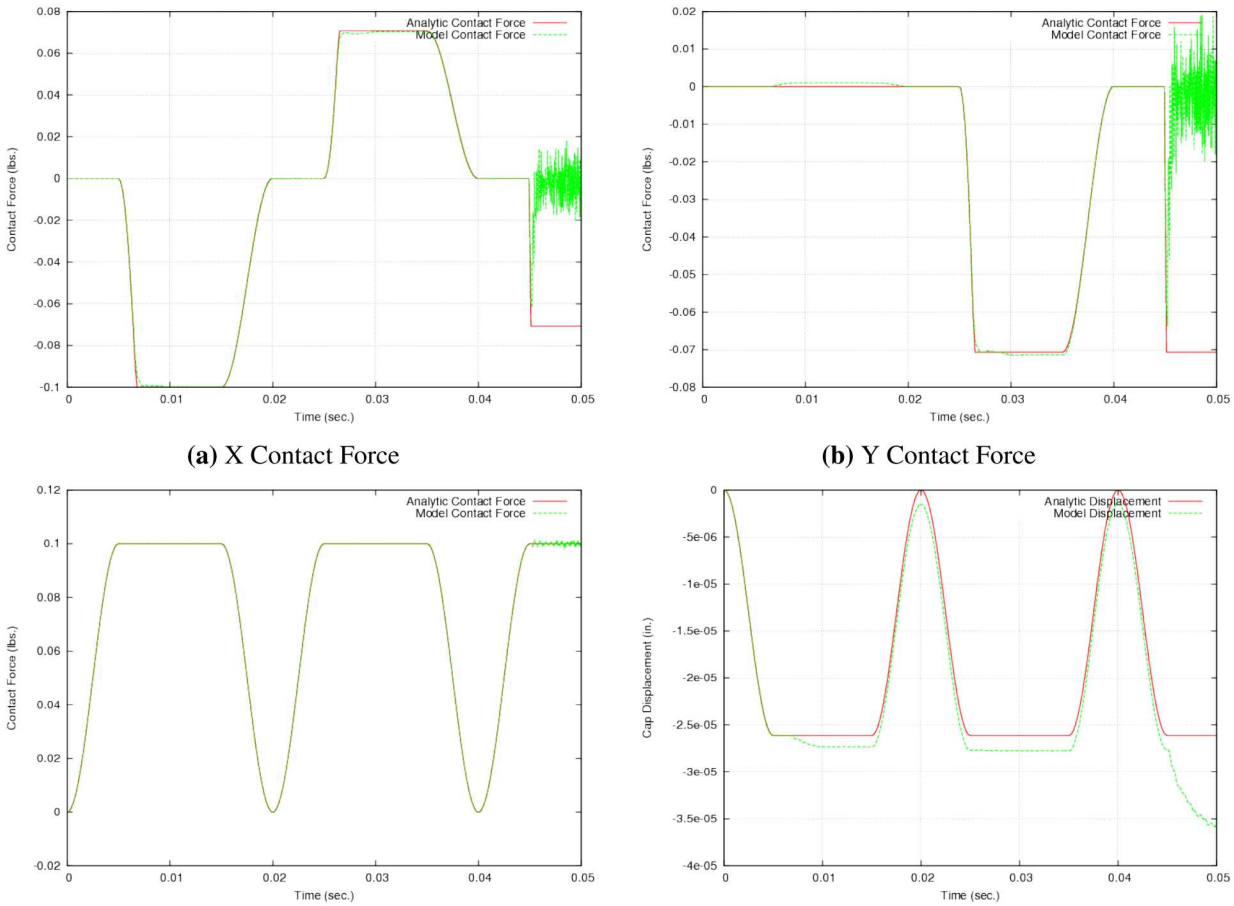


(c) Z Contact Force



(d) Z Displacement

Figure 2-36. Implicit Dynamics Face-Face



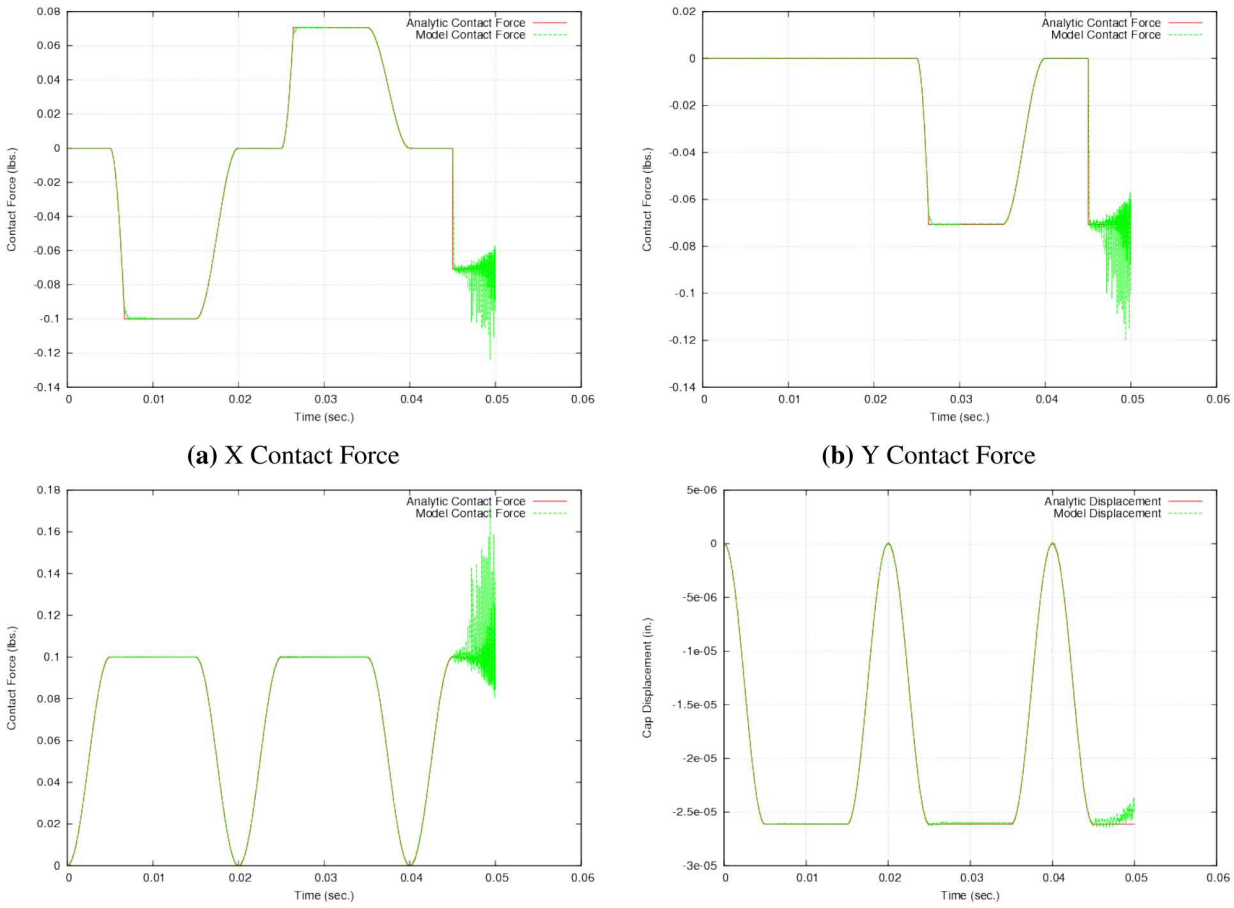
(a) X Contact Force

(b) Y Contact Force

(c) Z Contact Force

(d) Z Displacement

Figure 2-37. Implicit Dynamics ARS



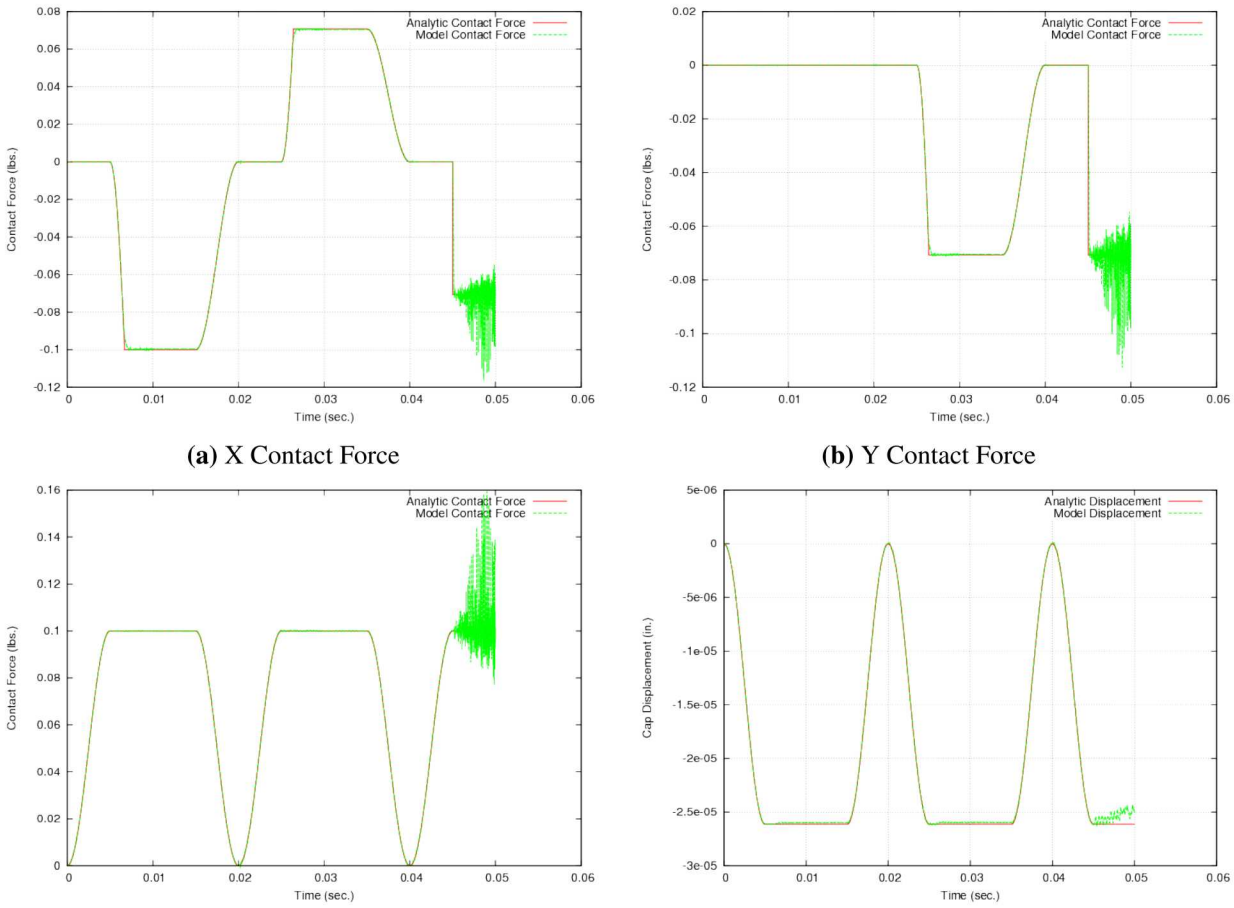
(a) X Contact Force

(b) Y Contact Force

(c) Z Contact Force

(d) Z Displacement

Figure 2-38. Explicit Dynamics Node-Face



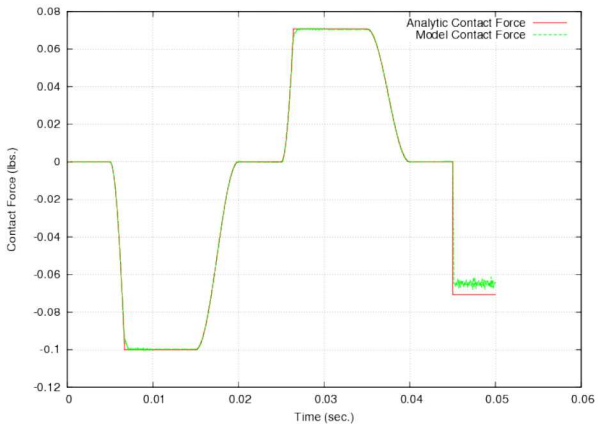
(a) X Contact Force

(b) Y Contact Force

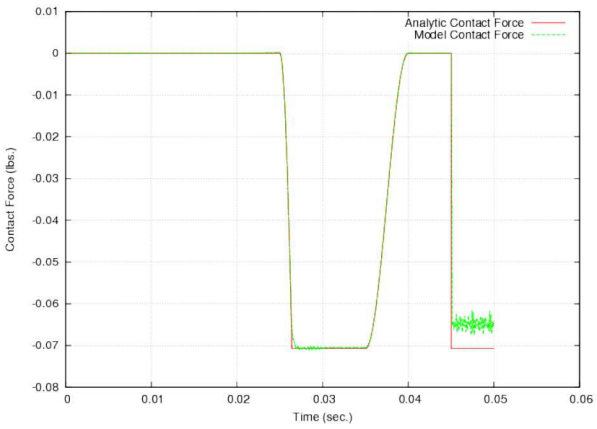
(c) Z Contact Force

(d) Z Displacement

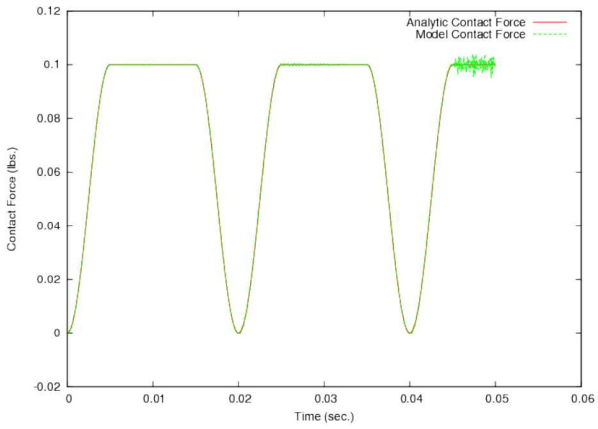
Figure 2-39. Explicit Dynamics Face-Face



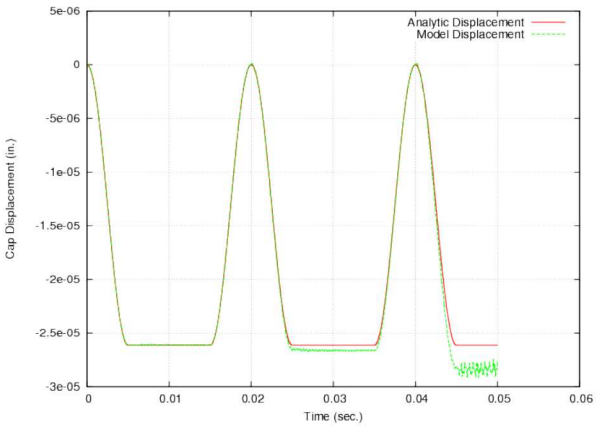
(a) X Contact Force



(b) Y Contact Force



(c) Z Contact Force



(d) Z Displacement

Figure 2-40. Explicit Dynamics ARS

2.12. OSCILLATING BLOCK SPRING WITH FRICTION

Analysis Type	Explicit Dynamics
Element Types	Hex8, Spring
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error Test
Verification Quantities	Displacement
Keywords	Coulomb Friction, Contact

2.12.1. Problem Description

This test checks the contact stick behavior in the context of a dynamic response. The analytical reference solution is based upon a single degree of freedom, dynamic system subjected to Coulomb damping. Specifically, consider a point mass m attached to a spring of stiffness k, resting on a surface with coefficient of friction μ, and acted upon vertically by gravity g, as depicted in Figure 2-41. The dynamic system is excited by prescribing an initial displacement of the mass toward the spring (compressing the spring). The mass is then released. If the spring force is greater than the frictional force, motion of the mass will ensue, and if not it will remain at rest with the spring deformed. If the spring has sufficient energy in the initial deformed state, the mass will not only move but will oscillate in a damped motion until friction stops it.

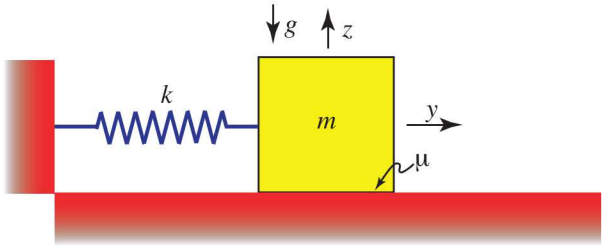


Figure 2-41. Analytical model: Single degree of freedom, dynamic system subjected to Coulomb friction.

The finite element model represents the mass and substrate as blocks of hexahedral elements, as depicted in Figure 2-42. The moving block of mass m is a cube with an edge dimension of 3 meters. The spring is depicted by the blue bar element attached at the center of the left face (top block) and has an initial length of 1.5 meters. The test is run as an explicit dynamic problem.

2.12.1.1. Boundary Conditions and Body Forces

A vertical body force exists on the top block due to a gravity load that is sinusoidally ramped (during the first 10 seconds) and then held constant. During this same initial loading time interval the block is displaced (via a sinusoidal ramping) to the left by 1 meter. In the second time interval

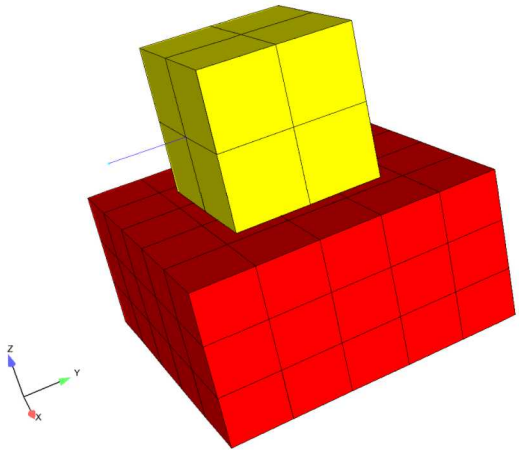


Figure 2-42. Mesh for block-spring dynamic system subjected to Coulomb friction.

the spring is released to either stick or move, depending upon the relative magnitudes of the friction and spring forces. The substrate is fixed at all nodes. The fixed end of the spring is fixed with respect to x and y translations.

2.12.1.2. Material Models

Each block uses an elastic material model where the values were picked for convenience, i.e., stiff enough to make the moving block act as a rigid body and compliant enough to not drive the time step too small for the explicit analysis.

Young's Modulus	E	$1.e+5$ Pa
Poisson's Ratio	ν	0.0
Density	ρ	0.148148 kg/m^3

Note that the density was set to give a mass of 4 kg for the moving block.

The spring stiffness was selected to be $16\pi^2$ N/m, so that the frequency of vibration would be 1 Hz. The linear force versus strain function (spring F vs ε) used as input for the spring element has force values of magnitude $24\pi^2$ N for strains of magnitude unity (1.5 times the stiffness since the initial length of the spring is 1.5 meters, and an engineering strain measure is used to define the strain in the spring).

2.12.1.3. Contact Interaction Model

The two blocks interact through a constant coefficient, Coulomb friction model. This model provides no resistance to surface separation (though none is induced here) and a maximum tangential contact force directly proportional to the normal contact force. The coefficient of friction is set in this problem to yield two cycles of oscillation for the block before it sticks due to friction.

Coefficient of Friction	μ	0.4471448

2.12.1.4. Feature Tested

This test examined the frictional contact force calculations, in the context of an initially dynamic response. It examines both sliding and stick conditions, but the emphasis of the test was upon the stick response.

2.12.2. Assumptions and notes

The FEM model used in this test is assumed to provide a sufficiently accurate representation of a single degree of freedom dynamic system. As such, the stiffness of the moving block must be high enough to not introduce any significant deformation, e.g., where the spring connects to the block or stress waves traveling through the block. Furthermore, tipping of the block due to the spring load acting above the plane of the frictional surface is assumed to have an insignificant effect upon the response.

2.12.3. Verification of Solution

The frictional force obeys the inequality

$$F_{\text{tang}} \leq \mu \, F_{\text{norm}}, \tag{2.18}$$

where F_{tang} is the tangential contact force, μ is the coefficient of friction, and F_{norm} is the normal contact force. When F_{tang} is less than μF_{norm}, motion is prevented by the frictional force, i.e., the stick condition. During motion of the block the above inequality is satisfied as an equality. This frictional behavior introduces a nonlinearity in the governing equations. Fortunately however, it is linear for each half-cycle and thus is amenable to analytical solution. Unlike viscous damping the friction force is not proportional to the velocity, but it does oppose the motion and thus acts in the opposite direction of the velocity. As such, for a given velocity direction, the equations of motion are those of an undamped system with a constant force opposing the motion, as given below.

$$\begin{aligned} \ddot{u} + \omega_n^2 u &= \quad \omega_n^2 U_D \quad \text{if } \dot{u} < 0 \\ \ddot{u} + \omega_n^2 u &= -\omega_n^2 U_D \quad \text{if } \dot{u} > 0 \end{aligned} \tag{2.19}$$

where $U_D = \mu g / {\omega_n}^2$.

The solution approach is to solve the linear ordinary differential equations for each half-cycle, for which the velocity direction remains constant. Each half-cycle solution inherits its initial conditions (prescribed displacement and zero velocity) from the end of the previous half-cycle solution. At the end of each half-cycle the frictional stick condition is evaluated to determine if the spring has sufficient force to continue the motion (*i.e.*, overcome friction). Thus the final solution is a sequence of solutions for each half-cycle - initial value problem solutions stitched together to give the response over time.

The solution to the equations of motion for each half-cycle case are given by

$$u(t) = \begin{cases} u_{t_0}\cos(\omega_n(t-t_0)) + U_D[1-\cos(\omega_n(t-t_0))] & \text{if } \dot{u} < 0 \\ u_{t_0}\cos(\omega_n(t-t_0)) - U_D[1-\cos(\omega_n(t-t_0))] & \text{if } \dot{u} > 0 \end{cases}$$

Each half-cycle ends when the velocity reduces to zero, which occurs on intervals of 1/2 seconds. The positions at the start of each half cycle are given by the sequence {-1, 1-2U_D, -1+4U_D, 1-6U_D, -1+8U_D} m. These are the displacement initial conditions for the corresponding half-cycle solutions. The solution for the $i^{textrmth}$ half cycle is then given by

$$u(t) = \begin{cases} (1-(4i-2)U_D)\cos(\omega_n(t-t_0)) + U_D[1-\cos(\omega_n(t-t_0))] & \text{if } \dot{u} < 0 \\ (-1+(4i-4)U_D)\cos(\omega_n(t-t_0)) - U_D[1-\cos(\omega_n(t-t_0))] & \text{if } \dot{u} > 0 \end{cases}$$

In the following figures the displacement history is presented for both the analytical and FEM solution. Note that over the three seconds depicted in Figure 2-43 the FEM solution shows good agreement with the analytical solution, especially during the oscillations. The deviation between the solutions once the block sticks, though slight, reveals a numerical artifact that is not physical and is related to the algorithm used for stick enforcement. Figure 2-44 examines, in more detail, the displacement response near when sticking occurs. The problem was designed to allow the block to oscillate for two cycles, so that there would be a sufficient number of steps for the solutions to differ when the stick condition occurred; this effect is apparent at a time of about 12 seconds.

The code solution does not stick at the precisely predicted analytic location. It is unknown at this time what the exact cause of the discrepancy is, may be related to the computational model have multiple degrees of freedom to capture rocking and vibration modes that are not relevant to the single degree of freedom analytic solution. An additional important feature of the response is the continued frictional creep behavior that is not consistent with the exact solution. This frictional creep is numerical artifact that causes the block to continue to slide at a very small velocity once the stick condition is reached.

The verification check on this problem requires the relative error in the displacement to be within the interval [0.009, 0.0101], i.e., approximately 1%. Specifically a one-norm in time is used for the evaluation.

An additional check is made on the frictional energy at the problem end time. The expected frictional energy should equal the frictional force times the displacement through which the block moves. The frictional force is constant at block mass times gravity times friction coefficient. The expected displacement is found by integral of the analytic displacement history function as shown in Figure 2-43.

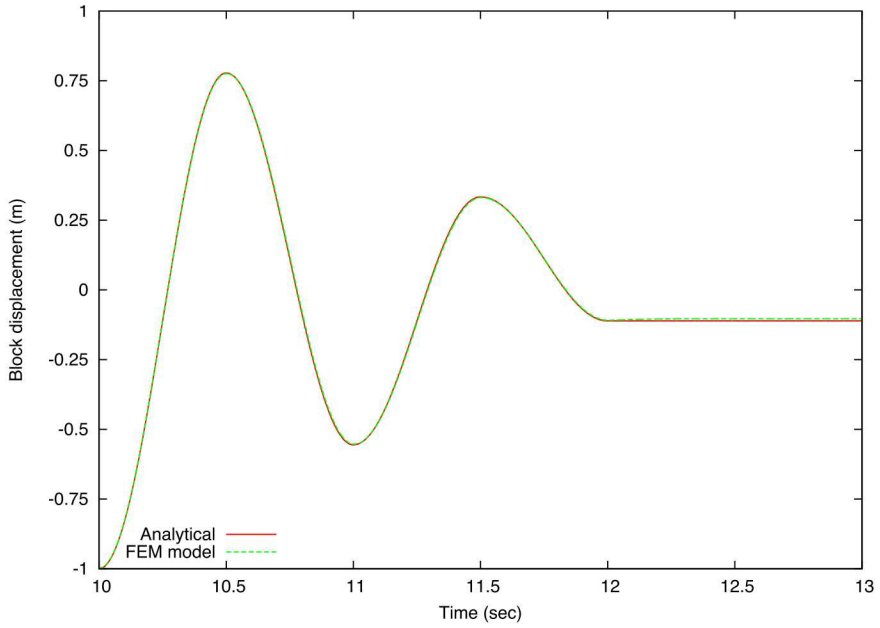


Figure 2-43. Analytical vs. FEM solution comparison for block displacement history.

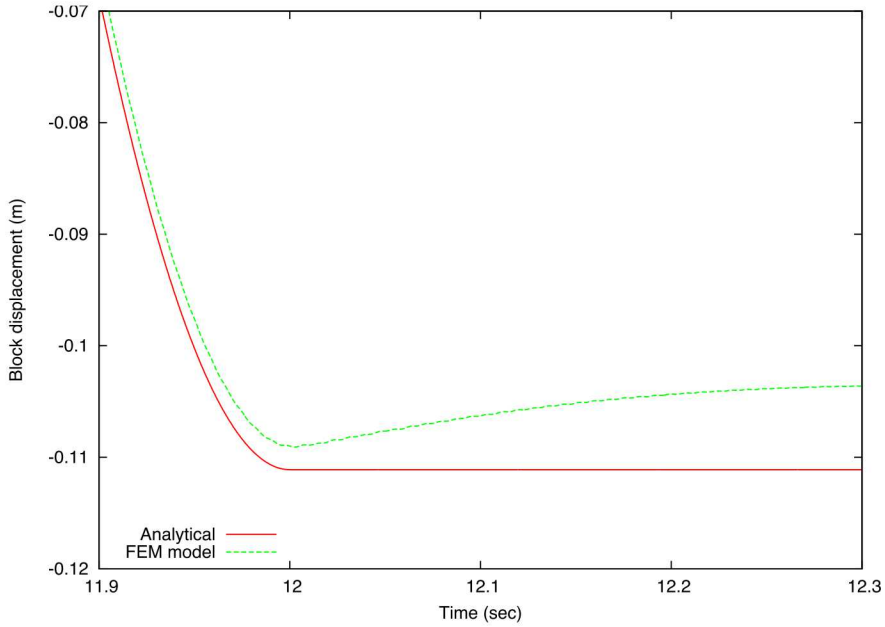


Figure 2-44. Analytical vs. FEM solution comparison for block displacement history – zoomed to show stick response.

2.12.4. References

1. Craig RR. **Structural Dynamics, An Introduction to Computer Methods.** John Wiley & Sons, 1981, pp 65-66.

For input deck example see Appendix B.12.

2.13. FRICTION WEDGE

Analysis Type	Explicit Dynamics
Element Types	Hex8, Rigid Body
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error and Convergence
Verification Quantities	Contact Stick/Slip
Keywords	Coulomb Friction, Contact slip

2.13.1. Problem Description

This test checks the contact stick behavior in the context of an essentially quasistatic response. The expected solution is simply that no slip is expected to occur in the problem due to Coulomb friction. The geometry and mesh for the problem are depicted in Figure 2-45. Boundary conditions for the tests are such that the top and bottom wedges squeeze the middle wedge, like two fingers squeezing a water melon seed, and consistent with this analogy under the right conditions the middle wedge can be dynamically expelled.

For the problem specification examined in this test, the middle block will only exhibit "contact creep" response, not dynamic motion. (Contact creep, which may also be referred to as frictional creep, is a slow slip response that occurs for a body in frictional contact and subjected to a tangential load; this response is an artifact of the contact algorithm and is not physical in nature.) The slope on the wedge faces is 0.2. A such, a simple examination of the statics for this problem, shows that in resolving the vertical forces transferred from the top and bottom wedges to the middle wedge, the ratio of tangential to normal force is 0.2. Thus we expect any coefficient of friction greater than 0.2 to hold the block in place. We refer to this value as the critical coefficient of friction. Other tests using this geometric configuration have been previously studied with Sierra/SM but with a different emphasis. A performance test having the same geometry uses a finer mesh than used here. Our focus for this test is upon examining the stick enforcement of the contact algorithm.

In the finite element model, the wedges are modeled with hexahedral elements, but the top and bottom wedges are prescribed to have rigid body motion, as such only the middle wedge will deform elastically. The bounding cube on the stack of wedges has edges with a length of 1 inch. The top and bottom wedges have a thickness that increases from 0.2 inches to 0.4 inches.

2.13.1.1. Boundary Conditions

The top and bottom wedges are prescribed to displace toward the middle wedge via velocity boundary conditions. The magnitude of these velocities (in the -y and +y directions, respectively) is depicted in Figure 2-46. Note that there are four stages in the wedge loading: (1) positive acceleration (time$\in$(0,0.001)), (2) constant velocity (time$\in$[0.001,0.002]) (3) negative acceleration

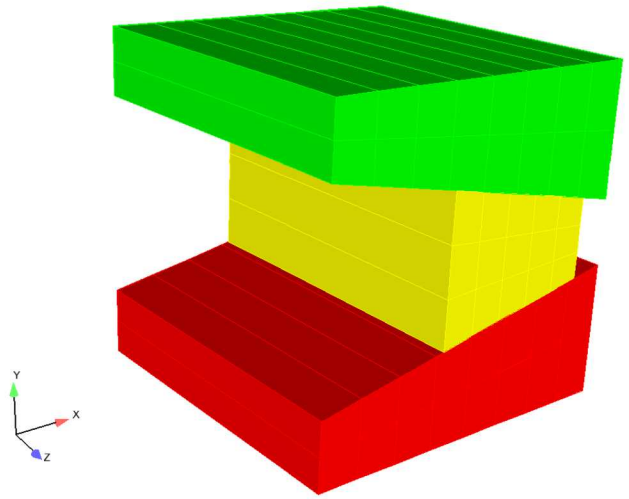


Figure 2-45. Mesh for the stack of wedges subjected to Coulomb friction.

(time∈(0.002,0.003), and (4) constant position (time∈(0.003,∞)). The corresponding displacement for this input is shown in Figure 2-47. The top and bottom wedges are restrained against rigid body translation in the xz-plane and against all rigid body rotations. The middle wedge is restrained against displacement in the z-direction.

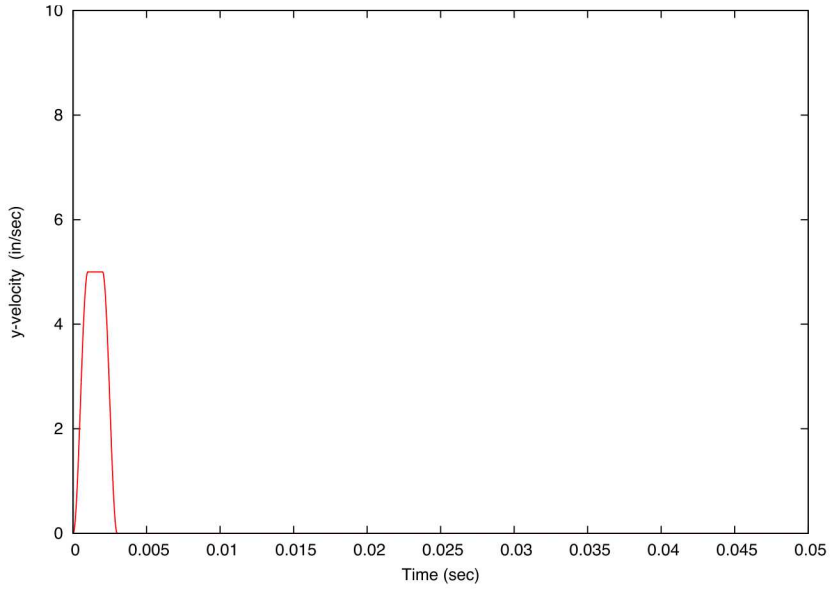


Figure 2-46. History of top and bottom wedges' |velocity|.

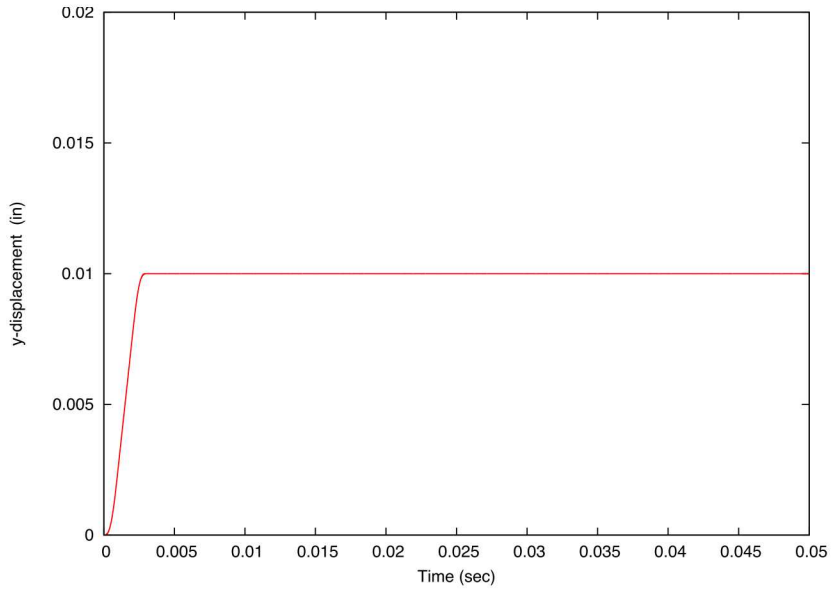


Figure 2-47. History of top and bottom wedges' |displacement|.

2.13.1.2. Material Models

The middle wedge is modeled as elastic with material model parameters as given in the table below. The stiffness and density of the material should have no effect on the ultimate stick/slip behavior of the wedge. However, the wedge must be stiff enough such that the dynamic effects due to the transient loading are relatively small.

Young's Modulus	E	$1e4$ psi
Poisson's Ratio	ν	0.0
Density	ρ	$7.4e-4$ lbf sec^2/in^4

2.13.1.3. Contact Interaction Model

The middle wedge interacts with the top and bottom wedges through a constant coefficient, Coulomb friction model. This model provides no resistance to surface separation and a maximum tangential contact force directly proportional to the normal contact force. The coefficient of friction is set in this problem to be just above the critical coefficient of friction (0.2), though parameter studies with larger values (e.g., 0.3) yielded qualitatively similar results in terms of the contact creep behavior.

Coefficient of Friction	μ	0.201

2.13.1.4. Feature Tested

This test examined the frictional contact force calculations, in the context of quasi-static response. The emphasis of the test was upon the stick response, and this version of the code exhibited an erroneous contact creep behavior where slip slowly occurs even when the friction law should require no slip. A parameter study, on the effect of the number of contact momentum balance iterations, will show contact creep can be significantly reduced by increasing the number of iterations.

2.13.2. Assumptions and notes

The loading is assumed to be sufficiently slow to not induce significant wave propagation within the middle wedge; dynamic response of this type could affect the normal tractions and thus the evaluation of the stick condition. While there is a variation in the normal traction along the wedge, due to the compliance variation that occurs with the changing height. During the initial loading small lateral (x) deformation in the wedge should occur until equilibrium is reached where the stick/slip condition is exactly reached everywhere on the contact surface. At this point the ratio of tangential to normal tractions is governed by the geometry. To verify that qualitatively the response was not sensitive to this assumption, we examined cases with coefficients of friction as large as 0.3.

2.13.3. Verification of Solution

The frictional force obeys the inequality

$$F_{\text{tang}} \leq \mu F_{\text{norm}}, \tag{2.20}$$

where F_{tang} is the tangential contact force, μ is the coefficient of friction, and F_{norm} is the normal contact force. When F_{tang} is less than μF_{norm}, motion is prevented by the frictional force, i.e. the stick condition. During motion of the block the above inequality is satisfied as an equality.

For the following results, the effect of the number of contact iterations for momentum balance is examined, since it is expected to affect the results for problems of this type. Figure 2-48 shows the history of the slip at the leading edge (bottom left corner of the front face, node 55) of the wedge. Clearly increasing the number of contact iterations results in a better enforcement of the stick condition, but the contact creep may still occur. Note that significant changes (albeit on a log scale) in the slip can occur in any stage of the loading (depending upon the number of contact iterations). Even in the last stage where the edge blocks are not moving slip continues to occur a significant rate if relatively few contact iterations are used.

Figure 2-49 depicts the average slip response over both contact surfaces and thus is inherently smoother. Examining the slip behavior alone can lead to a false positive interpretation of the response, because the slip can remain constant when either perfect stick or separation occurs. One can distinguish between these cases by examining the motion of the middle wedge; Figure 2-50

depicts the magnitude of the average nodal displacement of this wedge, and clearly reflects the contact creep motion. This displacement response is averaged over the whole body and is less sensitive to the number of contact iterations.

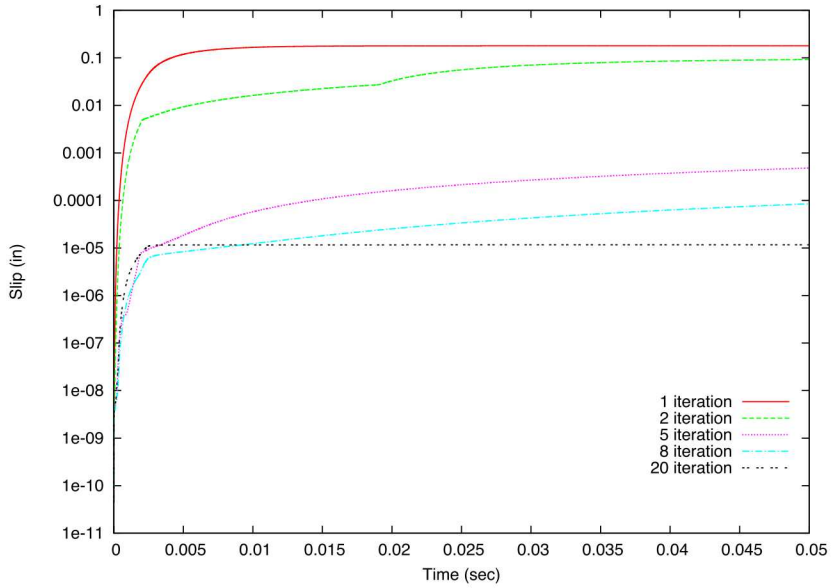


Figure 2-48. Slip history for bottom left corner, front face, of the middle wedge.

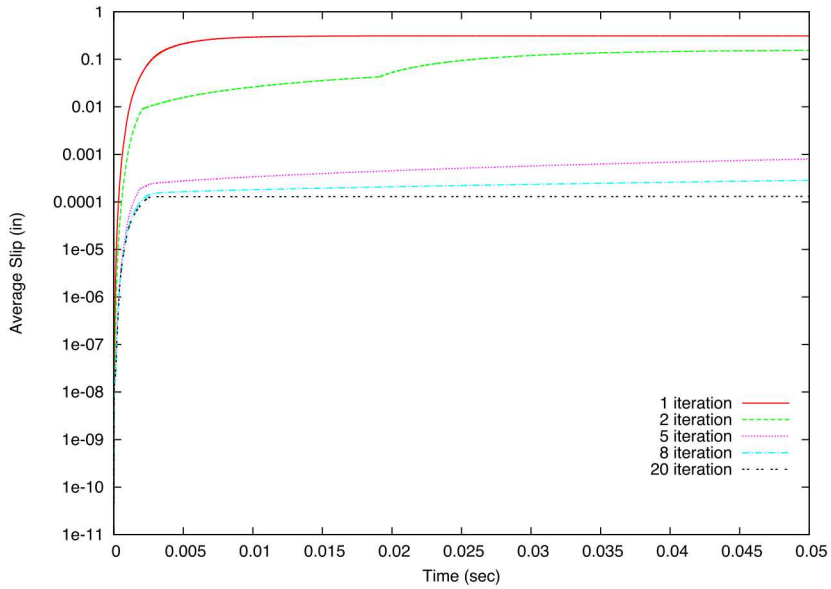


Figure 2-49. Average slip history of the middle wedge.

For any given time, Figure 2-49 indicates that average slip decreases with an increase in the number of contact iterations. Figure 2-51 plots the average slip, i.e., average error in the stick enforcement, as a function of the number of contact iterations.

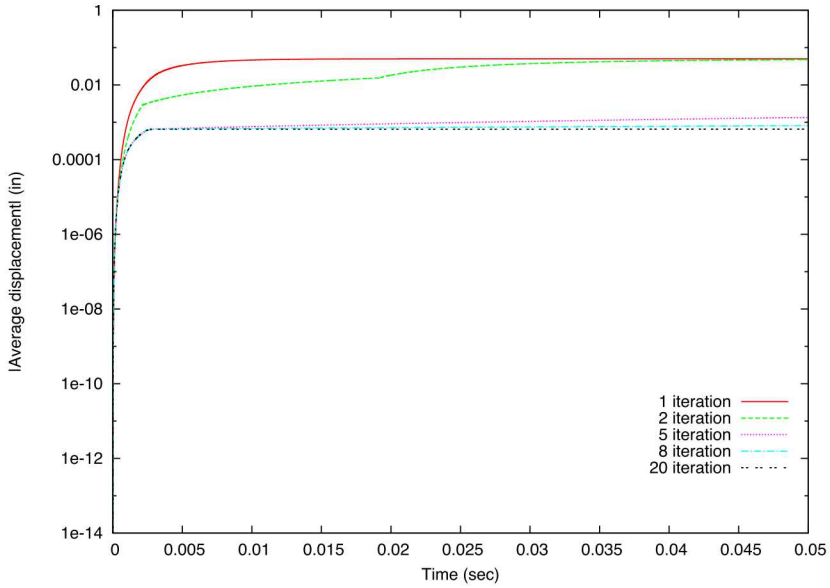


Figure 2-50. Average displacement history of the middle wedge.

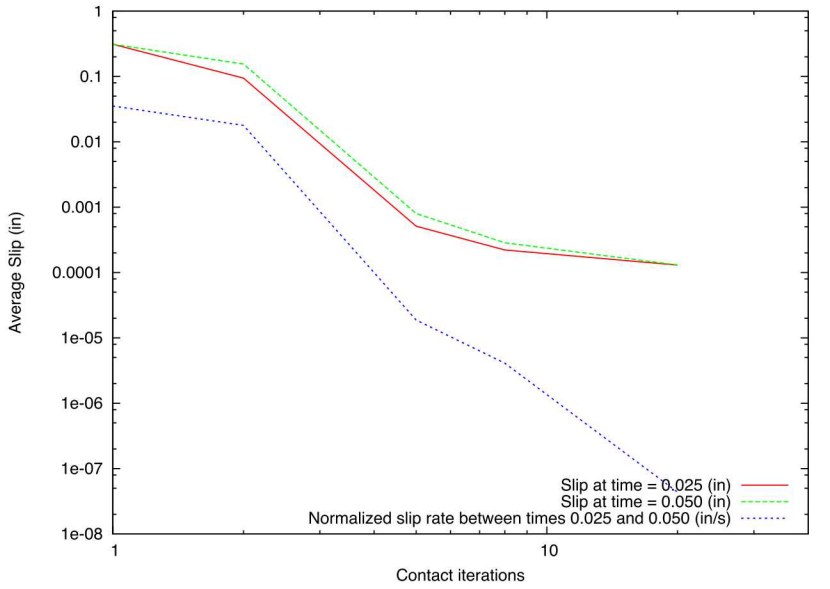


Figure 2-51. Average slip versus number of contact iterations, at two values of time.

Figure 2-51 shows the average slip that occurs late in time. Included is the accumulated slip half way through the analysis, at the end of the analysis, and a normalized slip rate between these two times. It is assumed that any dynamic or load driven displacement have occurred by the half way point of this analysis. Any additional slip past this point is likely a numerical error. The normalized slip rate plotted is given by Equation 2.21. The slip rate is normalized by the current value of contact normal force on the wedge. If the wedge slips then the compressive forces are

lessened. This reduces the expelling force on the wedge and thus will reduce slip rate, the normalization will eliminate this effect. When run with a small number of contact iterations the wedge may slip a significant distance lowering the compressive stress on the wedge by a factor of two or more. For relatively large number of contact iterations the wedge stays well stuck and the compressive stress remains constant between the different iteration cases.

$$rate = ((slip_{\text{end}} - slip_{\text{half}})/(time_{\text{end}} - time_{\text{half}}))/(\frac{1}{2}(force_{\text{end}} + force_{\text{half}})) \tag{2.21}$$

It is assumed that this long duration slip rate should ideally be as close to zero as possible. At a large number of iterations the slip rate starts to approach the numerically obtainable zero value of 1.0e-12. Certainly internal waves within the wedge could reduce the normal force point-wise on the interface and could contribute to contact creep. For the case presented above, the coefficient of friction is close to the critical value (within 0.5 percent), which makes the contribution of this possible effect more suspect.

The nightly verification for this test confirms that the slip rates given by Equation 2.21 match within 5% of those in Figure 2-51. For input deck example see Appendix B.13.

3. ELEMENT VERIFICATION TESTS

The following are tests that verify different element types and element formulations. This includes tests that elements have the correct response singly or as groups of elements.

3.1. HEX PATCH TESTS – QUASI-STATIC, LINEAR ELASTIC

Analysis Type	Quasi-statics
Element Types	Hex8, Hex20, Hex27
Element Formulations	Mean Quadrature, Q1P0, Selective Deviatoric, Fully Integrated
Strain Incrementations	Midpoint Increment and Strongly Objective
Material Models	Elastic, Neo-Hookean
Verification Category	Discretization Error
Verification Quantities	Stress Components
Number of Tests	14
Keywords	Patch Test, Linear Elastic

3.1.1. Brief Description

This problem is a patch test for hexahedral elements. A cubic domain is subjected to prescribed displacements on each surface. The magnitude of the displacements is defined to be sufficiently small that linear elasticity provides a reasonable approximation of the expected response. Fourteen test results are obtained for a combination of seven element formulations using two different elastic material models (Elastic and neo-Hookean).

3.1.1.1. Functionality Tested

Primary capabilities:

- The following element formulations:
  - eight-node hexahedron with the mean quadrature formulation and midpoint strain incrementation

- eight-node hexahedron with the mean quadrature formulation and strongly objective strain incrementation
- eight-node hexahedron with the Q1P0 formulation and strongly objective strain incrementation
- eight-node hexahedron with the selective deviatoric formulation and strongly objective strain incrementation
- eight-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation
- twenty-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation
- twenty-seven-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation

Secondary capabilities:

- prescribed displacement direction boundary conditions via analytic expressions
- resolution of kinematic boundary condition
- elastic and neo-Hookean material models in the small strain regime

3.1.1.2. Mechanics of Test

A unit cube domain is positioned such that diagonally-opposite vertices are at the origin and (1,1,1) with the faces aligned with global coordinate planes. The mesh consists of seven hexahedral elements, each of which is in a separate element block. Six of the elements have one face on the exterior, and one element has all interior faces. To provide a completely general test, the interior element has no parallel or perpendicular edges. The interior element (element block 1) is shown in Figure 3-1. None of the faces of the interior element are perpendicular or parallel to the planes xy, yz, zx defined by the x-, y-, and z-axes.

Figure 3-2 is drawn including all of the elements except the element defining element block 3, which has an exterior face with a normal in the positive z-direction. The interior element and four surrounding elements are visible. The element with an exterior face with a normal in the negative z-direction is not visible in this hidden-line drawing of the elements.

The element geometries for the hex20 and hex27 meshes are nominally the same. The hex20 mesh only differs with the hex8 mesh by adding mid-edge nodes. The hex27 mesh further differs with the hex8 by also adding mid-face nodes and a mid-element node.

The prescribed displacement field on the surface of the cube is given by:

$$u = t \times (1.0 \times 10^{-4}) \times (2x + y + z) \tag{3.1}$$

$$v = t \times (1.0 \times 10^{-4}) \times (x + 2y + z) \tag{3.2}$$

$$w = t \times (1.0 \times 10^{-4}) \times (x + y + 2z) \tag{3.3}$$

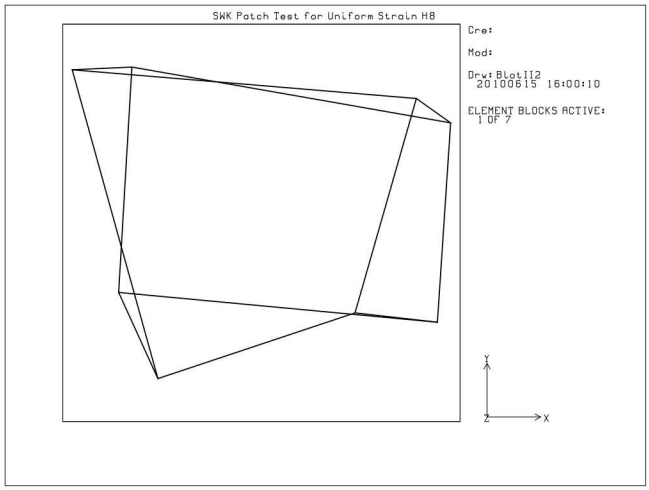


Figure 3-1. Interior element for patch-test cube

where t denotes time, u denotes the displacement in the x-direction, v denotes the displacement in the y-direction, and w denotes the displacement in the z-direction. The units for the displacement components are inches. The corresponding analytic functions in the input file are also labeled as u, v and w, respectively.

3.1.1.3. Material Model

The material models are elastic with the properties given below.

Young's Modulus	E	1.0×10^6 psi
Poisson's Ratio	ν	0.25

3.1.2. Expected Results

These tests assume that the displacements and strains will be sufficiently small for linear elasticity to provide a reasonable reference solution. For infinitesimal strains, the strain-displacement relations of linear elasticity give the strains at t=1 as:

$$\epsilon_{xx} = \epsilon_{yy} = \epsilon_{zz} = 2 \times 10^{-4} \tag{3.4}$$

and

$$\epsilon_{xy} = \epsilon_{yz} = \epsilon_{zx} = 1 \times 10^{-4} \tag{3.5}$$

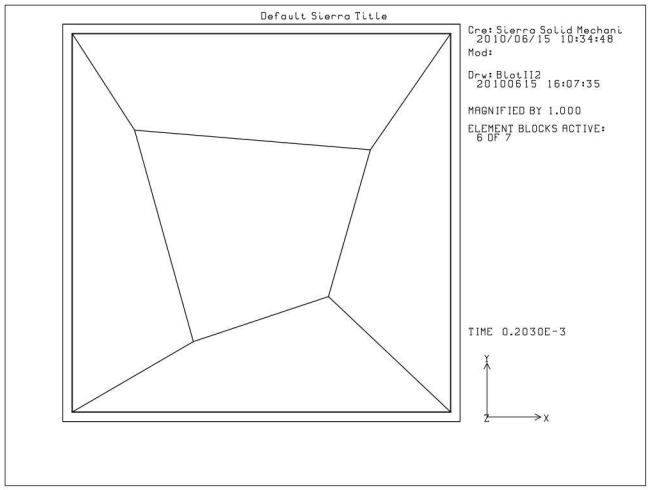


Figure 3-2. Patch-test cube with element block 3 not shown

Note that the size of sufficiently small depends upon the particular material model. Both the elastic model (a hypoelastic model) and the neo-Hookean model (a hyperelastic model) are used for these patch tests. For infinitesimal strains, responses from both constitutive models reduce to that of a linear elastic model, where the stress σ_{xx} is related to the strains ϵ_{xx}, ϵ_{yy}, and ϵ_{zz} by:

$$\sigma_{xx} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\left[\epsilon_{xx} + \frac{\nu}{(1-\nu)}(\epsilon_{yy}+\epsilon_{zz})\right] \tag{3.6}$$

Similar equations hold for σ_{yy} and σ_{zz}. The shear stress τ_{xy} is related to the shear strain ϵ_{xy} by:

$$\tau_{xy} = \frac{E}{(1+\nu)}\epsilon_{xy} \tag{3.7}$$

Similar equations hold for τ_{yz} and τ_{zx}.

The stress field produced by the above strain field at t=1 is given by:

$$\sigma_{xx} = \sigma_{yy} = \sigma_{zz} = 400\,psi \tag{3.8}$$

and

$$\tau_{xy} = \tau_{yz} = \tau_{zx} = 80\,psi \tag{3.9}$$

For all the hexahedral element patch tests, the error in each normal and shear stress component was examined, where the error was defined as the infinite norm (maximum over all elements) of

the differences between the linear elastic reference solution and the calculated results. The errors are examined for t=1 using 2 equal time steps. Based upon results for the hex8 element, using a few different numbers of time steps, the results did not appear to change with the number of time steps ranging from 1 to 10.

For both the hypoelastic (elastic) and hyperelastic (neo-Hookean) material models, the solution verification requirement was that each element have errors in each stress component of less than 0.1 percent – an indistinguishable difference on a plot. The results for the two different elastic models differed in the fourth digit of the results. All of the elements passed the patch test except the hex8 with the Q1P0 formulation. For all of the other element types, they failed the test at an error tolerance that was one order of magnitude smaller (0.01 percent). As such, the results only reproduced the linear elastic solution to three digits. Many authors have noted that the patch test results should have an accuracy approaching the precision of the floating point numbers on the particular computer (see, e.g., [2]). For the results presented here, the differences with reference solution are not a result of inaccurate computations but rather are a result of limited accuracy in the linear elastic reference solution. A better measure of the accuracy of the computations is provided by the finite deformation versions of these tests (in development), where the reference solutions are based upon the same finite deformation relationships as the code. To test the assertion that the linear elastic reference solution is the issue, we reduced the displacements by one order of magnitude. The resulting computational results agreed with the linear elastic reference solution in one additional digit. We also symbolically solved the finite deformation equations for the neo-Hookean model and obtained the same results as Sierra to the 6 digits displayed in the Ensight post-processor (i.e., the finite deformation solution differs with the linear elastic solution in the fourth digit too).

The Q1P0 element is the exception to the above general conclusions for the elements. The errors in the normal stress components were about 12 percent, and the errors in the shear stresses were about 1.4 percent. Note that the displacement-prescribed patch test is usually described as a means of showing that the element satisfies the polynomial completeness condition. Linear completeness combined with stability has been demonstrated to provide convergent element formulations (though apparently without a general proof as available for the Lax Equivalence theorem for finite difference approximations). Researchers have argued that the patch test is in general neither a necessary nor sufficient condition for convergence, and Hughes [3] mentions some of the earlier mathematical controversy in his text. It is currently unclear if failure of the patch test for the Q1P0 element reflects an implementation error, or if it is an example of the patch test not being a necessary condition for convergence. We do know that for two convergence tests (currently in adagio_rtest/verification/elements/hex8_Q1P0) this element formulation showed convergence properties similar to the other hex8 element formulations, yielding optimal convergence rates, but in those cases all elements were cubes – not distorted like the patch test. If the implementation of the element it valid, it does imply at the very least that mesh refinement can be necessary to represent even a uniform stress field. Additional work is needed to resolve the uncertainty. For additional information the SM implementation of this element, and its initial application at Sandia see references 4 and 5.

3.1.3. References

1. Fung, Y.C. *Foundations of Solid Mechanics*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1965.
2. Belytschko, T., Liu, W.K., and Moran B. *Nonlinear Finite Elements for Continua and Structures*, NY, NY: John Wiley & Sons, LTD, 2000.
3. Hughes, T.J.R. *The Finite Element Method, Linear Static and Dynamics Finite Element Analysis*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.
4. Kostka, T.D. "Clustered Void Growth in Ductile Metals: Final LDRD Report," SAND Report SAND2012-7892, Sandia National Laboratories, September 2012.
5. Kostka, T.D. "Development of a Material Model for Aluminum Alloys at High Temperature," SAND Report SAND2012-8058, Sandia National Laboratories, September 2012.

For input deck see Appendix B.14.

3.2. HEX PATCH TESTS – QUASI-STATIC, FINITE DEFORMATION

Analysis Type	Quasi-static (Adagio)
Element Types	Hex8, Hex20, and Hex27
Element Formulations	Mean Quadrature, Q1P0, Selective Deviatoric, Fully-Integrated
Strain Incrementations	Midpoint Increment and Strongly Objective
Material Models	Elastic, Neo-Hookean
Verification Category	Discretization Error
Verification Quantities	Spatial (left) stretch tensor and Cauchy stress tensor components
Number of Tests	28
Keywords	Patch test, finite deformation

3.2.1. Brief Description

This problem is a finite deformation patch test for hexahedral elements. A linear elastic version of this test (i.e., infinitesimal deformation version) is also in the test suite and manual. Unlike those tests, this test group uses an exact reference solution, since the continuum equations for finite deformations are solved for two elastic material cases, each having different kinematic descriptions. This patch test consists of a cubic domain subjected to prescribed displacements on each surface. The magnitude of the displacements is defined for two cases, one that has strains that are $O(1\%)$ and the other to give strains that are $O(100\%)$. Twenty-eight test results are obtained for a combination of seven element formulations, at two strain levels, using two different elastic material models (Elastic and neo-Hookean).

3.2.1.1. Functionality Tested

Primary capabilities:

The following element formulations:

- eight-node hexahedron with the mean quadrature formulation and midpoint strain incrementation
- eight-node hexahedron with the mean quadrature formulation and strongly objective strain incrementation
- eight-node hexahedron with the Q1P0 formulation and strongly objective strain incrementation
- eight-node hexahedron with the selective deviatoric formulation and strongly objective strain incrementation
- eight-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation

- twenty-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation
- twenty-seven-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation

Secondary capabilities:

- prescribed displacement direction boundary conditions via analytic expressions
- resolution of kinematic boundary condition
- elastic and neo-Hookean material models in the finite strain regime

3.2.1.2. Mechanics of Test

A unit cube domain is positioned such that diagonally-opposite vertices are at the origin and (1,1,1) with the faces aligned with global coordinate planes. The mesh consists of seven hexahedral elements, each of which is in a separate element block. Six of the elements have one face on the exterior, and one element has all interior faces. To provide a completely general test, the interior element has no parallel or perpendicular edges. The interior element (element block 1) is shown in Figure 3-3. None of the faces of the interior element are perpendicular or parallel to the planes *xy*, *yz*, *zx* defined by the *x*-, *y*-, and *z*-axes.

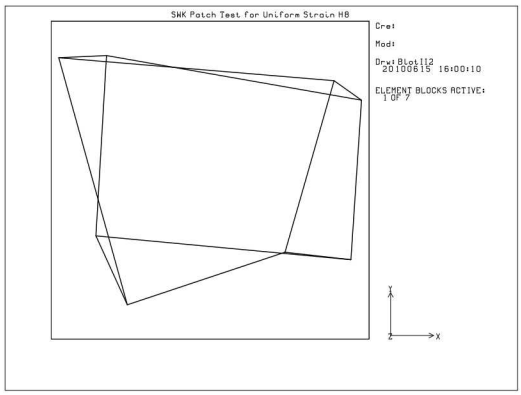


Figure 3-3. Interior element for patch-test cube

Figure 3-4 is drawn including all of the elements except the element defining element block 3, which has an exterior face with a normal in the positive *z*-direction. The interior element and four surrounding elements are visible. The element with an exterior face with a normal in the negative *z*-direction is not visible in this hidden-line drawing of the elements.

The element geometries for the hex20 and hex27 meshes are nominally the same. The hex20

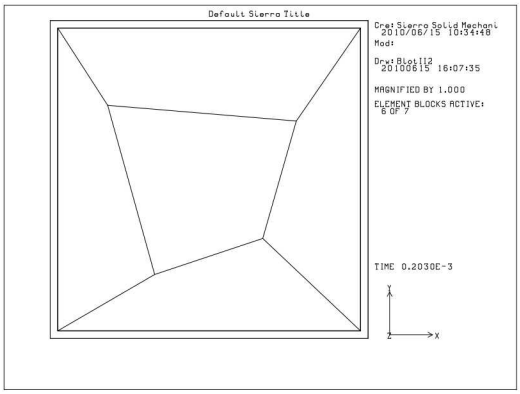


Figure 3-4. Patch-test cube with element block 3 not shown

mesh only differs with the hex8 mesh by adding mid-edge nodes. The hex27 mesh further differs with the hex8 by also adding mid-face nodes and a mid-element node.

The prescribed displacement field on the surface of the cube is given by:

$$u = c(2x + y + z)t \tag{3.10}$$

$$v = c(x + 2y + z)t \tag{3.11}$$

$$w = c(x + y + 2z)t \tag{3.12}$$

where c denotes a factor prescribing the displacement magnitude (units $1/time$), t denotes time, u denotes the displacement in the x-direction, v denotes the displacement in the y-direction, and w denotes the displacement in the z-direction. The units for the displacement components are inches. The factor c has the values of 0.005 and 1 for the $O(1\%)$ and $O(100\%)$ strain load cases, respectively. The normal logarithmic strain components are 1% and 100% to an accuracy of about 2 digits, but the corresponding maximum principal strains are about 2% and 161%. Time is increased from 0 to 1. For simplicity, because t and c will often occur as a multiplicative pair we will define $t_c = tc$ – a dimensionless, scaled time. The corresponding analytic functions for the displacement components in the input file are also labeled as u, v, and w.

The time step sizes for the set of analyses differed over a large range. For most of the element formulations, which adopt a strongly objective strain incrementation, the time step size is set to facilitate solution of the nonlinear systems of equations, i.e., the solution for each step provides a starting point to the solution for the next step, thus more time steps make it easier for the solver. The time step size also in some cases (those with higher order quadrature) was controlled to avoid element inversion. Note that for this problem, the hypoelastic case is simpler to integrate since **R**=**I**, which led to a stress-rate integrand that was a low order rational function in time. For the mean quadrature Hex8 using the midpoint strain increment, the accuracy of the results depends upon the size of the time steps, so the time steps are as much as two orders of magnitude smaller.

This strain incrementation is typically used in explicit analyses, so the lower accuracy for this case in this context simply reflects its misapplication to quasistatic analyses. With time step refinement the midpoint strain increment did show linear convergence (though that should be addressed in another test), but the time steps were not reduced to produce the optimum accuracy.

3.2.1.3. Material Model

Two material model cases are examined, one neo-Hookean (Lame's neo-Hookean model) and one hypoelastic (Lame's elastic model). Aprepro was used to express the Lame elastic constants (λ and μ) in terms of Young's modulus and Poisson's ratio, with the later property set given below.

Young's Modulus	E	1.0×10^6 psi
Poisson's Ratio	ν	0.25

3.2.2. Expected Results

These patch tests, unlike the linear elastic versions of the tests, make no assumptions on the size of the displacements and deformations. The reference solutions are exact for finite deformation, and two deformation levels are considered: one in which the strains are $O(1\%)$ and the other for which the strains are $O(100\%)$. Due to the spatial symmetries of the displacement field, it yields a symmetric deformation gradient tensor (**F**), which implies that **R**=**I** and thus **U**=**V**=**F**. Furthermore the spatial dependence of the displacement components result in all normal components of the following deformation and stress tensors to be equal (for a given tensor) and all shear components of the following deformation and stress tensors to be equal. Symmetry of **F** is a limitation of this test, with regard to coverage, that motivates another finite deformation patch test.

For Lame's [1] neo-Hookean model the deformation used in the constitutive equation is the left Cauchy-Green tensor (or *finger* tensor, **B**). The exact solutions for the normal and shear components of **B** are given by:

$$B_{normal} = 2\,t_c^2 + (1 + 2\,t_c)^2 \tag{3.13}$$

$$B_{shear} = t_c^2 + 2\,t_c\,(1 + 2\,t_c) \tag{3.14}$$

For Lame's [1] elastic model (a hypoelastic model) the deformation tensor used in the constitutive relationship is the rate of deformation tensor, **D**. The exact solutions for each normal and shear component of **D** are given by:

$$D_{normal} = \frac{2\,c\,(1 + 2\,t_c)}{(1 + t_c)\,(1 + 4\,t_c)} \tag{3.15}$$

$$D_{shear} = \frac{c}{(1 + t_c)\,(1 + 4\,t_c)} \tag{3.16}$$

The deformation examined in the test, for both constitutive model cases, is the spatial (or left) stretch tensor, **V**. The exact solutions for the normal and shear components of the tensor are given by:

$$V_{normal} = 1 + 2\,t_c \tag{3.17}$$

$$V_{shear} = t_c \tag{3.18}$$

Using the constitutive models, one can obtain the Cauchy stress tensor exactly for each problem. For the Lame neo-Hookean model, the Cauchy stress is expressed explicitly in terms of the left Cauchy-Green tensor . The normal stress components for the neo-Hookean material are given by the expression:

$$\sigma_{normal} = \frac{t_c\,\left(12 + 54\,t_c + 116\,t_c^2 + 129\,t_c^3 + 72\,t_c^4 + 16\,t_c^5\right)\,(3\,\lambda + 2\,\mu)}{6\,(1 + t_c)^2\,(1 + 4\,t_c)} \tag{3.19}$$

The shear stress components for the neo-Hookean material are given by the expression:

$$\sigma_{shear} = \frac{t_c\,(2 + 5\,t_c)\,\mu}{\left((1 + t_c)^2\,(1 + 4\,t_c)\right)^{\frac{5}{3}}} \tag{3.20}$$

For the Lame elastic model the Cauchy stress is obtained from a relationship that includes an integration of the pull-back (using only the rotation tensor) of the Green-McInnis-Naghdi objective stress rate to obtain the change (from time 0 to time t) in the rotated Cauchy stress. The normal stress components for the hypoelastic material are given by the expression:

$$\sigma_{normal} = \frac{(3\,\lambda + 2\,\mu)\,\left(2\,\log(1 + t_c) + \log(1 + 4\,t_c)\right)}{3} \tag{3.21}$$

The shear stress components for the hypoelastic material are given by the expression:

$$\sigma_{shear} = \frac{2\,\mu\,\left(-\log(1 + t_c) + \log(1 + 4\,t_c)\right)}{3} \tag{3.22}$$

Many authors have noted that the patch test results should have an accuracy approaching the precision of the floating point numbers on the particular computer (see e.g., ref. 2), with a maximum variation of a few digits (depending upon the number of digits used in the machine arithmetic). For a double-precision floating point word, we would thus expect a relative error $O(10^{-13})$ or less.

3.2.3. Verification Results

For all the hexahedral element patch tests, the error in each normal and shear stress component was examined, where the error was defined as the maximum over all elements and applicable components of the differences between the finite deformation reference solution and the calculated results, normalized by the exact value. The errors are examined for t=1.

Tables 3-1 through 3-4 show the relative errors and tolerances for the $O(1\%)$ strain loading cases, both for the neo-Hookean and hypoelastic material models. For all of the element formulations except (1) Hex8 mean quadrature, midpoint strain incrementation, and (2) hex8 Q1P0, strongly objective strain incrementation the results are within the expected high accuracy. Issues with the Q1P0 element formulation were previously discussed with the linear elastic version of the patch test and will not be discussed further here. As discussed above, the midpoint strain incrementation is expected to be less accurate for comparable time stepping. Note that the error in shear components of the stretch and stress tensor were as much as two orders of magnitude larger than those of the normal components but still within the expected range.

Table 3-1. Stretch tensor maximum errors and tolerances for each neo-Hookean, O(1%) strain test.

$Test\ ID$	$Max\ V_{normal\ error}$	$V_{normal\ toler}$	$Max\ V_{shear\ error}$	$V_{shear\ toler}$
$hex08\_meanq\_mp$	7.426E-09	5.000E-06	1.250E-06	5.000E-04
$hex08\_meanq\_so$	2.198E-16	1.000E-13	1.197E-14	5.000E-13
$hex08\_sdev\_so$	2.198E-16	2.000E-13	4.528E-14	5.000E-13
$hex08\_q1p0\_so$	1.603E-03	4.000E-03	6.308E-02	7.000E-02
$hex08\_full\_so$	4.397E-16	2.000E-13	4.528E-14	7.000E-13
$hex20\_full\_so$	6.595E-16	5.000E-13	6.748E-14	1.000E-12
$hex27\_full\_so$	6.595E-16	5.000E-13	7.858E-14	5.000E-13

Table 3-2. Cauchy stress tensor maximum errors and tolerances for each neo-Hookean, O(1%) strain test.

$Test\ ID$	$Max\ \sigma_{normal\ el\ error}$	$\sigma_{normal\ el\ toler}$	$Max\ \sigma_{shear\ el\ error}$	$\sigma_{shear\ el\ toler}$
$hex08\_meanq\_mp$	7.422E-07	2.000E-04	1.224E-06	5.000E-04
$hex08\_meanq\_so$	2.327E-14	5.000E-13	1.333E-14	9.000E-13
$hex08\_sdev\_so$	1.906E-14	5.000E-13	8.968E-15	9.000E-13
$hex08\_q1p0\_so$	1.239E-01	2.500E-01	1.908E-02	3.000E-02
$hex08\_full\_so$	2.034E-14	5.000E-13	1.109E-14	9.000E-13
$hex20\_full\_so$	1.356E-14	5.000E-13	1.310E-14	9.000E-13
$hex27\_full\_so$	1.264E-14	5.000E-13	6.018E-15	9.000E-13

Table 3-3. Stretch tensor maximum errors and tolerances for each hypoelastic, O(1%) strain test.

$Test\ ID$	$Max\ V_{normal\ error}$	$V_{normal\ toler}$	$Max\ V_{shear\ error}$	$V_{shear\ toler}$
$hex08\_meanq\_mp$	7.425E-09	1.000E-08	1.250E-06	2.000E-06
$hex08\_meanq\_so$	6.595E-16	1.000E-13	4.528E-14	5.000E-13
$hex08\_sdev\_so$	4.397E-16	2.000E-13	4.354E-14	5.000E-13
$hex08\_q1p0\_so$	1.574E-03	4.000E-03	6.167E-02	7.000E-02
$hex08\_full\_so$	6.595E-16	2.000E-13	4.354E-14	5.000E-13
$hex20\_full\_so$	8.794E-16	5.000E-13	6.748E-14	1.000E-12
$hex27\_full\_so$	8.794E-16	5.000E-13	1.435E-13	1.000E-12

Table 3-4. Cauchy stress tensor maximum errors and tolerances for each hypoelastic, O(1%) strain test.

$Test\ ID$	$Max\ \sigma_{normal\ el\ error}$	$\sigma_{normal\ el\ toler}$	$Max\ \sigma_{shear\ el\ error}$	$\sigma_{shear\ el\ toler}$
$hex08\_meanq\_mp$	2.201E-13	5.000E-13	4.345E-13	1.000E-12
$hex08\_meanq\_so$	1.356E-14	5.000E-13	9.646E-14	6.000E-13
$hex08\_sdev\_so$	1.155E-14	5.000E-13	4.823E-14	1.000E-12
$hex08\_q1p0\_so$	1.217E-01	1.300E-01	1.559E-02	2.000E-02
$hex08\_full\_so$	1.283E-14	3.000E-13	6.135E-14	9.000E-13
$hex20\_full\_so$	1.594E-14	3.000E-13	5.606E-14	6.000E-13
$hex27\_full\_so$	2.236E-14	5.000E-13	6.078E-14	6.000E-13

Tables 3-5 through 3-8 show the relative errors and tolerances for the $O(100\%)$ strain loading cases, both for the neo-Hookean and hypoelastic material models. Again for all of the element formulations except (1) Hex8 mean quadrature, midpoint strain incrementation, and (2) hex8 Q1P0, strongly objective strain incrementation the results are within the expected high accuracy. As seen for the lower stain level, the error in shear components of the stretch and stress tensor were in a few cases as much as two orders of magnitude larger than those of the normal components but still within the expected range. For this loading case, this level of difference between the normal and shear components was limited to Cauchy stresses for three element formulations using the neo-Hookean model.

Table 3-5. Stretch tensor maximum errors and tolerances for each neo-Hookean, O(100%) strain test.

$Test\ ID$	$Max\ V_{normal\ error}$	$V_{normal\ toler}$	$Max\ V_{shear\ error}$	$V_{shear\ toler}$
$hex08\_meanq\_mp$	9.998E-05	1.000E-04	2.499E-04	5.000E-04
$hex08\_meanq\_so$	3.405E-15	1.000E-12	1.199E-14	1.000E-12
$hex08\_sdev\_so$	1.273E-14	5.000E-13	3.286E-14	5.000E-13
$hex08\_q1p0\_so$	9.669E-01	2.000E+00	1.440E+00	3.000E+00
$hex08\_full\_so$	1.140E-14	5.000E-13	2.520E-14	5.000E-13
$hex20\_full\_so$	3.153E-14	5.000E-13	5.207E-14	1.000E-12
$hex27\_full\_so$	4.086E-14	5.000E-13	1.740E-13	1.000E-12

Table 3-6. Cauchy stress tensor maximum errors and tolerances for each neo-Hookean, O(100%) strain test.

$Test\ ID$	$Max\ \sigma_{normal\ el\ error}$	$\sigma_{normal\ el\ toler}$	$Max\ \sigma_{shear\ el\ error}$	$\sigma_{shear\ el\ toler}$
$hex08\_meanq\_mp$	2.110E-04	5.000E-04	2.143E-05	5.000E-05
$hex08\_meanq\_so$	5.462E-15	1.000E-12	1.472E-13	1.000E-12
$hex08\_sdev\_so$	8.123E-15	5.000E-13	3.140E-14	1.000E-12
$hex08\_q1p0\_so$	2.256E+00	3.000E+00	1.504E+02	2.000E+02
$hex08\_full\_so$	4.201E-16	5.000E-13	5.552E-14	1.000E-12
$hex20\_full\_so$	1.120E-15	5.000E-13	3.982E-14	9.000E-13
$hex27\_full\_so$	9.803E-16	5.000E-13	4.844E-14	9.000E-13

Table 3-7. Stretch tensor maximum errors and tolerances for each hypoelastic, O(100%) strain test.

$Test\ ID$	$Max\ V_{normal\ error}$	$V_{normal\ toler}$	$Max\ V_{shear\ error}$	$V_{shear\ toler}$
$hex08\_meanq\_mp$	9.998E-05	1.000E-04	2.499E-04	5.000E-04
$hex08\_meanq\_so$	5.329E-15	5.000E-13	4.441E-15	1.000E-12
$hex08\_sdev\_so$	5.329E-15	5.000E-13	1.310E-14	5.000E-13
$hex08\_q1p0\_so$	2.282E-01	5.000E-01	8.709E-02	2.000E-01
$hex08\_full\_so$	3.849E-15	5.000E-13	7.994E-15	5.000E-13
$hex20\_full\_so$	1.850E-14	5.000E-13	3.131E-14	5.000E-13
$hex27\_full\_so$	1.658E-14	5.000E-13	3.375E-14	5.000E-13

Table 3-8. Cauchy stress tensor maximum errors and tolerances for each hypoelastic, O(100%) strain test.

$Test\ ID$	$Max\ \sigma_{normal\ el\ error}$	$\sigma_{normal\ el\ toler}$	$Max\ \sigma_{shear\ el\ error}$	$\sigma_{shear\ el\ toler}$
$hex08\_meanq\_mp$	2.345E-09	9.000E-09	6.644E-09	9.000E-09
$hex08\_meanq\_so$	5.013E-15	2.000E-13	1.203E-14	9.000E-13
$hex08\_sdev\_so$	2.332E-16	5.000E-13	2.859E-15	9.000E-13
$hex08\_q1p0\_so$	2.372E-01	5.000E-01	2.544E-01	5.000E-01
$hex08\_full\_so$	8.161E-16	2.000E-13	2.978E-15	9.000E-13
$hex20\_full\_so$	1.166E-15	2.000E-13	2.859E-15	9.000E-13
$hex27\_full\_so$	9.326E-16	5.000E-13	2.144E-15	9.000E-13

3.2.4. References

1. Scherzinger, W.M. and Hammerand, D.C. *Constitutive Models in Lame*. Sandia Report SAND2007-5873, September 2007.
2. Belytschko, T., Liu, W.K., and Moran B. *Nonlinear Finite Elements for Continua and Structures*, NY, NY: John Wiley & Sons, LTD, 2000.
3. Hughes, T.J.R. *The Finite Element Method, Linear Static and Dynamics Finite Element Analysis*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.
4. Fung, Y.C. *Foundations of Solid Mechanics*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1965.
5. Bonet, J., and Wood, D. *Nonlinear Continuum Mechanics for Finite Element Analysis*, 2nd edition, NY, NY: Cambridge University Press, 2008.

For input deck see Appendix B.15.

3.3. HEX PATCH TEST – UNIFORM GRADIENT, STRONGLY OBJECTIVE

Analysis Type	Explicit Transient Dynamics
Element Type	Hex8
Strain Incrementation	Strongly Objective
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Displacemnet, Stress Fields
Number of Tests	1
Keywords	Patch Test

3.3.1. Brief Description

This problem is a patch test for a uniform-strain, eight-node hexahedral element with a strongly-objective formulation for the strain.

3.3.1.1. Functionality Tested

Primary capabilities:

- uniform-strain, three-dimensional, eight-node hexahedral element with strongly objective strain formulation for the strain

Secondary capabilities:

- prescribed displacement direction boundary condition
- resolution of kinematic boundary condition

3.3.1.2. Mechanics of Test

The mesh in this problem is a unit cube with seven elements. Each of the elements represents a single element block. Six of the elements have one face on the exterior, and one element has all interior faces. To provide a completely general test, the interior element has no parallel or perpendicular edges. The interior element (element block 1) is shown in Figure 3-5. None of the faces of the interior element are perpendicular or parallel to the planes xy, yz, zx defined by the x-, y-, and z-axes.

Figure 3-6 is drawn from all of the elements except the element defining element block 3, which has an exterior face with a normal in the positive z-direction. The interior element and four surrounding elements are visible. The element with an exterior face with a normal in the negative z-direction is not visible in this hidden-line drawing of the elements.

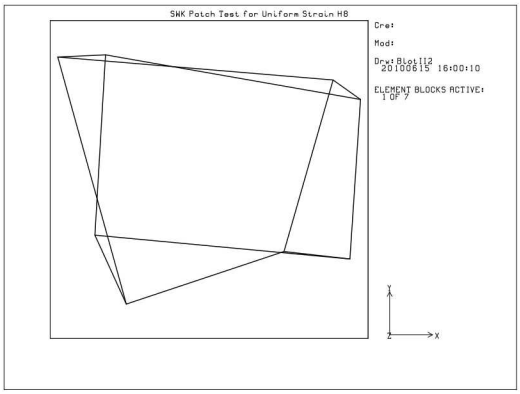


Figure 3-5. Interior element for patch-test cube

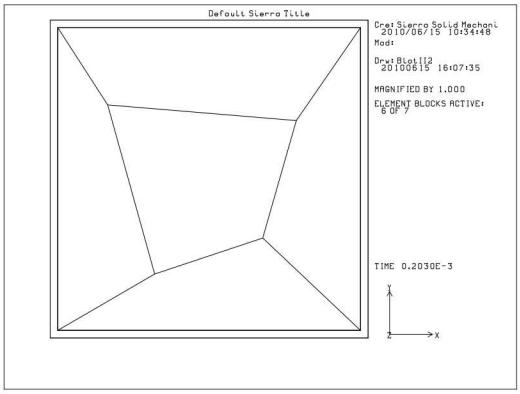


Figure 3-6. Patch-test cube with element block 3 not shown

An applied displacement field is described by the following equations:

$$u = f(t) \times (1.0 \times 10^{-3}) \times (2x + y + z) \tag{3.23}$$

$$v = f(t) \times (1.0 \times 10^{-3}) \times (x + 2y + z) \tag{3.24}$$

$$w = f(t) \times (1.0 \times 10^{-3}) \times (x + y + 2z) \tag{3.25}$$

The displacement in the x-direction is u, the displacement in the y-direction is v, and the displacement in the z-direction is w. In the above equations, the function $f(t)$ is defined by:

$$f(t) = 0.5\left(1 - cos\left(\frac{\pi \times t}{2.0 \times 10^{-3}}\right)\right) \tag{3.26}$$

from 0 to 2.0×10^{-3} sec. For time greater than 2.0×10^{-3} sec, $f(t)$ is defined by:

$$f(t) = 1.0 \tag{3.27}$$

The function $f(t)$ brings the displacement field to a steady-state condition for time t greater than 2.0×10^{-3}.

3.3.1.3. Material Model

The material model is linear elastic with the values:

Young's Modulus	E	1.0×10^6 Psi
Poisson's Ratio	ν	0.25
Density	ρ	2.61×10^{-3} lbm/in^3

3.3.2. Expected Results

For the small-strain case, the stress σ_{xx} is related to the strains ϵ_{xx}, ϵ_{yy}, and ϵ_{zz} by:

$$\sigma_{xx} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\left[\epsilon_{xx} + \frac{\nu}{(1-\nu)}(\epsilon_{yy} + \epsilon_{zz})\right] \tag{3.28}$$

Similar equations hold for σ_{yy} and σ_{zz}. The shear stress τ_{xy} is related to the shear strain ϵ_{xy} by:

$$\tau_{xy} = \frac{E}{(1+\nu)}\epsilon_{xy} \tag{3.29}$$

Similar equations hold for τ_{yz} and τ_{zx}.

For the steady-state conditions,

$$\epsilon_{xx} = \epsilon_{yy} = \epsilon_{zz} = 2 \times 10^{-3} \tag{3.30}$$

and

$$\epsilon_{xy} = \epsilon_{yz} = \epsilon_{zx} = 1 \times 10^{-3} \tag{3.31}$$

For the small-strain case, the stress field produced by the displacement field in the steady-state region is given by:

$$\sigma_{xx} = \sigma_{yy} = \sigma_{zz} = 4000\,psi \tag{3.32}$$

and

$$\tau_{xy} = \tau_{yz} = \tau_{zx} = 800\,psi \tag{3.33}$$

The following plots are the stresses from element 1, although elements 1 through 7 should have the exact same stresses.

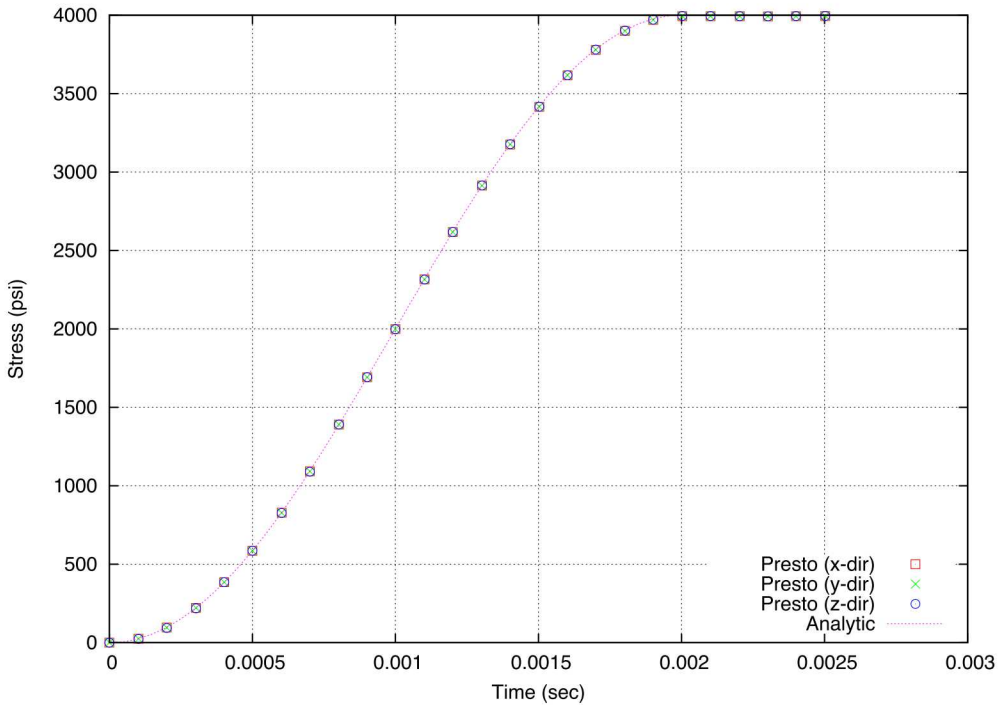


Figure 3-7. Stress x-, y-, and z-direction for element 1

3.3.3. References

1. Fung, Y.C. **Foundations of Solid Mechanics**. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1965.

For input deck see Appendix B.16.

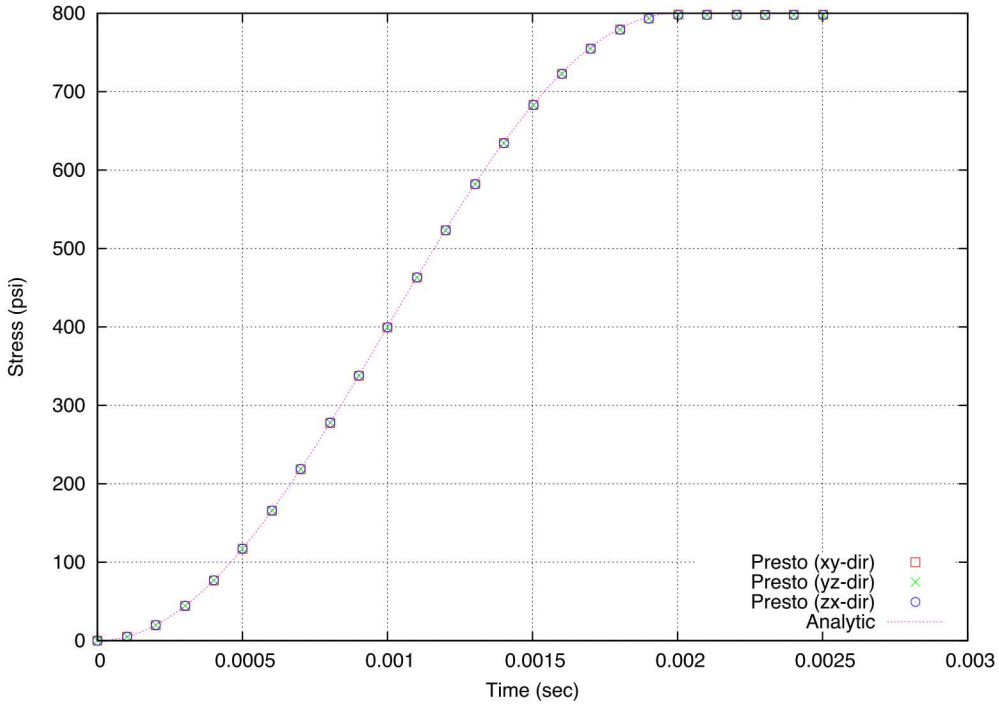


Figure 3-8. Stress in xy-, yz- and zx-direction for element 1

3.4. HEX PATCH TEST – UNIFORM GRADIENT, MIDPOINT INCREMENT

Analysis Type	Explicit Transient Dynamics
Element Type	Hex8
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Displacement, Stress Fields
Number of Tests	1
Keywords	Patch Test

3.4.1. Brief Description

This problem is a patch test for a uniform-strain, eight-node hexahedral element with a midpoint-increment formulation for the strain.

3.4.1.1. Functionality Tested

Primary capabilities:

- uniform-strain, three-dimensional, eight-node hexahedral element with midpoint
- increment strain formulation for the strain

Secondary capabilities:

- prescribed displacement direction boundary condition
- resolution of kinematic boundary condition

3.4.1.2. Mechanics of Test

The mesh in this problem is a unit cube with seven elements. Each of the elements represents a single element block. Six of the elements have one face on the exterior, and one element has all interior faces. To provide a completely general test, the interior element has no parallel or perpendicular edges. The interior element (element block 1) is shown in Figure 3-9. None of the faces of the interior element are perpendicular or parallel to the planes xy, yz, zx defined by the x-, y-, and z-axes.

Figure 3-10 is drawn from all of the elements except the element defining element block 3, which has an exterior face with a normal in the positive z-direction. The interior element and four surrounding elements are visible. The element with an exterior face with a normal in the negative z-direction is not visible in this hidden-line drawing of the elements.

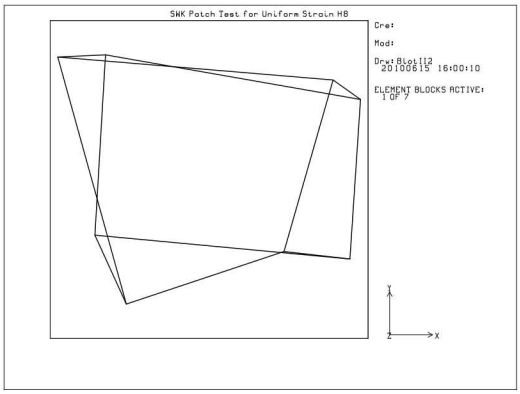


Figure 3-9. Interior element for patch-test cube

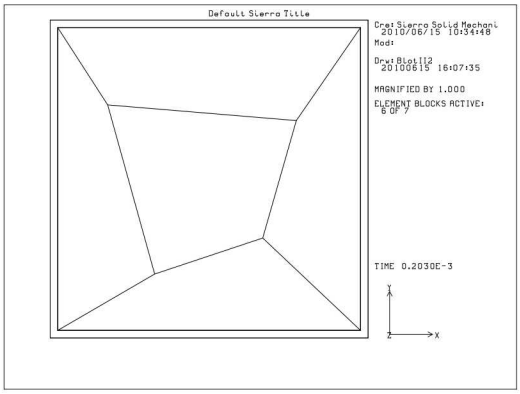


Figure 3-10. Patch-test cube with element block 3 not shown

An applied displacement field is described by the following equations:

$$u = f(t) \times (1.0 \times 10^{-3}) \times (2x + y + z) \tag{3.34}$$

$$v = f(t) \times (1.0 \times 10^{-3}) \times (x + 2y + z) \tag{3.35}$$

$$w = f(t) \times (1.0 \times 10^{-3}) \times (x + y + 2z) \tag{3.36}$$

The displacement in the x-direction is u, the displacement in the y-direction is v, and the displacement in the z-direction is w. In the above equations, the function $f(t)$ is defined by:

$$f(t) = 0.5\left(1 - cos\left(\frac{\pi \times t}{2.0 \times 10^{-3}}\right)\right) \tag{3.37}$$

from 0 to 2.0×10^{-3} sec. For time greater than 2.0×10^{-3} sec, $f(t)$ is defined by:

$$f(t) = 1.0 \tag{3.38}$$

The function $f(t)$ brings the displacement field to a steady-state condition for time t greater than 2.0×10^{-3}.

3.4.1.3. Material Model

The material model is linear elastic with the values:

Young's Modulus	E	1.0×10^6 Psi
Poisson's Ratio	ν	0.25
Density	ρ	2.61×10^{-3} lbm/in^3

3.4.2. Expected Results

For the small-strain case, the stress σ_{xx} is related to the strains ϵ_{xx}, ϵ_{yy}, and ϵ_{zz} by:

$$\sigma_{xx} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\left[\epsilon_{xx} + \frac{\nu}{(1-\nu)}(\epsilon_{yy} + \epsilon_{zz})\right] \tag{3.39}$$

Similar equations hold for σ_{yy} and σ_{zz}. The shear stress τ_{xy} is related to the shear strain ϵ_{xy} by:

$$\tau_{xy} = \frac{E}{(1+\nu)}\epsilon_{xy} \tag{3.40}$$

Similar equations hold for τ_{yz} and τ_{zx}.

For the steady-state conditions,

$$\epsilon_{xx} = \epsilon_{yy} = \epsilon_{zz} = 2 \times 10^{-3} \tag{3.41}$$

and

$$\epsilon_{xy} = \epsilon_{yz} = \epsilon_{zx} = 1 \times 10^{-3} \tag{3.42}$$

For the small-strain case, the stress field produced by the displacement field in the steady-state region is given by:

$$\sigma_{xx} = \sigma_{yy} = \sigma_{zz} = 4000\,psi \tag{3.43}$$

and

$$\tau_{xy} = \tau_{yz} = \tau_{zx} = 800\,psi \tag{3.44}$$

The following plots are the stresses from element 1, although elements 1 through 7 should have the exact same stresses.

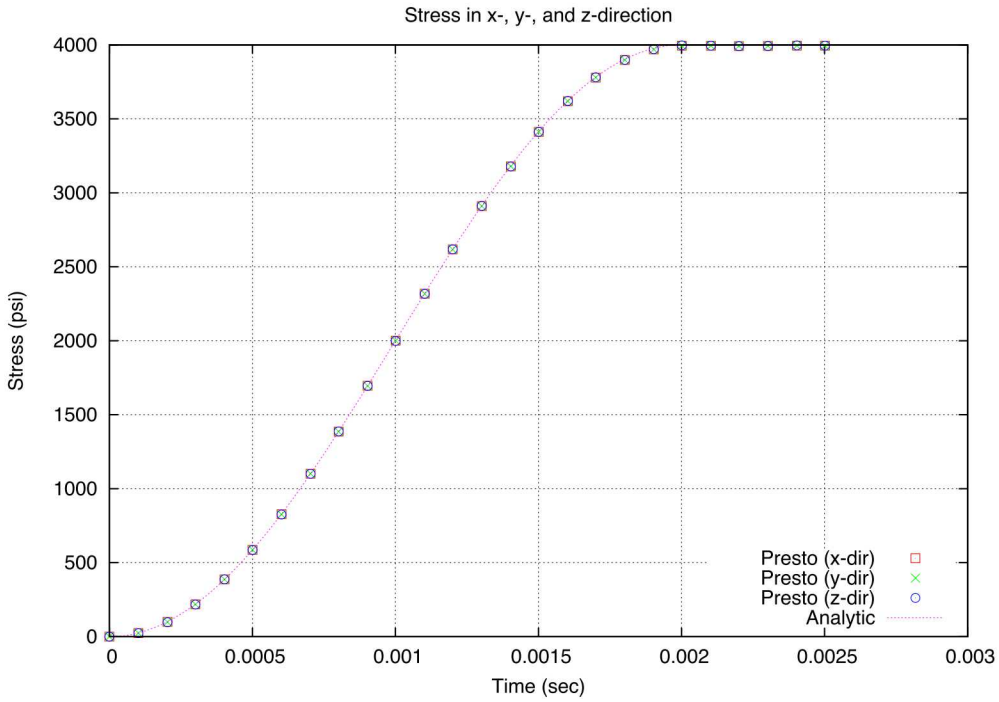


Figure 3-11. Stress x-, y-, and z-direction for element 1

3.4.3. References

1. Fung, Y.C. **Foundations of Solid Mechanics**. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1965.

For input deck see Appendix B.17.

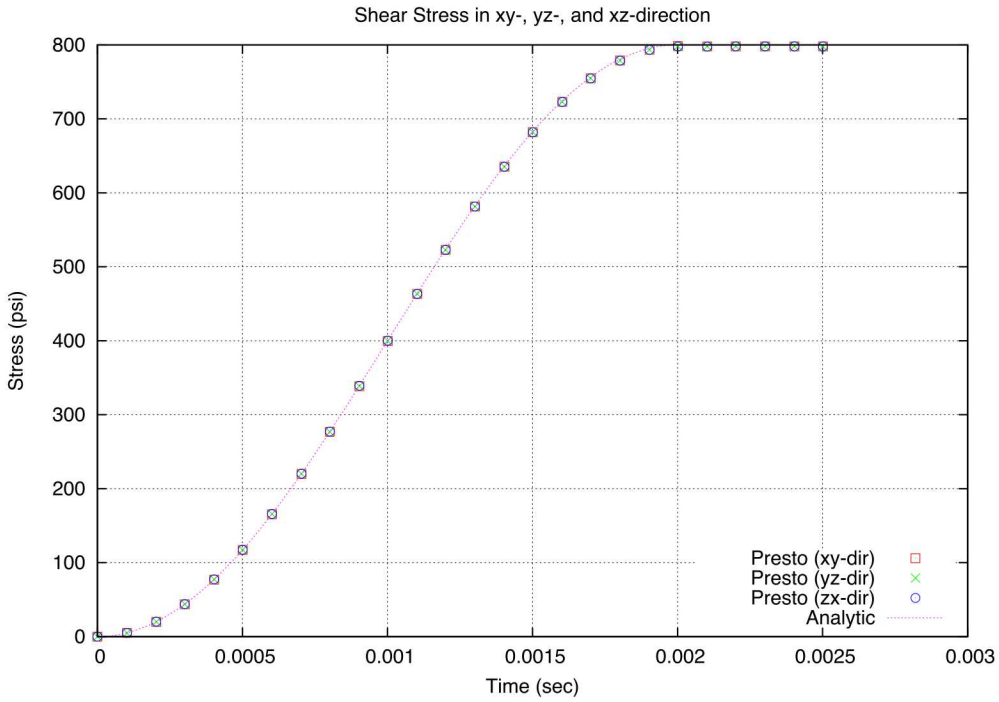


Figure 3-12. Stress in xy-, yz- and zx-direction for element 1

3.5. HEX PATCH TEST – UNIFORM GRADIENT, MIDPOINT INCREMENT, THERMAL

Analysis Type	Explicit Transient Dynamics
Element Type	Hex8
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Displacement, Stress Field
Number of Tests	1
Keywords	Patch Test, Thermal Strains, Elastic

3.5.1. Problem Description

This problem is a version of the hexahedral patch tests, but in this case with free thermal expansion.

3.5.1.1. Functionality Tested

Primary capabilities:

- uniform thermal strains

Secondary capabilities:

- uniform-gradient, eight-node, three-dimensional hexahedron with midpoint-increment formulation
- fixed component displacement boundary condition

3.5.1.2. Boundary Conditions

This problem uses the same mesh as that used for the other hex8 patch tests. The mesh is a simple cube; boundary conditions are applied so that the elements in the mesh undergo a free expansion.

The mesh in this problem is a unit cube with seven elements. Each of the elements represents a single block. Six of the elements have one face on the exterior, and one element has all interior faces. To provide a completely general test, the interior element has no parallel or perpendicular edges. The interior element (element block 1) is shown in Figure 3-13. None of the faces of the interior element are perpendicular or parallel to the planes xy, yz, and zx defined by the x-, y-, and z-axes.

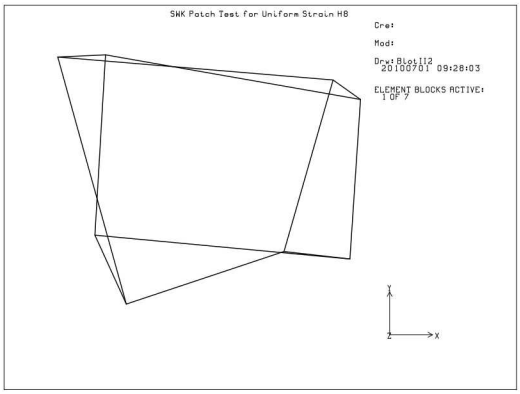


Figure 3-13. Interior element for patch-test cube.

Figure 3-14 is drawn from all of the elements except the element defining element block 3, which has an exterior face with a normal in the positive z-direction. The interior element and four surrounding elements are visible. The element with an exterior face with a normal in the negative z-direction is not visible in this hidden-line drawing of the elements.

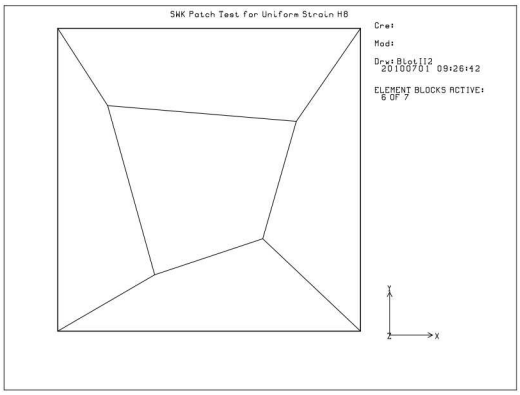


Figure 3-14. Patch-test cube with element block 3 not shown.

Boundary and temperature conditions are applied to generate a free thermal expansion of the block. Zero displacement boundary conditions are placed on the $x = 0$, $y = 0$, and $z = 0$ planes. These fixed displacement boundary conditions are symmetry conditions. This test problem represents, therefore, a cube made of eight unit cubes centered on the origin. The only loading on

the model is a thermal load. The full cube (eight unit cubes centered on the origin) undergoes a free thermal expansion. The temperature of the unit cube, T, increases by one degree from time $t = 0$ to time $t = 2.0 \times 10^{-3}$ sec based on the function:

$$T(t) = 0.5 \left(1 - cos \left(\frac{\pi t}{2.0 \times 10^{-3}}\right)\right) \tag{3.45}$$

The temperature remains constant at:

$$T(t) = 1.0 \tag{3.46}$$

for $t > 2.0 \times 10^{-3}$. The thermal-strain-versus-temperature curve for the material is a line with a slope of 0.001 inch/degree. Thus a temperature increase of one degree will generate a thermal strain of 0.001 inch.

3.5.1.3. Material Model

The material properties are linear elastic.

Table 3-9. Material Properties

Young's Modulus	E	1.0×10^6 psi
Poisson's Ratio	ν	0.25
Mass Density	ρ	2.61×10^{-4} $lb \cdot s^2/in^4$

3.5.2. Verification of Solution

The block in this problem undergoes thermal expansion with no kinematic boundary conditions. The stresses in all elements should be zero. The nodal displacement in the x-direction for any node lying in the plane $x = 1.0$ should be $u_x = 0.001$ inch, the nodal displacement in the y-direction for any node lying in the plane $y = 1.0$ should be $u_y = 0.001$ inch, and the nodal displacement in the z-direction for any node lying in the plane $z = 1.0$ should be $u_z = 0.001$ inch.

A discussion of thermal strains can be found in reference 1.

For time $t > 2.0 \times 10^{-3}$ sec, the thermal load represents a steady loading case. Since this is a transient dynamics problem, the solution will oscillate about the steady-state solution for time $t > 2.0 \times 10^{-3}$ sec. Figure 3-15, shows that for $t > 2.0 \times 10^{-3}$, the stresses in the model oscillate about zero with a maximum amplitude between 1 and 2 psi. The stress in the x-direction for the interior element, element 1, is shown in Figure 3-15. For $t > 2.0 \times 10^{-3}$, the displacement in the x-direction for a node lying in the plane $x = 1.0$ oscillates about the value 0.001 inch. The displacement in the x-direction for node 14, which is at $x = 1.0$ inch, $y = 1.0$ inch, and $z = 1.0$ inch is shown in Figure 3-16. The oscillation about the value 0.001 inch in the steady-state portion of the curve is very small.

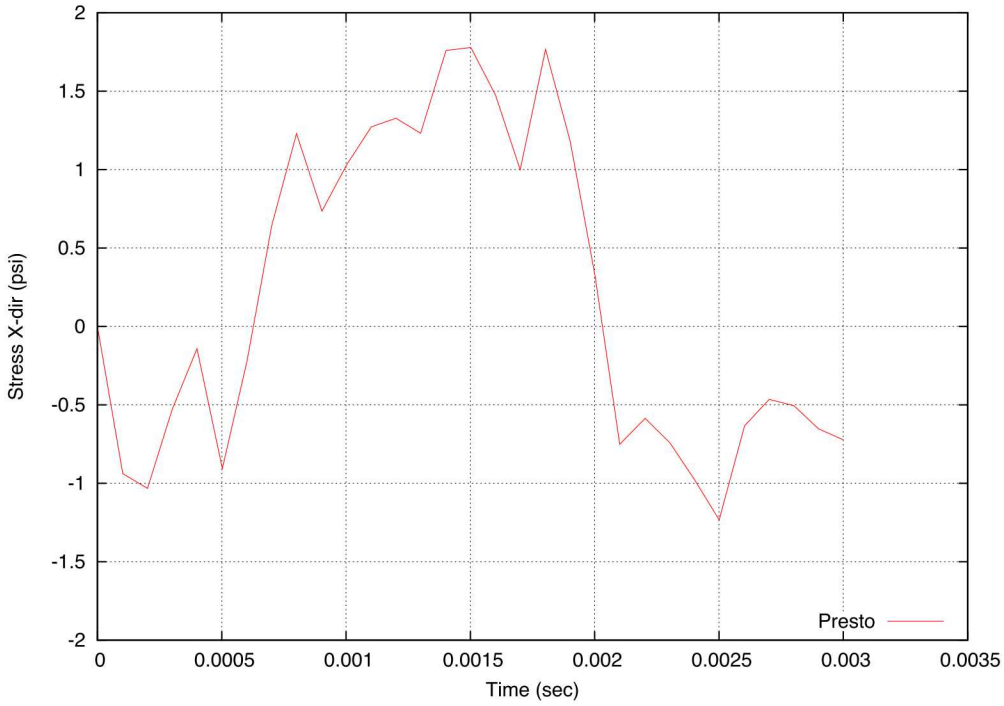


Figure 3-15. Stress in the x-direction for interior element 1.

3.5.3. References

1. Fung, Y. C. **Foundations of Solid Mechanics**. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1965.

For input deck see Appendix B.18.

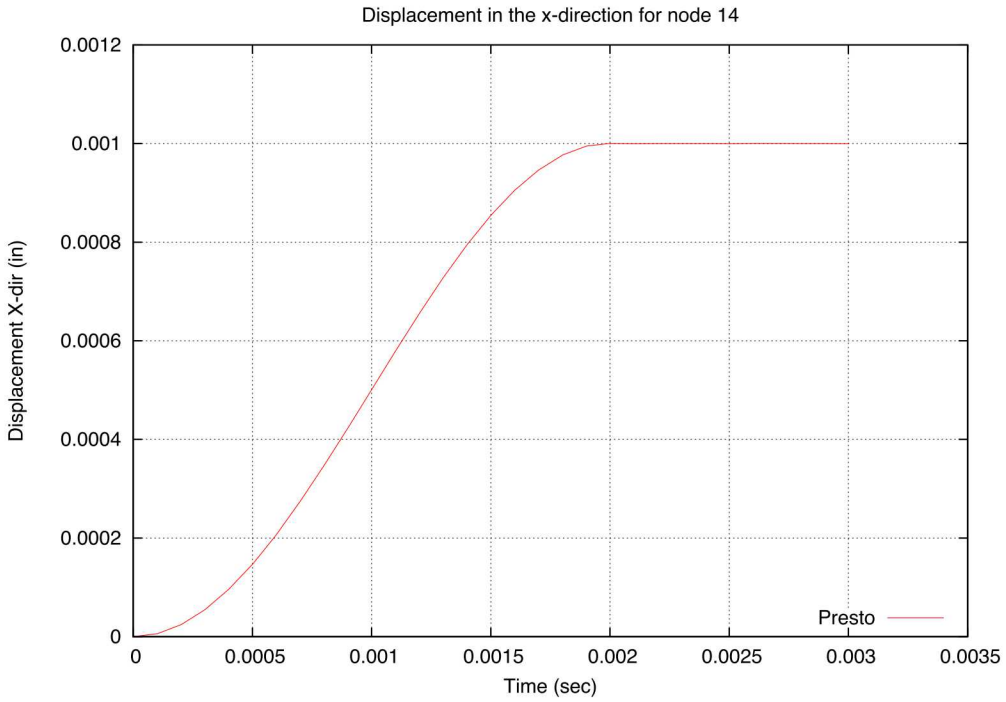


Figure 3-16. Displacement in the x-direction for node 14.

3.6. HEX CONVERGENCE TEST – CANTILEVER BEAM

Analysis Type	Quasi-statics
Element Types	Hex8
Element Formulations	Mean Quadrature, Q1P0, Selective Deviatoric, Fully Integrated
Strain Incrementation	Midpoint Increment, Strongly Objective
Material Models	Elastic
Verification Category	Convergence
Verification Quantities	Displacement, Stress Fields
Number of Tests	5
Keywords	Linear Elastic, Beam Theory

3.6.1. Brief Description

This set of analyses demonstrates the convergence of displacements and stresses for a relatively simple mechanical analysis. The system analyzed is a cantilever beam under unit tip loading. The reference solution is derived from Euler-Bernoulli beam theory. As such, the solution is not exact, and the finite element solution (if sufficiently refined) will deviate from this reference solution. Five different formulations of the hex8 are currently tested. Higher order hex elements will be included in the test once they have the capability to integrate surface tractions consistently with the underlying shape functions.

3.6.1.1. Functionality Tested

Primary capabilities:

- The following element formulations:
  - eight-node hexahedron with the mean quadrature formulation and midpoint strain incrementation
  - eight-node hexahedron with the mean quadrature formulation and strongly objective strain incrementation
  - eight-node hexahedron with the Q1P0 formulation and strongly objective strain incrementation
  - eight-node hexahedron with the selective deviatoric formulation and strongly objective strain incrementation
  - eight-node hexahedron with the fully-integrated formulation and strongly objective strain incrementation
- subjected to the limitations of linear elasticity.

Secondary capabilities:

- prescribed displacement boundary conditions
- resolution of kinematic boundary condition
- user subroutine application of traction boundary conditions

3.6.1.2. Mechanics of Test

The geometry of the general problem is shown in Figure 3-17. The beam is slender, with a length-to-depth ratio of 10, and it has a square cross-section.[1] The long direction of the beam is aligned with the x-axis, and the unit transverse shear loading is applied in the y-direction as shown.

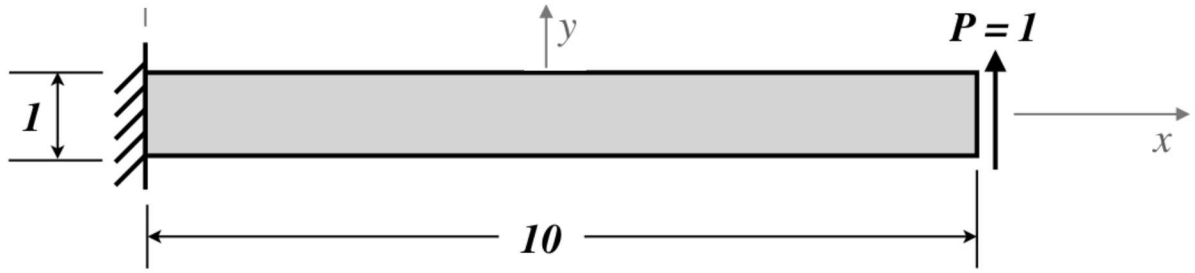


Figure 3-17. Beam geometry.

3.6.1.3. Material Model

The material model used for this problem is the elastic model implemented in Lame [1].
The selected properties are given as follows.

Young's Modulus	E	1.0×10^6 psi
Poisson's Ratio	ν	0.25
Density	ρ	1

3.6.1.4. Boundary Conditions

The boundary conditions for this problem are depicted in Figure 3-18 below. These are not the simplest boundary conditions, but they were defined to be consistent with the shear stress distribution of Euler-Bernoulli beam theory, while still preventing rigid body motions.

3.6.1.5. Meshes

The meshes used in this study are shown in Figure 3-19. These meshes consist of cubical finite elements, in densities ranging from 2 to 16 elements through the beam depth. Note that each node

[1] Units are consistent with those of length and force depicted in Figure 3-17 but not explicitly defined.

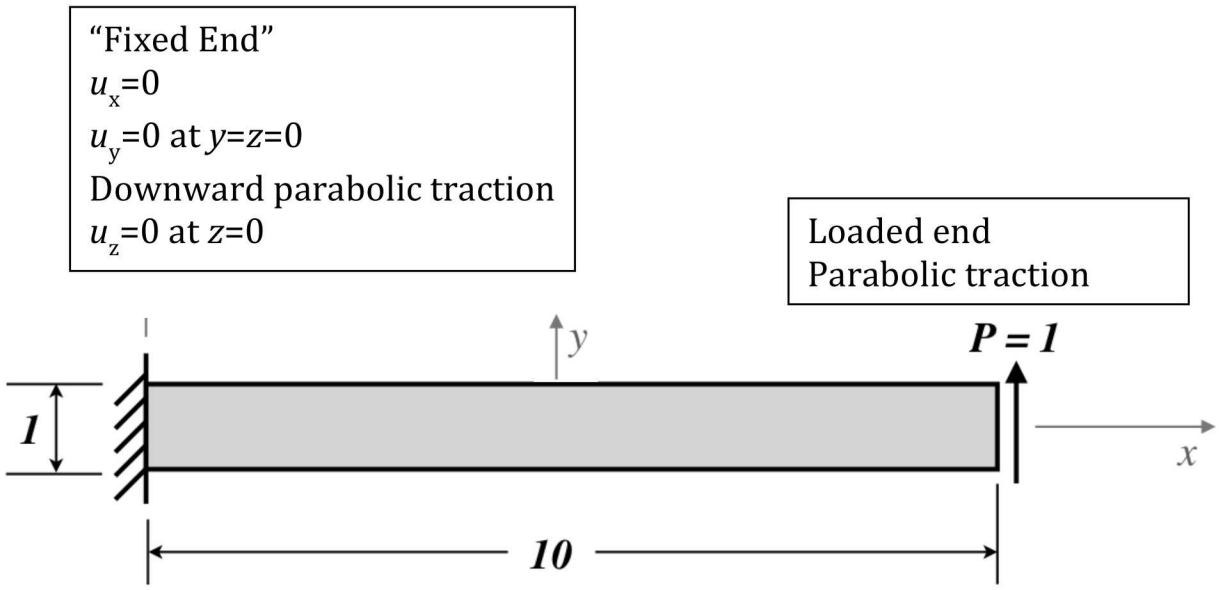
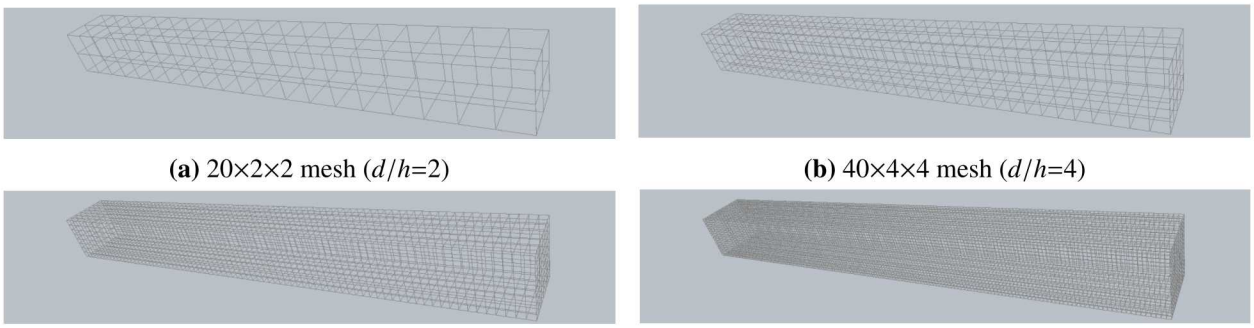


Figure 3-18. Beam boundary conditions.

in a relatively coarse mesh is present in the subsequent refined mesh. This arrangement implies that each successive refined finite element approximation subspace contains the last as a proper subset.



(a) 20×2×2 mesh (d/h=2)

(b) 40×4×4 mesh (d/h=4)

(c) 80×8×8 mesh (d/h=8)

(d) 160×16×16 mesh (d/h=16)

Figure 3-19. Meshes used in this study

3.6.2. Expected Results

As previously mentioned, the reference solution for this problem is obtained from Euler-Bernoulli beam theory. As such, it not only is based upon the approximation of linear elasticity but also has the kinematic constraint that plane sections perpendicular to the neutral reference surface (axis in the one-dimensional idealization) remain planner and perpendicular to the neutral axis under deformation. For brevity the expressions for the displacement and stress field are not formally presented here, but both are presented as string functions in the associated Encore input file. In brief, solving the fourth order differential equation of beam theory gives a vertical displacement that is cubic in x. The longitudinal displacement can be obtained from the rotation (combined with the kinematic assumption) or integrating the longitudinal strain from the fixed-end of the beam, and the transverse displacement can be obtained from integrating the transverse normal strain (ϵ_{zz}). Beam theory gives a linear distribution of the normal stress with respect to y, and a quadratic distribution of shear stress (σ_{xy}) with respect to y. Clearly the existence of the shear stress with no corresponding shear deformation reflects the approximations made in beam theory, approximations that while sufficiently accurate for engineering calculations of slender members conflict with the equations of elasticity. The lack of exactness of this reference solution, weakens its usefulness toward code verification [2], but the convergence tendencies are still apparent. While a higher order beam theory for linear elasticity that includes rotation of sections would be more appropriate for this problem, it would still be inexact relative to the underlying mathematical model of the code (since the code is based upon finite deformations).

3.6.3. Verification Results

In this test we examine the observed rates of convergence using the displacement vector and stress tensor [2] fields. In both cases we use a relative error measure of the norms of the errors divided by the norms of the Euler-Bernoulli reference response (versus the element size).

The slopes of the relative error curves between the data points corresponding to two meshes (on the log-log plots) yield observed rates of convergence. For an exact reference solution, the assessment of the rate of convergence improves with mesh refinement, assuming other sources of numerical error (e.g., solver accuracy) do not corrupt the results; this follows from each mesh refinement producing results that more accurately represent the asymptotic behavior. For this problem we are not using an exact solution, so an improvement in the convergence estimate is not guaranteed. In fact typically for problems without an exact solution there is a sweet range where the approximations are in the asymptotic range but not refined enough to measure the inexactness of the references solution. The following tables give the observed rates of convergence for each Hex8 element formulation between each sequential pair of meshes, where h_{fine} denotes the element size of the finer mesh of the pair. The following plots show the corresponding graphical representations of the difference data as a function of the element size. Because of the higher accuracy of the displacement response [2], the effect of the inexact reference solution is apparent

[2] L_2-norms are used for all the norm calculations. The integration of the norm calculations are approximated by Gauss quadrature over each element domain [2,3]. For the stress tensor, the norm is a vector norm of the Voigt notation representation of the stress tensor.

for the finest mesh (d/h=16); generally the more accurate the element formulation was for this problem, the greater its deviation from the expected convergence rate for the finest mesh.

Default parameters are used for the elements, and the selective deviatoric element uses a deviatoric parameter of one-half.

Figure 3-20 shows the convergence results for the displacement field for the Hex8 element formulations. For reference note that the Q1P0 element exhibits an observed rate of convergence of approximately 2. The fully-integrated element exhibits a slightly higher rate of convergence but with lower accuracy than the other elements until the finest mesh.

Figures 3-21 and 3-22 depict the convergence results for the element and nodal representations of the stress field, respectively, for all five hex8 element formulations. The results for the element stress fields are in reasonably close agreement with the optimal rates of convergence (linear). The results for the nodal stress field show an improvement in the observed rates of convergence within the approximate range 1.5 to 1.7. These results give a measure of verification for the elements for a linear elastic BVP but are less rigorous than an error quantification test (based upon an exact solution).

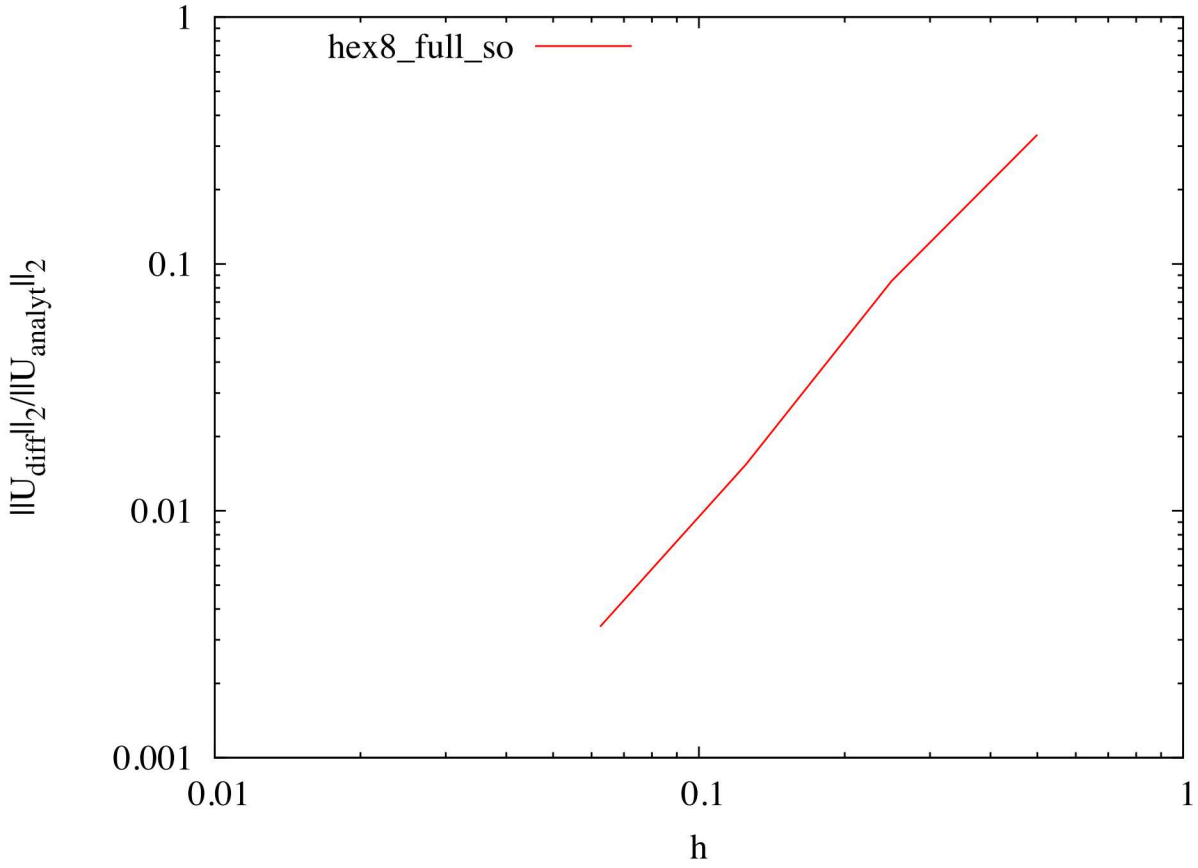


Figure 3-20. Convergence of the displacement vector in the L_2 norm – solution difference versus element size.

Table 3-10. Convergence rates for Hex8, fully integrated, strongly objective

h_{fine}	$\|\|U_{diff}\|\|_2/\|\|U_{analyt}\|\|_2$	$\|\|\sigma_{nodal\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$	$\|\|\sigma_{element\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$
0.2500	1.97	1.63	1.13
0.1250	2.47	1.74	1.06
0.0625	2.18	1.65	1.02

Table 3-11. Convergence rates for Hex8,mean quadrature,midpoint increment

h_{fine}	$\|\|U_{diff}\|\|_2/\|\|U_{analyt}\|\|_2$	$\|\|\sigma_{nodal\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$	$\|\|\sigma_{element\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$
0.2500	2.67	1.60	1.02
0.1250	2.30	1.58	1.00
0.0625	-1.18	1.47	1.00

Table 3-12. Convergence rates for Hex8, mean quadrature, strongly objective

h_{fine}	$\|\|U_{diff}\|\|_2/\|\|U_{analyt}\|\|_2$	$\|\|\sigma_{nodal\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$	$\|\|\sigma_{element\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$
0.2500	2.67	1.60	1.02
0.1250	2.30	1.58	1.00
0.0625	-1.18	1.47	1.00

Table 3-13. Convergence rates for Hex08, Q1P0, strongly objective

h_{fine}	$\|\|U_{diff}\|\|_2/\|\|U_{analyt}\|\|_2$	$\|\|\sigma_{nodal\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$	$\|\|\sigma_{element\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$
0.2500	2.08	1.64	1.08
0.1250	2.74	1.69	1.03
0.0625	0.92	1.58	1.01

Table 3-14. Convergence rates for Hex8,selective deviatoric, strongly objective

h_{fine}	$\|\|U_{diff}\|\|_2/\|\|U_{analyt}\|\|_2$	$\|\|\sigma_{nodal\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$	$\|\|\sigma_{element\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$
0.2500	2.41	1.62	1.03
0.1250	3.83	1.61	1.01
0.0625	-1.81	1.50	1.00

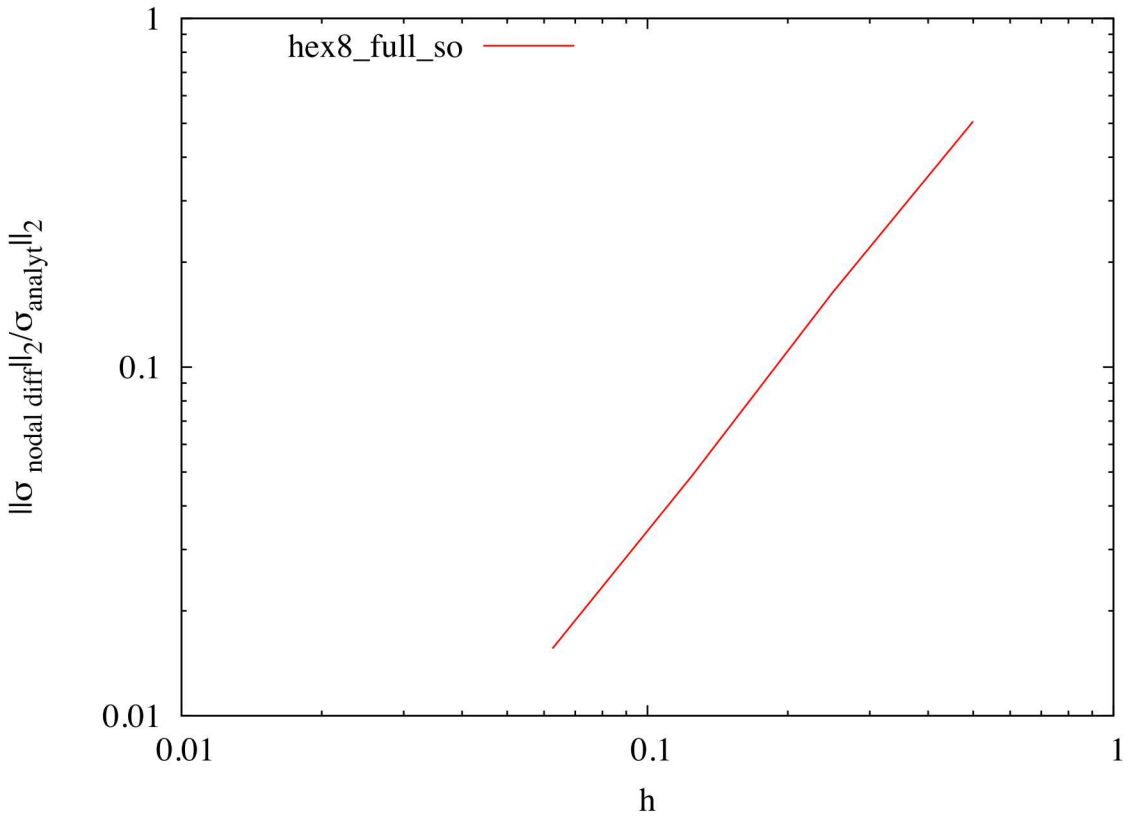


Figure 3-21. Convergence of the nodal representation of the Cauchy stress tensor in the L_2 norm – solution difference versus element size.

Summary of results: the Hex8 elements approximately exhibited the expected rates of convergence for displacements (quadratic) and for element stresses (linear), the exception being when the meshes became sufficiently fine for the displacements to reveal the inexactness of the reference solution; i.e., solutions based upon the finest finite element meshes are converging toward the exact solution and deviating from the beam theory solution. The nodal extrapolation of the stress field improves its rate of convergence to about 1.5. Since the reference solution is not exact, even for linear elasticity, this is a weaker convergence test, as discussed in the manual's introduction.

3.6.4. References

1. Scherzinger, W.M. and Hammerand, D.C. *Constitutive Models in Lame*. Sandia Report SAND2007-5873, September 2007.
2. Cox, J.V. and Mish, K.D. *Sierra solid mechanics example verification problems to highlight the use of Sierra verification tools*, October 2012 (in publication).
3. Copps, K.D. and Carnes, B.R. *Encore User Guide*, Sandia Report, October 2009.

For input deck see Appendix B.19.

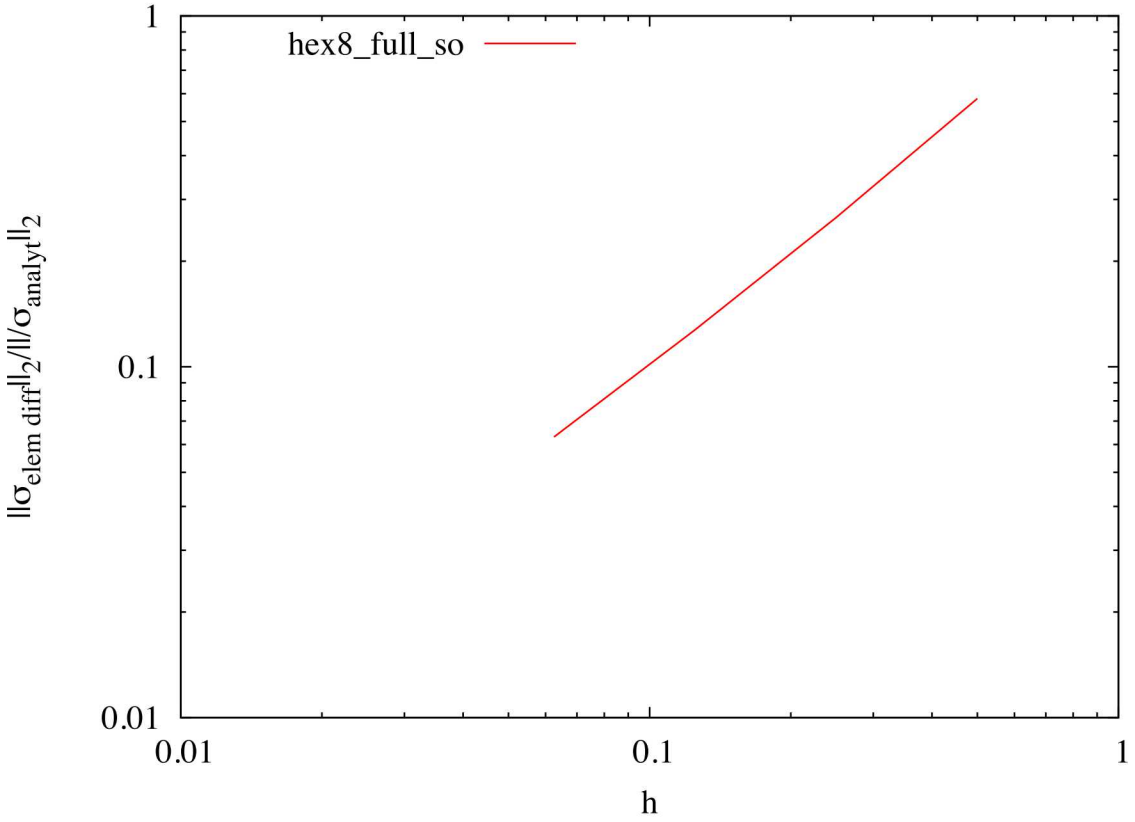


Figure 3-22. Convergence of the element representation of the Cauchy stress tensor in the L_2 norm – solution difference versus element size.

3.7. TET PATCH TESTS – QUASI-STATIC, LINEAR ELASTIC

Analysis Type	Quasi-statics
Element Types	Tet4, Tet10
Element Formulations	Mean Quadrature, Fully Integrated, Composite Tet
Strain Incrementation	Strongly Objective, Node Based
Material Models	Elastic, Neo-hookean
Verification Category	Discretization Error
Verification Quantities	Stress Components
Number of Tests	10
Keywords	Patch Test, Linear Elastic

3.7.1. Brief Description

This problem is a linear elastic patch test for tetrahedral elements with a strongly objective strain incrementation. The magnitude of the displacements is defined to be sufficiently small that linear elasticity provides a reasonable approximation of the expected response. A cubic domain is subjected to prescribed displacements on each surface. Ten test results are obtained for a combination of four element formulations (regular tet4, nodal-based tet4, complete quadratic tet10, and composite tet10) using two different elastic material models (Elastic and neo-Hookean).

3.7.1.1. Functionality Tested

Primary capabilities:

- The following element formulations with strongly objective strain incrementation:
  - regular, four-node tetrahedron
  - node-based, four-node tetrahedron
  - complete quadratic, ten-node tetrahedron
  - composite, ten-node tetrahedron

Secondary capabilities:

- prescribed displacement direction boundary conditions via analytic expressions
- resolution of kinematic boundary condition
- elastic and neo-Hookean material models in the small strain regime

3.7.1.2. Mechanics of Test

A unit cube domain is positioned such that diagonally-opposite vertices are at the origin and (1,1,1) with the faces aligned with global coordinate planes. The mesh consists of thirty tetrahedral elements that are arranged to give arbitrary alignment and shapes. For simplicity, Figure 3-23 shows the surface of the mesh (i.e., excluding interior nodes and edges). Figure 3-24 depicts the mesh without hidden lines – a rather tangled web of tets. The mesh depicted is for tet4 elements and has 5 internal nodes. The element geometry for the tet10 meshes is the same. The tet10 mesh differs by adding mid-edge nodes.

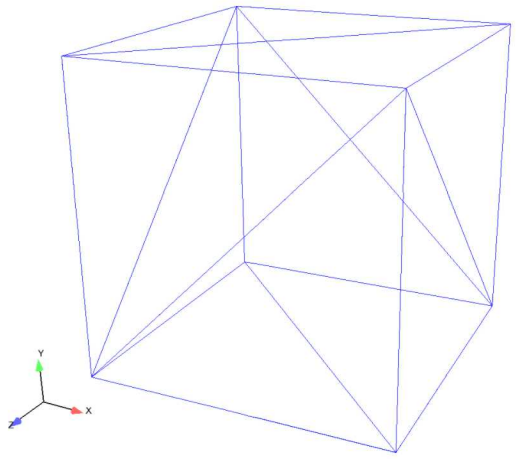


Figure 3-23. Patch-test cube showing only surface faces and edges.

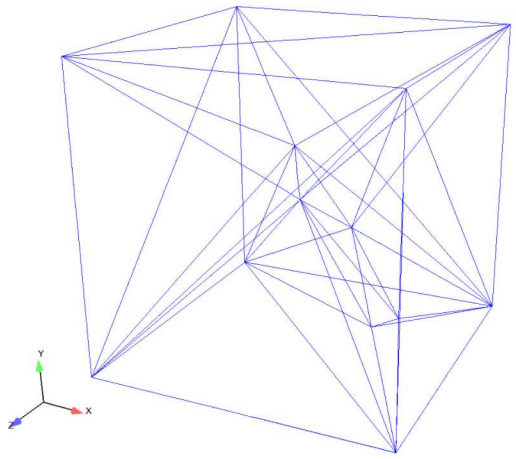


Figure 3-24. Patch-test cube showing all element edges.

The prescribed displacement field on the surface of the cube is given by:

$$u = t \times (1.0 \times 10^{-4}) \times (2x + y + z) \tag{3.47}$$
$$v = t \times (1.0 \times 10^{-4}) \times (x + 2y + z) \tag{3.48}$$
$$w = t \times (1.0 \times 10^{-4}) \times (x + y + 2z) \tag{3.49}$$

where t denotes time, u denotes the displacement in the x-direction, v denotes the displacement in the y-direction, and w denotes the displacement in the z-direction. The units for the displacement components are inches. The corresponding analytic functions in the input file are also labeled as u, v and w, respectively.

3.7.1.3. Material Model

The material models are elastic with the properties given below.

Young's Modulus	E	1.0×10^6 psi
Poisson's Ratio	ν	0.25

3.7.2. Expected Results

These tests assume that the displacements and strains will be sufficiently small for linear elasticity to provide a reasonable reference solution. For infinitesimal strains, the strain-displacement relations of linear elasticity give the strains at t=1 as:

$$\epsilon_{xx} = \epsilon_{yy} = \epsilon_{zz} = 2 \times 10^{-4} \tag{3.50}$$

and

$$\epsilon_{xy} = \epsilon_{yz} = \epsilon_{zx} = 1 \times 10^{-4} \tag{3.51}$$

Note that the size of sufficiently small depends upon the particular material model. Both the elastic model (a hypoelastic model) and the neo-Hookean model (a hyperelastic model) are used for these patch tests. For infinitesimal strains, responses from both constitutive models reduce to that of a linear elastic model, where the stress σ_{xx} is related to the strains ϵ_{xx}, ϵ_{yy}, and ϵ_{zz} by:

$$\sigma_{xx} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \left[\epsilon_{xx} + \frac{\nu}{(1-\nu)} (\epsilon_{yy} + \epsilon_{zz}) \right] \tag{3.52}$$

Similar equations hold for σ_{yy} and σ_{zz}. The shear stress τ_{xy} is related to the shear strain ϵ_{xy} by:

$$\tau_{xy} = \frac{E}{(1+\nu)} \epsilon_{xy} \tag{3.53}$$

Similar equations hold for τ_{yz} and τ_{zx}.

The stress field produced by the above strain field at t=1 is given by:

$$\sigma_{xx} = \sigma_{yy} = \sigma_{zz} = 400\, psi \tag{3.54}$$

and

$$\tau_{xy} = \tau_{yz} = \tau_{zx} = 80\, psi \tag{3.55}$$

For all the tetrahedral element patch tests, the error in each normal and shear stress component was examined, where the error was defined as the infinite norm (maximum over all elements) of the differences between the linear elastic reference solution and the calculated results. The errors are examined for t=1 using 2 equal time steps. Based upon results for the tet4 element, and a few different numbers of time steps, the results did not appear to change with the number of time steps ranging from 1 to 10.

For the hypoelastic (elastic) and hyperelastic (neo-Hookean) material models the solution verification requirement was that each element have errors in each stress component of less than 0.1 percent – an indistinguishable difference on a plot. The results for the two different elastic models differed in the fourth digit of the results. All the element types failed the test at an error tolerance of 0.01 percent. As such, the results only reproduced the linear elastic solution to three digits. The differences with reference solution are not a result of inaccurate computations but rather are a result of limited accuracy in the linear elastic reference solution. A better measure of the accuracy of the computations is provided by the finite deformation versions of these tests, where the reference solutions are based upon the same finite deformation relationships as the code. To test the assertion that the linear elastic reference solution is the issue, we reduced the displacements by one order of magnitude. The resulting computational results agreed with the linear elastic reference solution in one additional digit. To further verify this assertion we symbolically solved the finite deformation equations for the neo-Hookean model (reference the finite deformation version of the test) and found that the finite deformation solution differs with the linear elastic solution in the fourth digit too.

3.7.3. References

1. Fung, Y.C. **Foundations of Solid Mechanics**. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1965.

For input deck see Appendix B.20.

3.8. TET CONVERGENCE TEST – CANTILEVER BEAM

Analysis Type	Quasi-statics
Element Types	Tet4
Element Formulations	Mean Quadrature
Strain Incrementations	Strongly Objective, Node-based
Material Models	Elastic
Verification Category	Convergence
Verification Quantities	Displacement, Stress Fields
Number of Tests	2
Keywords	Linear Elastic, Beam Theory

3.8.1. Brief Description

This set of analyses demonstrates the convergence of displacements and stresses for a relatively simple mechanical analysis. The system analyzed is a cantilever beam under unit tip loading. The reference solution is derived from Euler-Bernoulli beam theory. As such, the solution is not exact, and the finite element solution (if sufficiently refined) will deviate from this reference solution. Two different formulations of the Tet4 are currently tested. Higher order Tet elements will be included in the test once they have the capability to integrate surface tractions consistently with the underlying shape functions.

3.8.1.1. Functionality Tested

Primary capabilities:

- The following element formulations:
  - Four-node tetrahedron with the Mean Quadrature formulation and Strongly Objective strain incrementation
  - Four-node tetrahedron with the Mean Quadrature formulation and Node Based strain incrementation
- subjected to the limitations of linear elasticity.

Secondary capabilities:

- prescribed displacement boundary conditions
- resolution of kinematic boundary condition
- user subroutine application of traction boundary conditions

3.8.1.2. Mechanics of Test

The geometry of the general problem is shown in Figure 3-25. The beam is slender, with a length-to-depth ratio of 10, and it has a square cross-section.[3] The long direction of the beam is aligned with the x-axis, and the unit transverse shear loading is applied in the y-direction as shown.

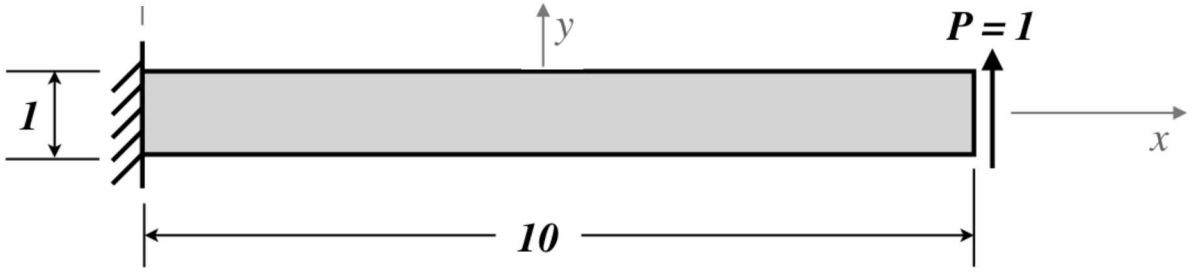


Figure 3-25. Beam geometry.

3.8.1.3. Material Model

The material model used for this problem is the elastic model implemented in Lame [1]. The selected properties are given as follows.

Young's Modulus	E	1.0×10^6 psi
Poisson's Ratio	ν	0.25
Density	ρ	1

3.8.1.4. Boundary Conditions

The boundary conditions for this problem are depicted in Figure 3-26 below. These are not the simplest boundary conditions, but they were defined to be consistent with the shear stress distribution of Euler-Bernoulli beam theory, while still preventing rigid body motions.

3.8.1.5. Meshes

The meshes used in this study are shown in the Figures 3-27-3-29 below. These meshes consist of tetrahedral finite elements, in densities ranging from 2 to 8 elements through the beam depth. These meshes were created using a mesh generation feature that subdivides hexahedral (hex) elements into tetrahedral (tet) elements; the unstructured approach used here subdivides each hex into 6 tet elements and retains the same number of nodes. Retaining the same number of nodes is important for comparison with results from hex elements, though for Sierra/SM those types of comparisons are not valid for other reasons (to be discussed below).

[3] Units are consistent with those of length and force depicted in Figure 3-25 but not explicitly defined.

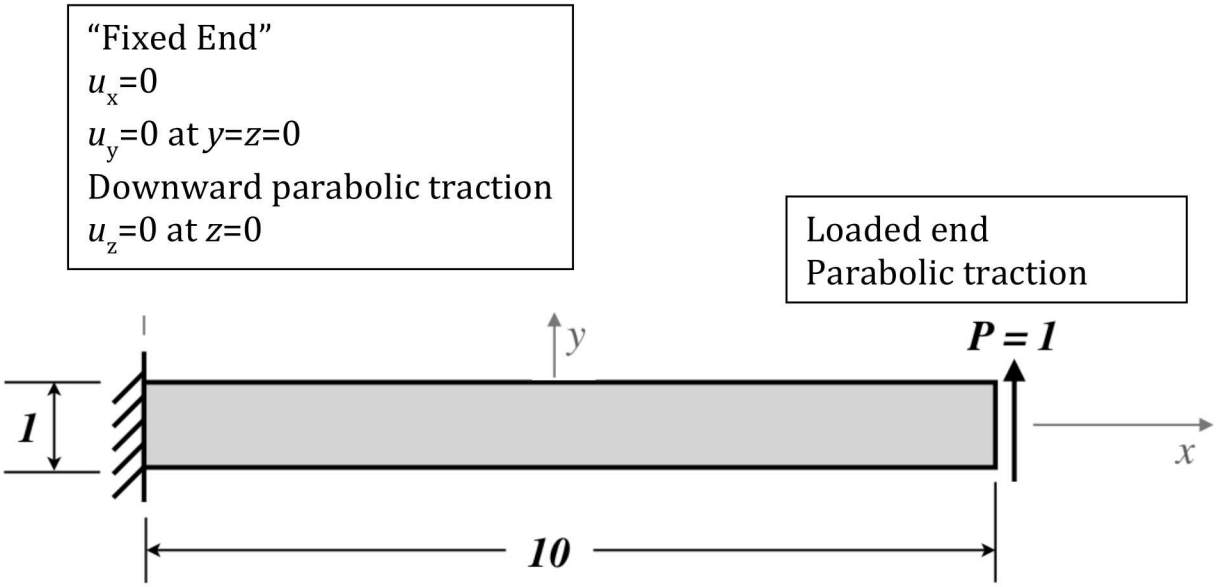


Figure 3-26. Beam boundary conditions.

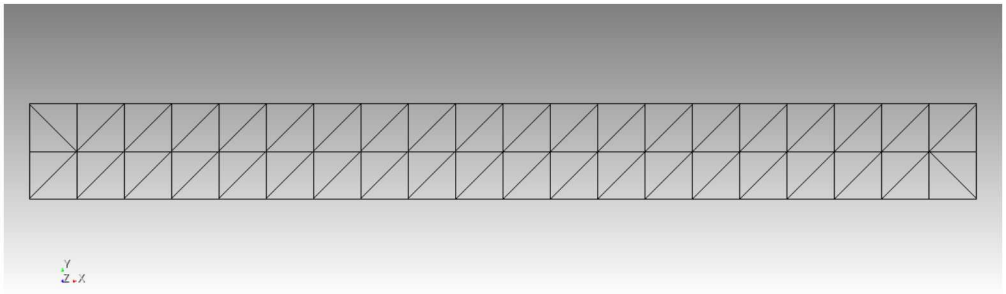


Figure 3-27. 20×2×2 mesh (d/h=2).

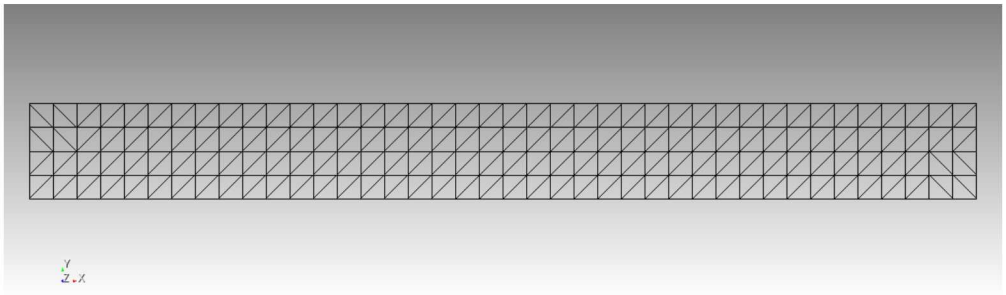


Figure 3-28. 40×4×4 mesh (d/h=4).

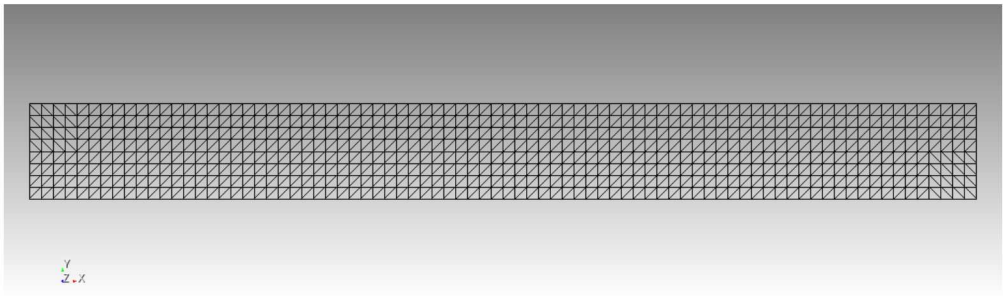


Figure 3-29. 80×8×8 mesh (d/h=8).

3.8.2. Expected Results

As previously mentioned, the reference solution for this problem is obtained from Euler-Bernoulli beam theory. As such, it not only is based upon the approximation of linear elasticity but also has the kinematic constraint that plane sections perpendicular to the neutral reference surface (axis in the one-dimensional idealization) remain planner and perpendicular to the neutral axis under deformation. For brevity the expressions for the displacement and stress field are not formally presented here, but both are presented as string functions in the associated Encore input file. In brief, solving the fourth order differential equation of beam theory gives a vertical displacement that is cubic in x. The longitudinal displacement can be obtained from the rotation (combined with the kinematic assumption) or integrating the longitudinal strain from the fixed-end of the beam, and the transverse displacement can be obtained from integrating the transverse normal strain (ϵ_{zz}). Beam theory gives a linear distribution of the normal stress with respect to y, and a quadratic distribution of shear stress (σ_{xy}) with respect to y. Clearly the existence of the shear stress with no corresponding shear deformation reflects the approximations made in beam theory, approximations that while sufficiently accurate for engineering calculations of slender members conflict with the equations of elasticity. The lack of exactness of this reference solution, weakens its usefulness toward code verification [2], but the convergence tendencies are still apparent. While a higher order beam theory for linear elasticity that includes rotation of sections would be more appropriate for this problem, it would still be inexact relative to the underlying mathematical model of the code (since the code is based upon finite deformations).

3.8.3. Verification Results

In this test we examine the observed rates of convergence using the displacement vector and stress tensor [4] fields. In both cases we use a relative error measure of the norms of the errors divided by the norms of the Euler-Bernoulli reference response (versus the element size).

The slopes of the relative error curves between the data points corresponding to two meshes (on the log-log plots) yield observed rates of convergence. For an exact reference solution, the assessment of the rate of convergence improves with mesh refinement, assuming other sources of numerical error (e.g., solver accuracy) do not corrupt the results; this follows from each mesh refinement producing results that more accurately represent the asymptotic behavior. For this problem we are not using an exact solution, so an improvement in the convergence estimate is not guaranteed. In fact typically for problems without an exact solution there is a sweet range where the approximations are in the asymptotic range but not refined enough to measure the inexactness of the references solution. The following tables give the observed rates of convergence for each Tet4 element formulation between each sequential pair of meshes, where h_{fine} denotes the element size of the finer mesh of the pair. The following plots show the corresponding graphical representations of the solution differences as a function of the element size.

Figure 3-30 shows the convergence results for the displacement field for the Tet4 element

[4] L_2-norms are used for all the norm calculations. The integration of the norm calculations are approximated by Gauss quadrature over each element domain [2,3]. For the stress tensor, the norm is a vector norm of the Voigt notation representation of the stress tensor.

formulations. The asymptotic rates are not clearly obtained for this problem. The regular Tet4 element appears to be approaching quadratic convergence for the two mesh pairs, while the rate of convergence appears to be higher than quadratic for the nodal based tet. Since the reference solution is not exact, we can't make a strong statement about the convergence, but for this level of test the results appear to be reasonable.

Figures 3-31 and 3-32 depict the convergence results for the element and nodal representations of the stress field, respectively, for both Tet4 element formulations. The results for the element stress fields for the regular Tet4 element appear to be approaching the optimal rates of convergence (linear), and as expected the element fields for the nodal based tet are useless. For the regular Tet4, the results for the nodal stress field show improvements in the observed rates of convergence, and for the finest mesh pair yield results very close to those of the nodal based Tet4. These results give a measure of verification for the elements for a linear elastic BVP but are less rigorous than an error quantification test (based upon an exact solution).

Table 3-15. Convergence rates for Tet4: Mean Quadrature - Strongly Objective

h_{fine}	$\|\|U_{diff}\|\|_2/\|\|U_{analyt}\|\|_2$	$\|\|\sigma_{nodal\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$	$\|\|\sigma_{element\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$
0.2500	1.3265	1.2516	0.5752
0.1250	1.9524	1.7158	0.8747

Table 3-16. Convergence rates for Tet4: Mean Quadrature - Node Based

h_{fine}	$\|\|U_{diff}\|\|_2/\|\|U_{analyt}\|\|_2$	$\|\|\sigma_{nodal\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$	$\|\|\sigma_{element\,diff}\|\|_2/\|\|\sigma_{analyt}\|\|_2$
0.2500	2.5478	1.6733	0.0973
0.1250	2.9205	1.7279	0.0253

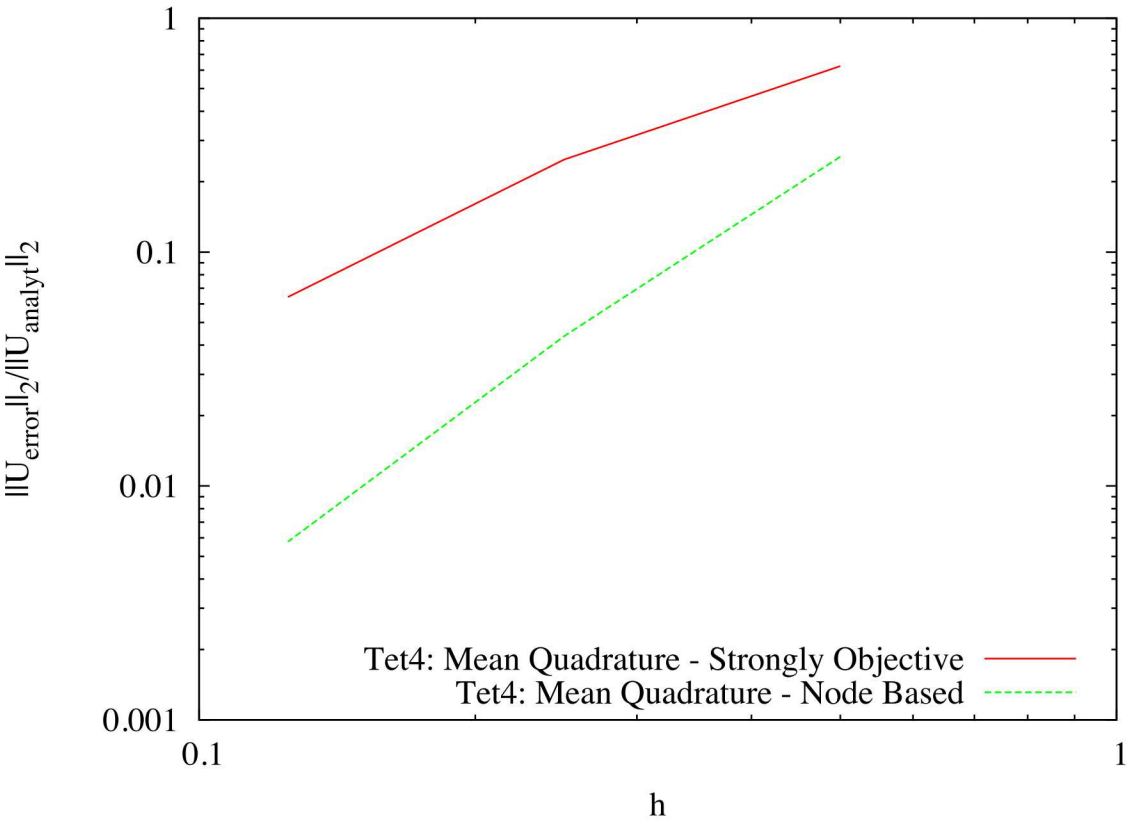


Figure 3-30. Convergence of the displacement vector in the L_2 norm – solution difference versus element size.

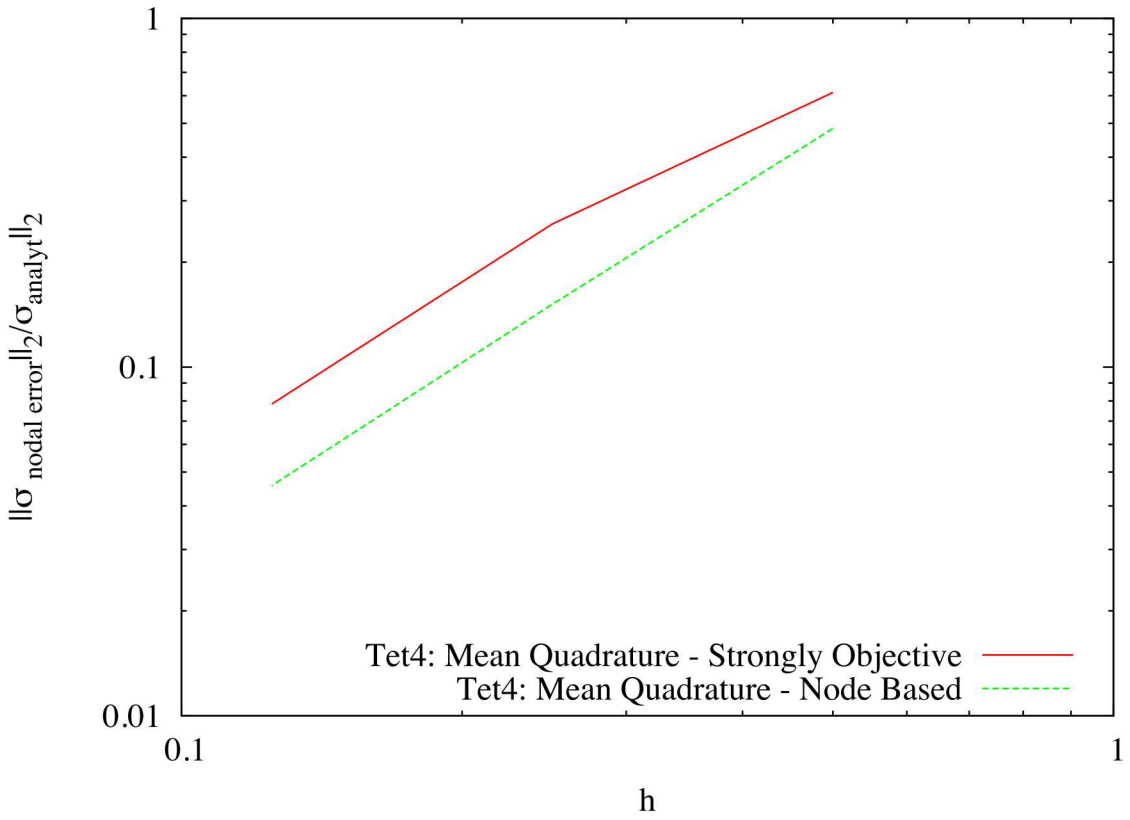


Figure 3-31. Convergence of the nodal representation of the Cauchy stress tensor in the L_2 norm – solution difference versus element size.

Summary of results: The regular Tet4 element showed trends toward exhibiting the expected rates of convergence for displacements (quadratic), but the inexactness of the reference solution (as with the hex elements) appears to affect the quality of the convergence results. For the nodal based tet, the displacement convergence rate is even less clear, but in this case appears to be closer to cubic convergence than quadratic. The element stresses for the regular Tet4 element and finest mesh pairs exhibited a rate of convergence approaching the expected asymptotic rate, with an observed rate of 0.87. The nodal results (one by formulation and one by stress recovery) yield improved rates of convergence of approximately 1.72. Since the reference solution is not exact, even for linear elasticity, this is a weaker convergence test, as discussed in the manual's introduction.

3.8.4. References

1. Scherzinger, W.M. and Hammerand, D.C. *Constitutive Models in Lame*. Sandia Report SAND2007-5873, September 2007.
2. Cox, J.V. and Mish, K.D. *Sierra solid mechanics example verification problems to highlight the use of Sierra verification tools*, October 2012 (in publication).
3. Copps, K.D. and Carnes, B.R. *Encore User Guide*, Sandia Report, October 2009.

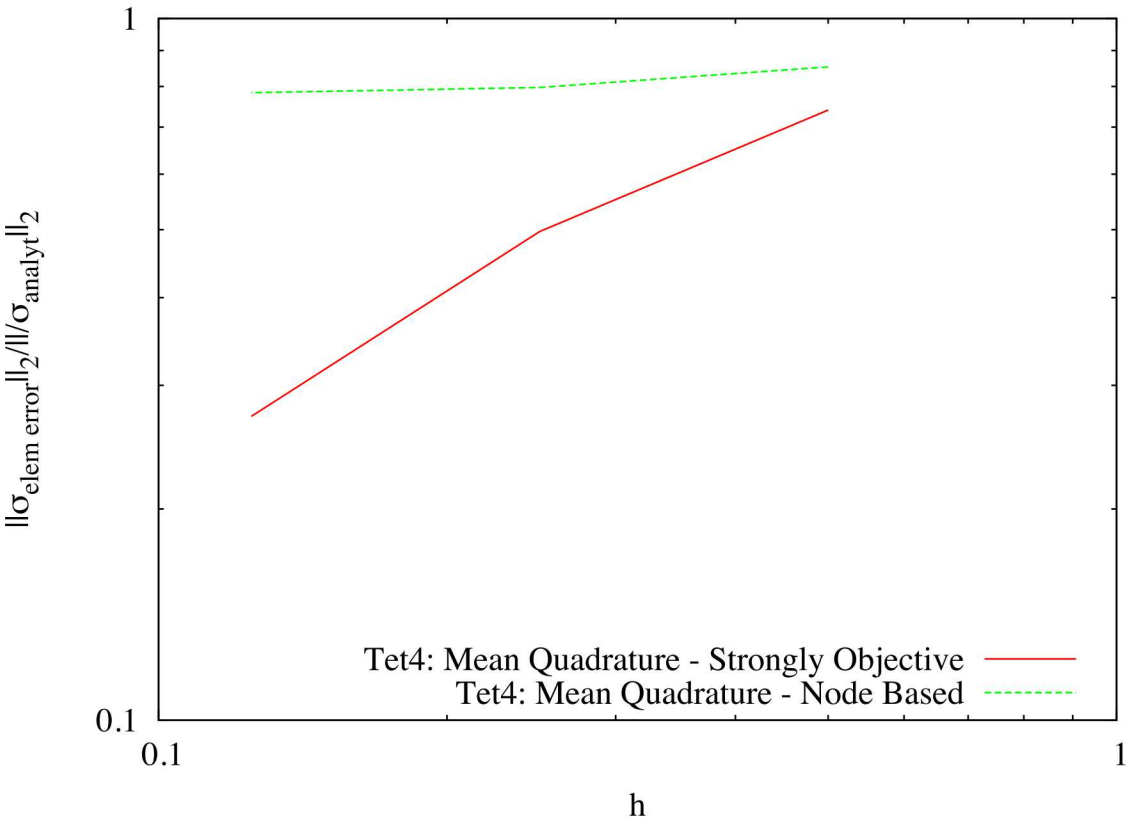


Figure 3-32. Convergence of the element representation of the Cauchy stress tensor in the L_2 norm – solution difference versus element size.

For input deck see Appendix B.21.

3.9. QUAD MEMBRANE PATCH TEST – SELECTIVE DEVIATORIC, MIDPOINT INCREMENT

Analysis Type	Explicit Dynamics
Element Type	Quad4 Membrane
Strain Incrementation	Midpoint Increment
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Displacement, Stress
Number of Tests	1
Keywords	Patch Test

3.9.1. Brief Description

This problem is a patch test for a selective deviatoric, four-node membrane element with a midpoint-increment formulation for the strain. This test is described in Reference 1.

3.9.1.1. Functionality Tested

Primary functionality tested:

- selective deviatoric, four-node membrane element with midpoint-increment strain formulation

Secondary capabilities:

- prescribed displacement direction boundary condition
- linear elastic material model

3.9.1.2. Mechanics of Test

The test consists of five midpoint-increment membrane elements arranged in a 0.12 *inch* $\times$ 0.24 *inch* planar rectangle. Four of the elements have edges along the outside edges of the rectangle, and one element has edges completely internal to the rectangle. The fifth element has skewed edges inside the plane such that no two edges of an element are parallel.The mesh for this problem is shown in Figure 3-33.

The bottom left corner of the rectangular domain is fixed. The other three corners of the rectangle have prescribed displacements. Their values at time 2.0×10^{-3} sec are

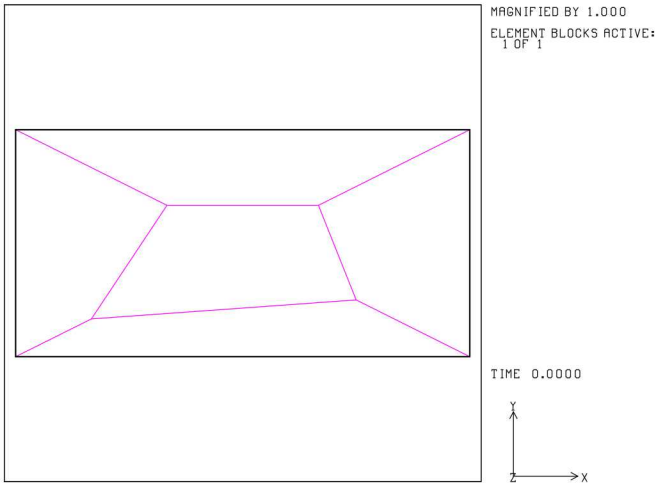


Figure 3-33. Mesh for patch test.

$$U = 1.0 \times 10^{-3}(x + \frac{y}{2}) \tag{3.56}$$

$$V = 1.0 \times 10^{-3}(\frac{x}{2} + y) \tag{3.57}$$

$$W = 0 \tag{3.58}$$

where U is the displacement in the x-direction, V is displacement in the y-direction, and W is displacement in the z-direction. Displacements are constrained to the xy-plane. Also, the x and y variables within the U and V equations correspond to the location of each node.

The loading is ramped up from 0 to 2.0×10^{-3} sec and then held constant for time greater than 2.0×10^{-3} sec. For $t \leq 2.0 \times 10^{-3}$:

$$u(t) = U \times 0.5(1 - cos(\frac{\pi t}{2.0 \times 10^{-3}})) \tag{3.59}$$

For $t > 2.0 \times 10^{-3}$:

$$u(t) = U \tag{3.60}$$

3.9.1.3. Material Model

The material model is linear elastic with the values:

Young's Modulus	E	1.0×10^6 psi
Poisson's Ratio	ν	0.25
Density	ρ	2.61×10^{-4} lbm/in^3

3.9.2. Verification of Solution

This loading applies a spatially constant strain field to the elements, assuming a quasi-static rate of loading (i.e., without significant inertia effects). For the small strain case the "exact" stress field produced by the displacement field is:

$$\sigma_{xx} = \sigma_{yy} = 1333\, psi \tag{3.61}$$

$$\sigma_{zz} = 0\, psi \tag{3.62}$$

$$\tau_{xy} = 400\, psi \tag{3.63}$$

The following plots are the stresses from element 1, although elements 1 through 5 should have the exact same stresses. Figure 3-34 depicts the time history of the in-plane normal stress components, and Figure 3-35 depicts the time history of the in-plane shear stress component. The normal and shear stress components agree with the exact solution to 4 digits. Note that lowering the loading level could increase the accuracy of this result, since it may be due to the code's formulation for finite deformations.

3.9.3. References

1. MacNeal, R. H., and R. L. Harder. A Proposed Standard Set of Problems to Test Finite Element Accuracy. *Finite Elements in Analysis and Design* 1 (1985): 3-20.

For input deck see Appendix B.22.

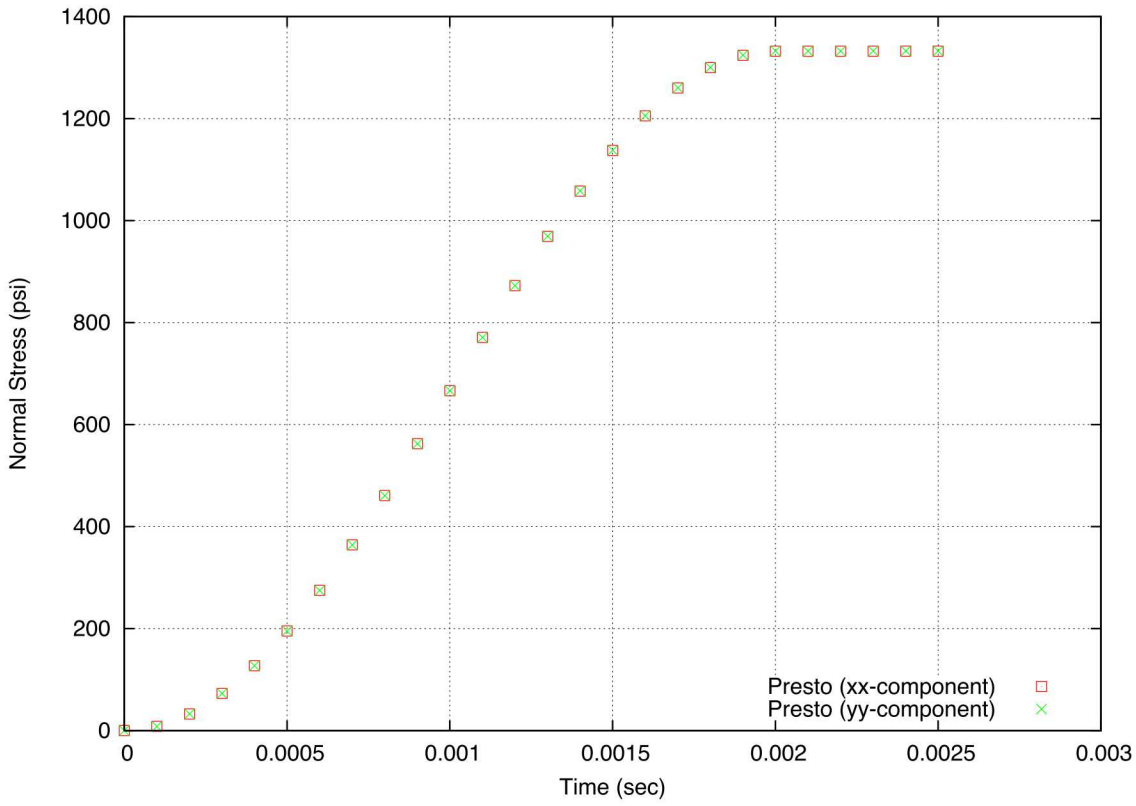


Figure 3-34. Stress xx- and yy-components for element 1.

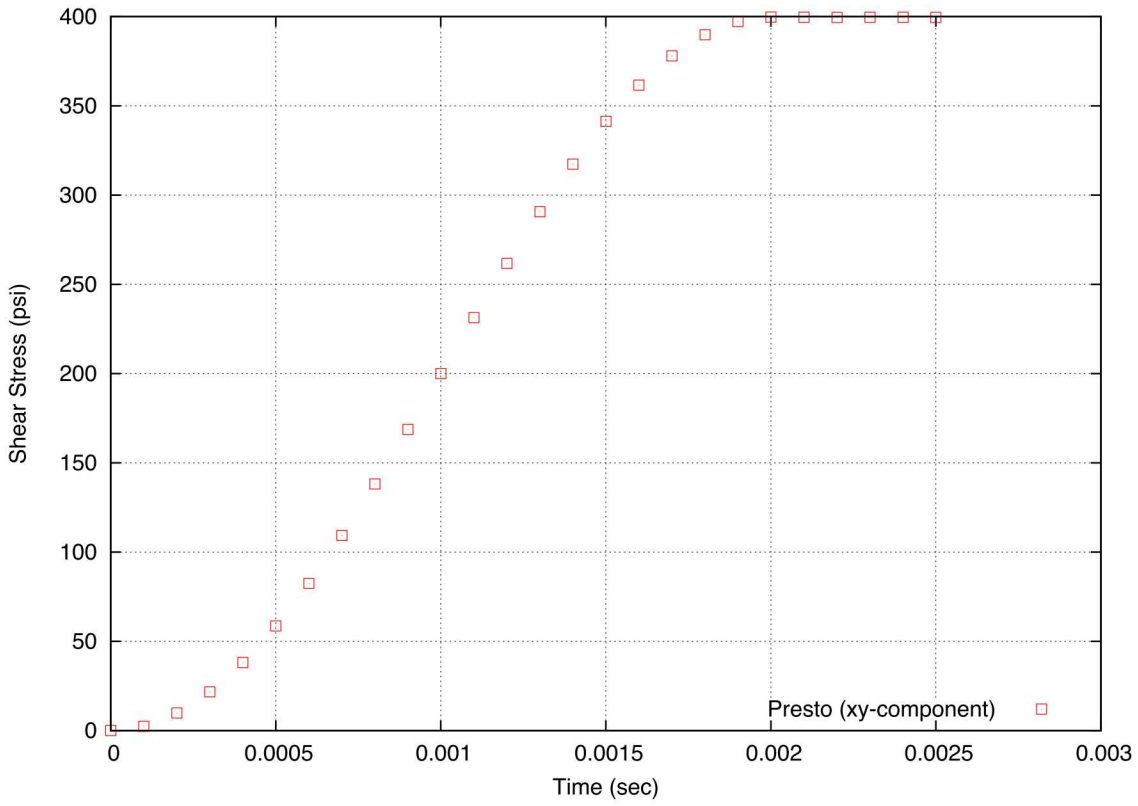


Figure 3-35. Stress xy-component for element 1.

3.10. ELASTIC BEAM IN AXIAL TENSION

Analysis Type	Quasi-statics
Element Type	Beam
Material Model	Elastic-Plastic
Verification Category	Discretization Error
Verification Quantities	Axial Reaction Force
Number of Tests	13
Keywords	Beam Section, Elastic, Axial

3.10.1. Problem Description

This problem puts several beam sections through strain paths that exercise elastic section response under pure axial extension. The elements used in this analysis are 2D beams. Note the actual computational elements are two-noded, single elements.

All beams consistent of a single element and are of a total length of five (5) meters along the *r*-direction.

3.10.1.1. Boundary Conditions

The beam end conditions are prescribed. Axial extension is accomplished by fixing one end of the beam and extending the far end in the *r*- direction. This produces a constant axial strain state in the beam.

Deformation magnitudes are kept low to enable the code results to match the linear elastic solution for a one-dimensional axial member. The applied displacement magnitude is selected so that the section remains elastic throughout the loading history.

Loading Condition	Boundary Condition
Axial Extension	d-dz/dL = 4.0e-6

3.10.1.2. Material Model

Material is elastic perfectly plastic. Yield stress of the material is set to a relatively low value, similar to general aluminum. This a small deformation to remain in the elastic region to best match small strain beam theory.

3.10.1.3. Feature Tested

Behavior of beam sections in elastic regime under axial tension loading.

Young's Modulus	E	$3.0e+3$
Shear Modulus	G	$1.15384615e+3$
Yield Stress	$y0$	290
Hardening Modulus	h	0.0

3.10.2. Verification of Solution

Analytical values were used to determine the cross sectional area and the reactive force developed during a linear loading process to a prescribed displacement value. The free ends of the beams are linearly displaced to the prescribed value over 20 load steps and then held constant for the remainder of the test.

The actual results verified against analytic results in the input file are the response quantities throughout the loading history (presumably fully elastic regime).

Table 3-17 provides the upper bound for the errors between the computed and analytic solutions for all beam sections in the elastic regime. Full load history results can be found in Figures 3-36, 3-37, and 3-38.

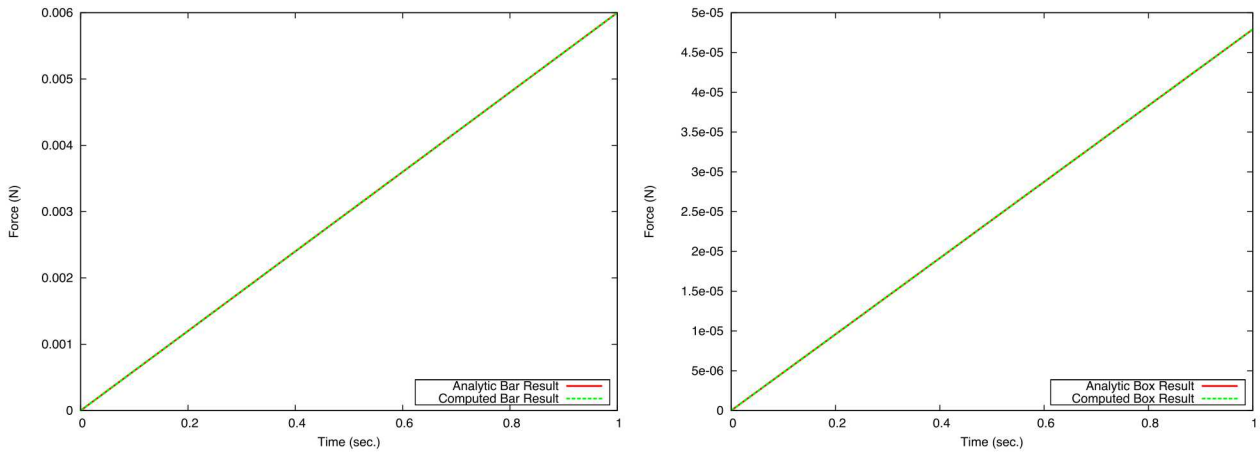
Table 3-17. Percent difference between computed and analytic solution.

Loading	For All Beams
Axial Elastic	<0.002

3.10.3. Conclusions

Result for axial extension of all sections is nearly exact. This is due to axial resistance being a function of section area, which is explicitly computed from section properties.

For input deck see Appendix B.23.



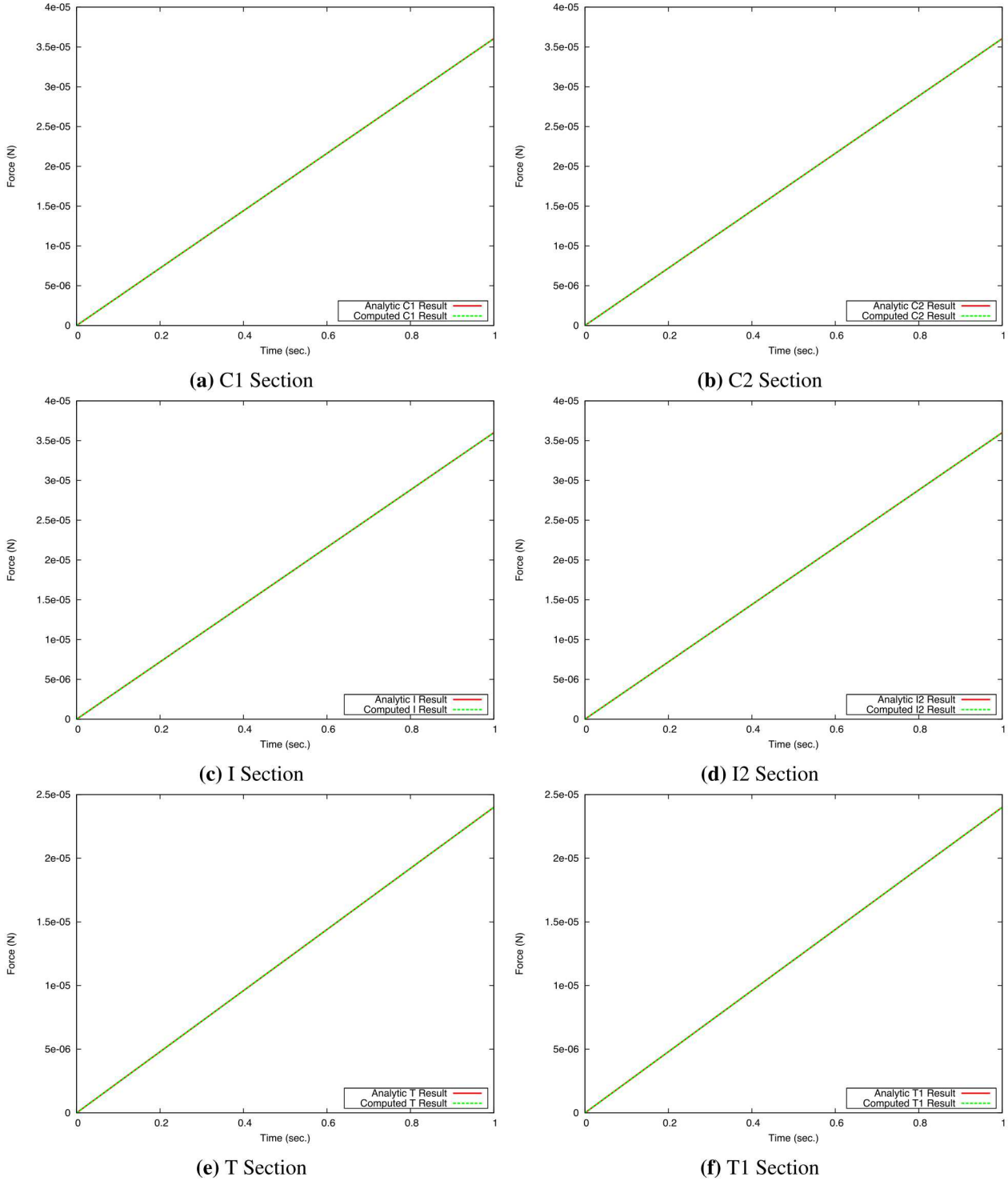
(a) Bar Section

(b) Box Section

(c) Rod Section

(d) Tube Section

Figure 3-36. Bar, box, rod, and tube section results: Axial load *vs* time.

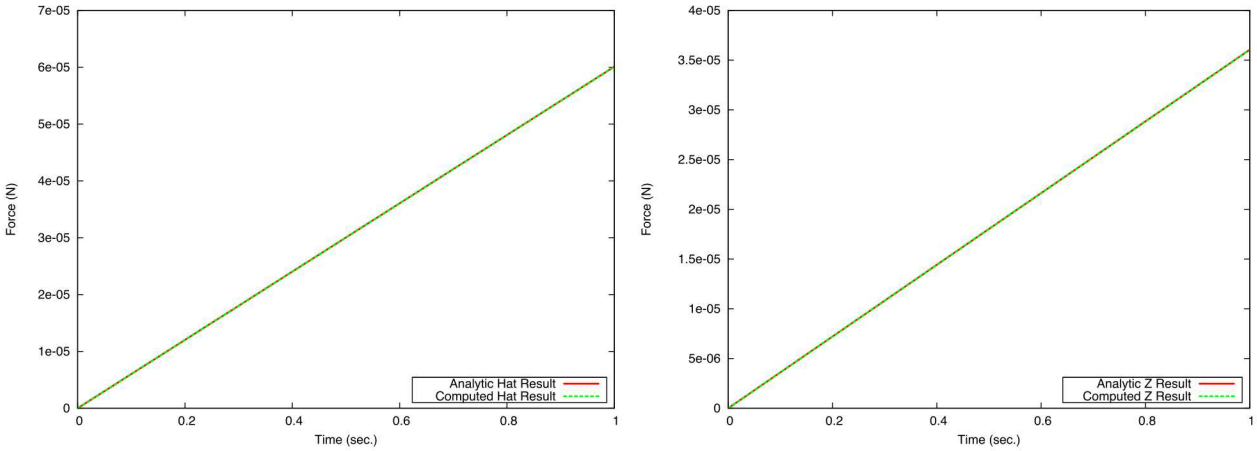


(a) C1 Section **(b)** C2 Section

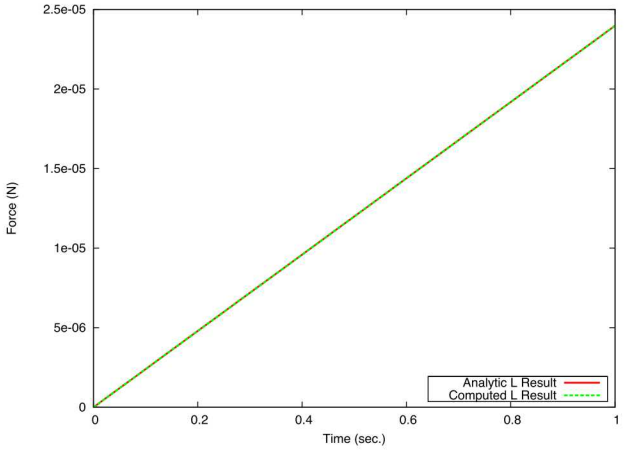
(c) I Section **(d)** I2 Section

(e) T Section **(f)** T1 Section

Figure 3-37. C, I, and T section results: Axial load *vs* time.



(a) Hat Section **(b)** Z Section



(c) L Section

Figure 3-38. Hat, Z, and L section results: Axial load *vs* time.

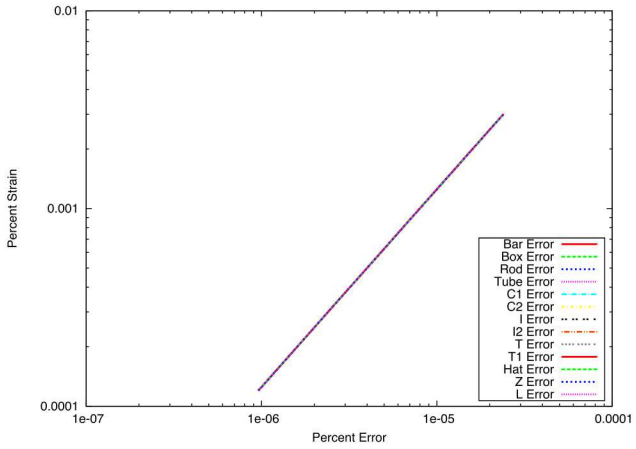


Figure 3-39. Log-Log plot of Strain vs Error

3.11. ELASTIC BEAM IN BENDING

Analysis Type	Explicit Transient Dynamics
Element Type	Beam
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Bending, Torsional Stiffnesses
Number of Tests	1
Keywords	Beam Section, Elastic, Bending

3.11.1. Problem Description

This problem puts several beam sections through strain paths that exercise the elastic response of the beams under pure bending and pure torsion. The elements used in this analysis are 2D beams. The lofted geometry of the beam sections is shown in Figures 3-40 and 3-41.

Note the actual computational elements are two noded line elements. The geometry shown in Figures 3-40 and 3-41 was produced by plotting supplemental lofted surface geometry on top of the line elements.

The top row of beams is subjected to uniform bending about the Y (t) axis. The second row of beams is subjected to uniform bending about the Z (s) axis . The third row of beams is subjected to uniform torsion about the X (r) axis.

3.11.1.1. Boundary Conditions

The beam end conditions are prescribed. For bending one end of the beam is fixed and other end is subjected to a prescribed S or T force. For torsion one end of the beam is fixed and the other end subjected to a prescribed R moment.

Deformation magnitudes are kept low to enable matching code results to small strain beam theory. The applied force and moment magnitudes are selected so that the section remains elastic.

Loading Condition	Boundary Condition
S Bending	F = 1.0
T Bending	F = 1.0
R Torsion	F1 = (1/64)*(1-cos(x*pi/3.2e-2)), F2 = 3.125e-2

3.11.1.2. Material Model

The material is specified as elastic

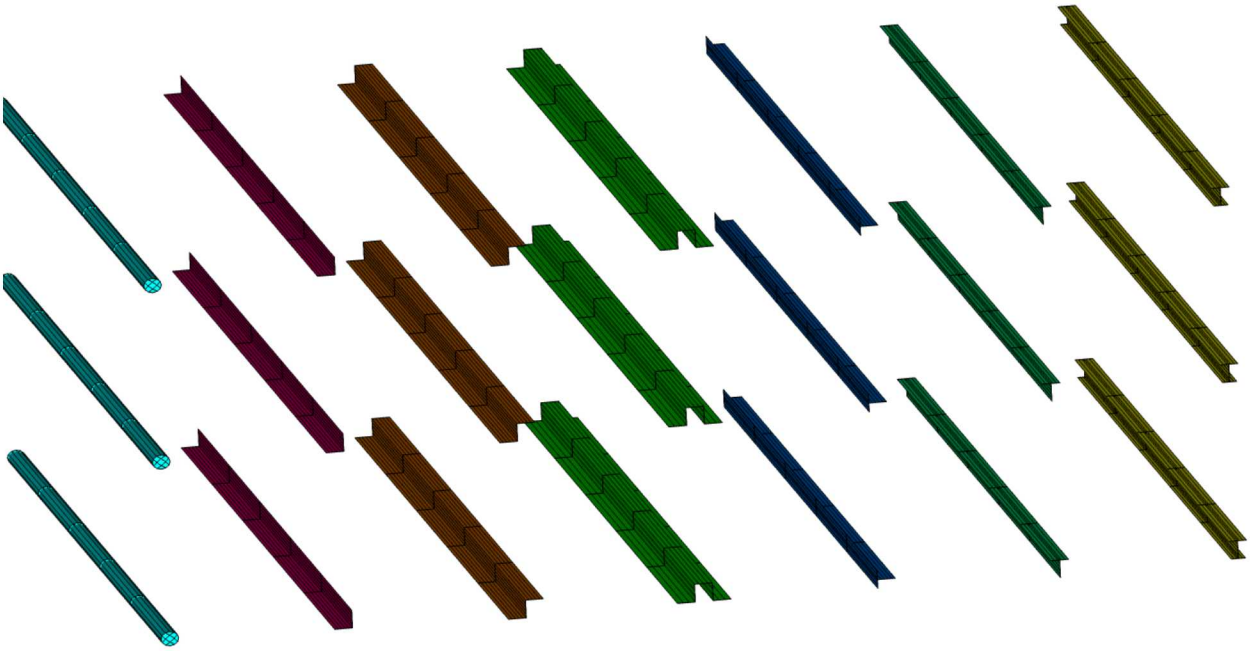


Figure 3-40. Section Representation 1-7

Young's Modulus	E	$10.0e+6$
Poisson's Ratio	v	0.3

3.11.1.3. Feature Tested

Behavior of beam sections in elastic regime.

3.11.2. Verification of Solution

The analytic result for I_x, I_y and I_z is calculated using aprepro and is included in the input file.

Table 3-18 and 3-19 lists the error between computed analytic solution for each beam section in the elastic regime.

3.11.3. Conclusions

Bending results for all sections tends to be decent. Due to a finite number of integration points code results do not precisely track analytic results. Some sections (such as the rod and bar) have integration point locations and weights optimized to provide accurate elastic responses. The Z

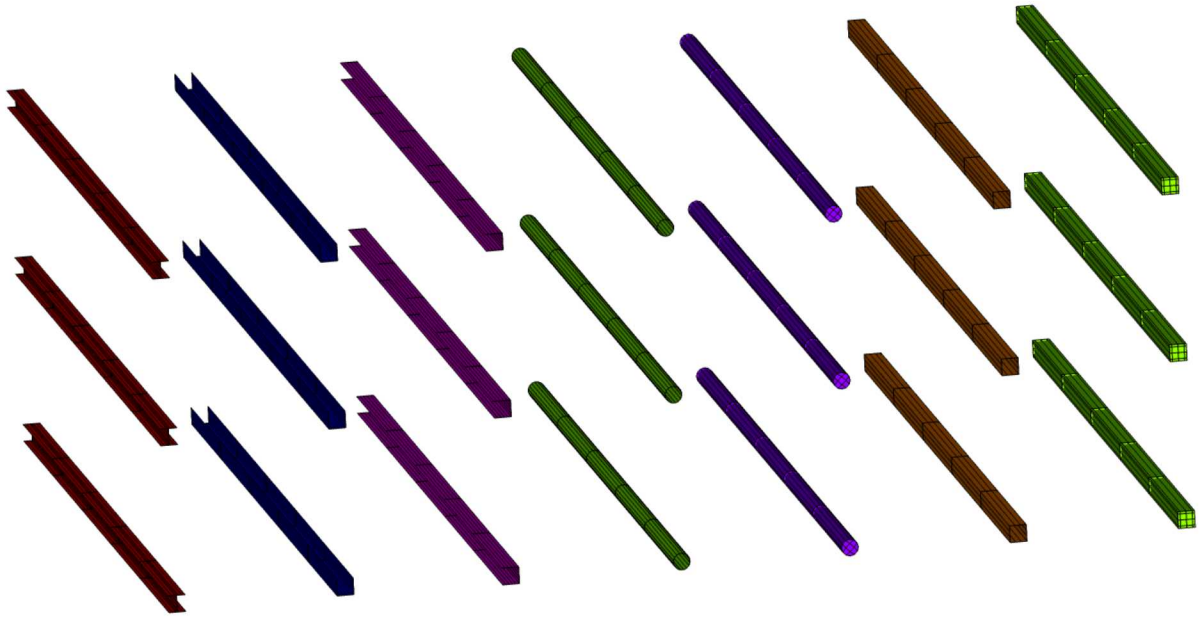


Figure 3-41. Section Representation 8-14

Loading	Bar	Box	Rod	Tube	C1	C2	I
T Bend Elastic	0.56	7.2	0.68	1.7	0.57	1.5	0.008
S Bend Elastic	0.56	7.2	0.68	1.7	1.5	0.57	0.73
R Twist Elastic	16.0	10.5	0.014	21.0	33.0	33.0	33.0

Table 3-18. Percent Difference Between Computed and Analytic Solution

beam and the L beam tend to produce inaccurate bending results. All beam sections excluding the Z and L beam produce an error less than 10 percent of the analytical solution.

Torsion results tend to show the highest deviations from analytic response. The torsional response of the circular rod is good. Torsional response of the solid bar and closed box sections is OK. Compact or closed rectangular sections will undergo a small amount of warping during torsion. Torsional response of the HAT, T, L, Z and I sections are vastly different than the engineering solution.

For input deck see Appendix B.24.

Loading	I2	T	T1	HAT	Z	L	Ellipse
T Bend Elastic	0.56	1.1	3.1	2.0	0.74	1.4	0.68
S Bend Elastic	3.2	3.1	1.1	0.18	0.92	1.2	0.68
R Twist Elastic	30.0	78.0	78.0	67.0	78.5	70.0	3.2

Table 3-19. Percent Difference Between Computed and Analytic Solution

3.12. ELASTIC AND PLASTIC BEAM

Analysis Type	Quasi-statics
Element Type	Beam
Material Model	Elastic-Plastic
Verification Category	Discretization Error
Verification Quantities	Reaction Forces, Moments
Number of Tests	40
Keywords	Beam Sections, Elastic, Plastic

3.12.1. Problem Description

This problem puts several beam sections through strain paths that exercise both the elastic and plastic section response under pure bending, pure torsion, and pure axial extension. The elements used in this analysis are 2D beams. The lofted geometry of the beam sections is shown in Figure 3-42. Note the actual computational elements are two noded line elements. The geometry shown in Figure 3-42 was produced by plotting supplemental lofted surface geometry on top of the line elements.

The top row of beams is subjected to axial extension. The second row of beams is subjected to uniform bending about the X (t) axis. The third row of beams is subjected to uniform bending about the Y (s) axis. The fourth row of beams is subjected to uniform torsion about the Z (r) axis.

All beams consistent of twenty length 0.1 meter elements to make a total beam length of 2 meters.

3.12.1.1. Boundary Conditions

The beam end conditions are prescribed. Axial extension is accomplished by fixing one end of the beam and extending the far end in the r direction. This produces a constant change in length per length of beam. For bending one end of the beam is fixed and other end subjected to a prescribed S or T rotation. This produces a constant bend per length of the beam. For torsion one end of the beam is fixed and the other end subjected to a prescribed R rotation this produces a constant rotation per length of the beam.

Deformation magnitudes are kept low to enable matching code results to small strain beam theory. The applied displacement and rotation magnitudes are selected so that the section remains elastic for roughly the first third of the loading history, transitions from elastic to plastic in the second third of the loading history, and attains nearly the maximum plastic load by the end of the loading history.

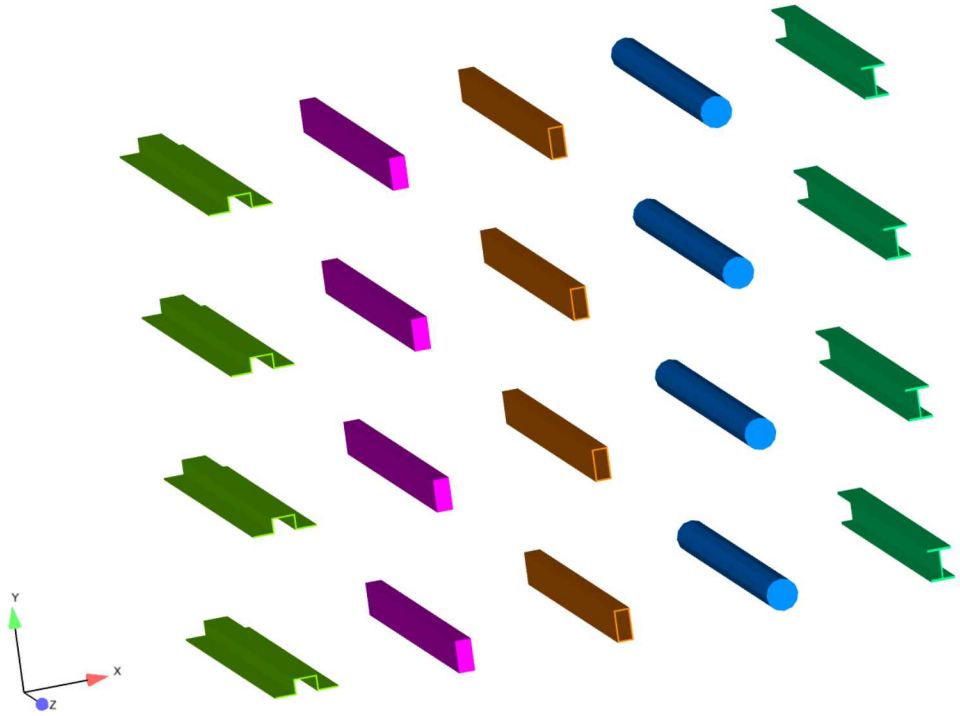


Figure 3-42. Section Representation

Loading Condition	Boundary Condition
Axial Extension	d-dz/dL = 5.0e-6
S Bending	d-thetay/dL = 5.0e-5
T Bending	d-thetax/dL = 5.0e-5
R Torsion	d-thetaz/dL = 5.0e-5

3.12.1.2. Material Model

Material is elastic perfectly plastic. Yield stress of the material is set to a very low value. The low yield stress enables yield of the beam at low total strain to best match small strain beam theory.

Young's Modulus	E	$1.0e+6$
Shear Modulus	G	$5.0e+5$
Yield Stress	$y0$	1.0
Hardening Modulus	h	0.0

3.12.1.3. Feature Tested

Behavior of beam sections in both elastic and plastic regimes.

3.12.2. Verification of Solution

Mathematica code is included in the input file to compute analytic result for the M_{rr}, M_{ss}, M_{tt}, and F_{rr} moments and forces for the different loading cases. The Mathematica code is based on integration of an elastic plastic material model on a beam section using the standard plane sections remain plane assumptions. For the computation of the torsional resistance two analytic results are provided. One analytic result is for torsional resistance is computed assuming no out of plane warping of the beam section. A second analytic result is provided based on common engineering assumptions for the torsional resistance of different beam shapes.

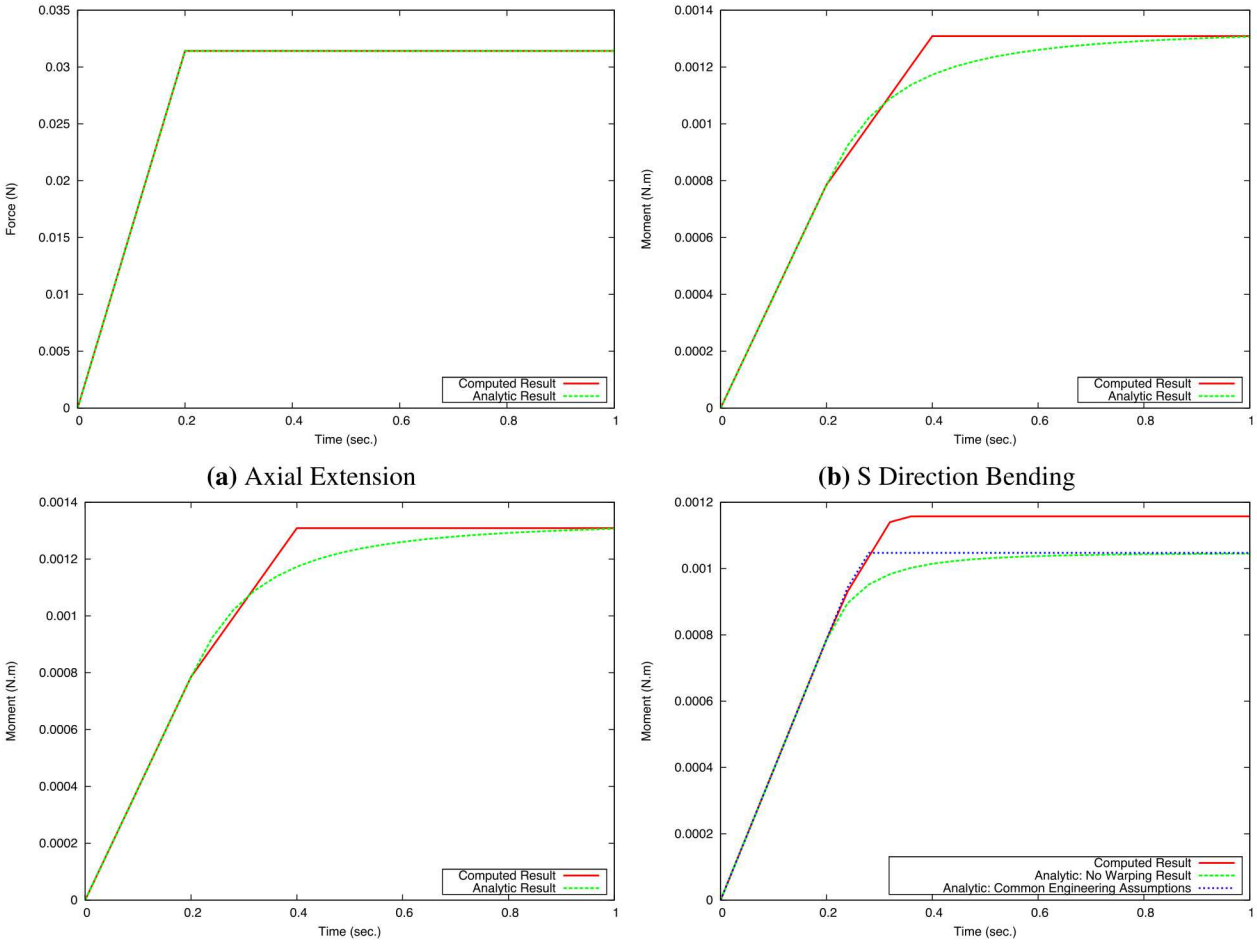
For non-circular sections, particularly unrestrained open sections, there will be substantial out of plane warping during torsion. Out of plane warping of beam sections tends to substantially reduce the beam's capacity to resist torsion. The common engineering assumption applies commonly used correction factors to account for section warping. Description of the torsion correction factors is given in the Mathematica code in the input deck.

The end displacements are applied by a linear ramp over 25 load steps. The actual results verified against analytic results in the input file are the response quantities at the first load step (presumably fully elastic regime) and the response quantities at the last load step (presumably fully plastic regime.)

Table 3-20 lists the error between computed an analytic solution for each beam section in the elastic regime and in the fully plastic regime. Full load history results can be found in Figures 3-43 through 3-47.

Loading	Rod	Bar	Box	Hat	I
Axial Elastic	0.002	0.002	0.002	0.002	0.002
Axial Plastic	0.0005	0.0005	0.0005	0.0005	0.0005
S Bend Elastic	0.02	0.12	5.0	1.00	0.20
S Bend Plastic	0.16	14.0	16.0	0.10	1.0
T Bend Elastic	0.02	0.05	24.0	1.36	0.20
T Bend Plastic	0.16	9.0	9.0	0.36	7.5
R Twist Elastic	0.0015	25.0	91.0	2500.0	9700
R Twist Plastic	10.0	34.0	50.0	1000.0	2400

Table 3-20. Percent Difference Between Computed and Analytic Solution



(a) Axial Extension

(b) S Direction Bending

(c) T Direction Bending

(d) R Direction Torsion

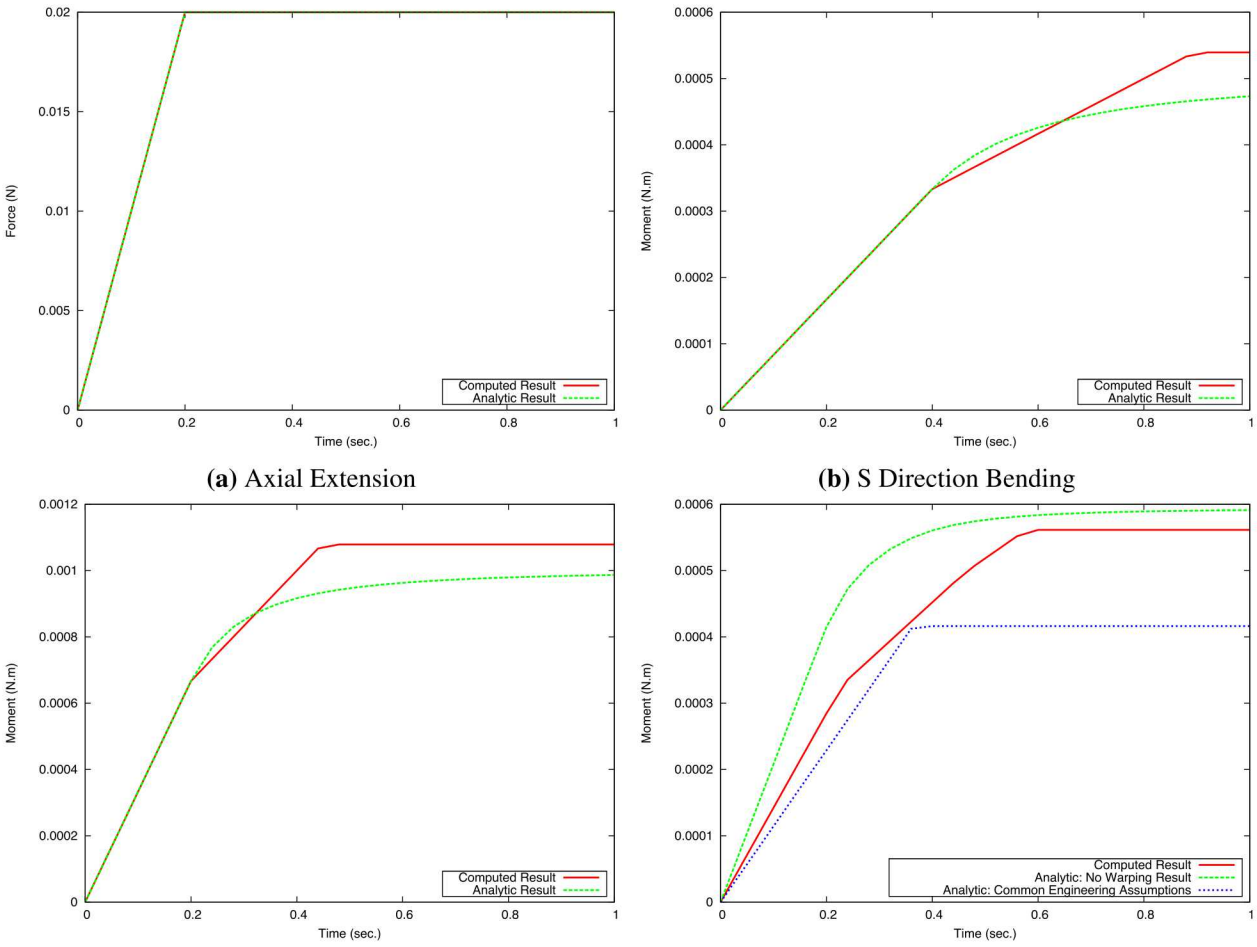
Figure 3-43. Rod Section

3.12.3. Conclusions

Result for axial extension of all sections is nearly exact. This is due to axial resistance being a function of section area, which is explicitly computed from section properties.

Bending results for all sections tends to be decent. Due to a finite number of integration points code results do not precisely track analytic results. Some sections (such as the rod and bar) have integration point locations and weights optimized to provide accurate elastic responses those same weights will be less optimal in the plastic regime. Sections that have the most integration points through the thickness, such as the hat, tend to provide the most accurate bending response in plasticity.

Torsion results tend to show the highest deviations from analytic response. The torsional response of the circular rod is good. A torqued circular rod will have no section warping, this matches both the no warping analytic and engineering solution. Torsional response of the solid bar and closed box sections is OK. Compact or closed rectangular sections will undergo a small amount of warping during torsion. Torsional response of the open hat and I sections is vastly different than



(a) Axial Extension

(b) S Direction Bending

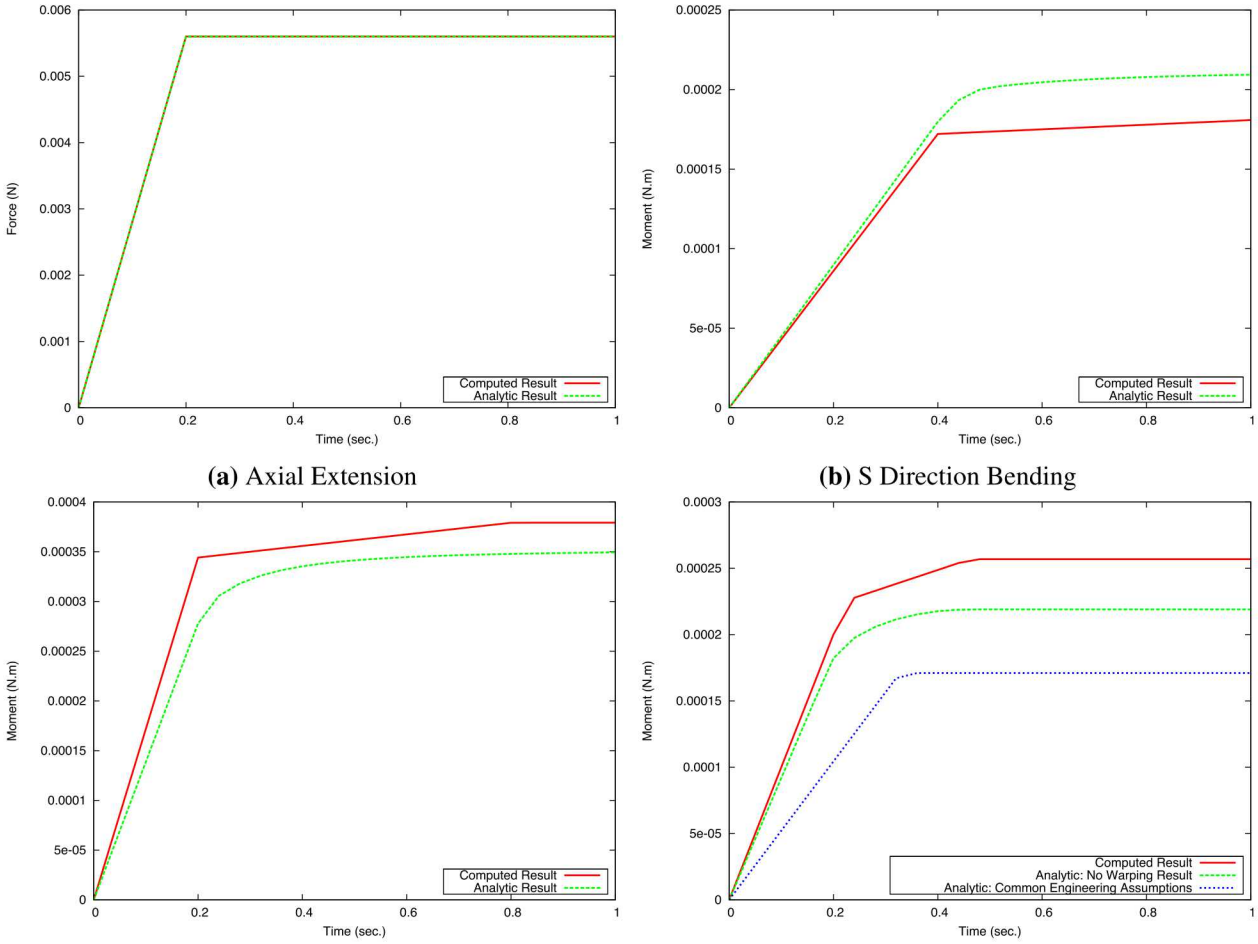
(c) T Direction Bending

(d) R Direction Torsion

Figure 3-44. Bar Section

the engineering solution. The hat section will warp substantially under torsion, the effect of this warping is not fully accounted for in the solution given by the code.

For input deck see Appendix B.25.



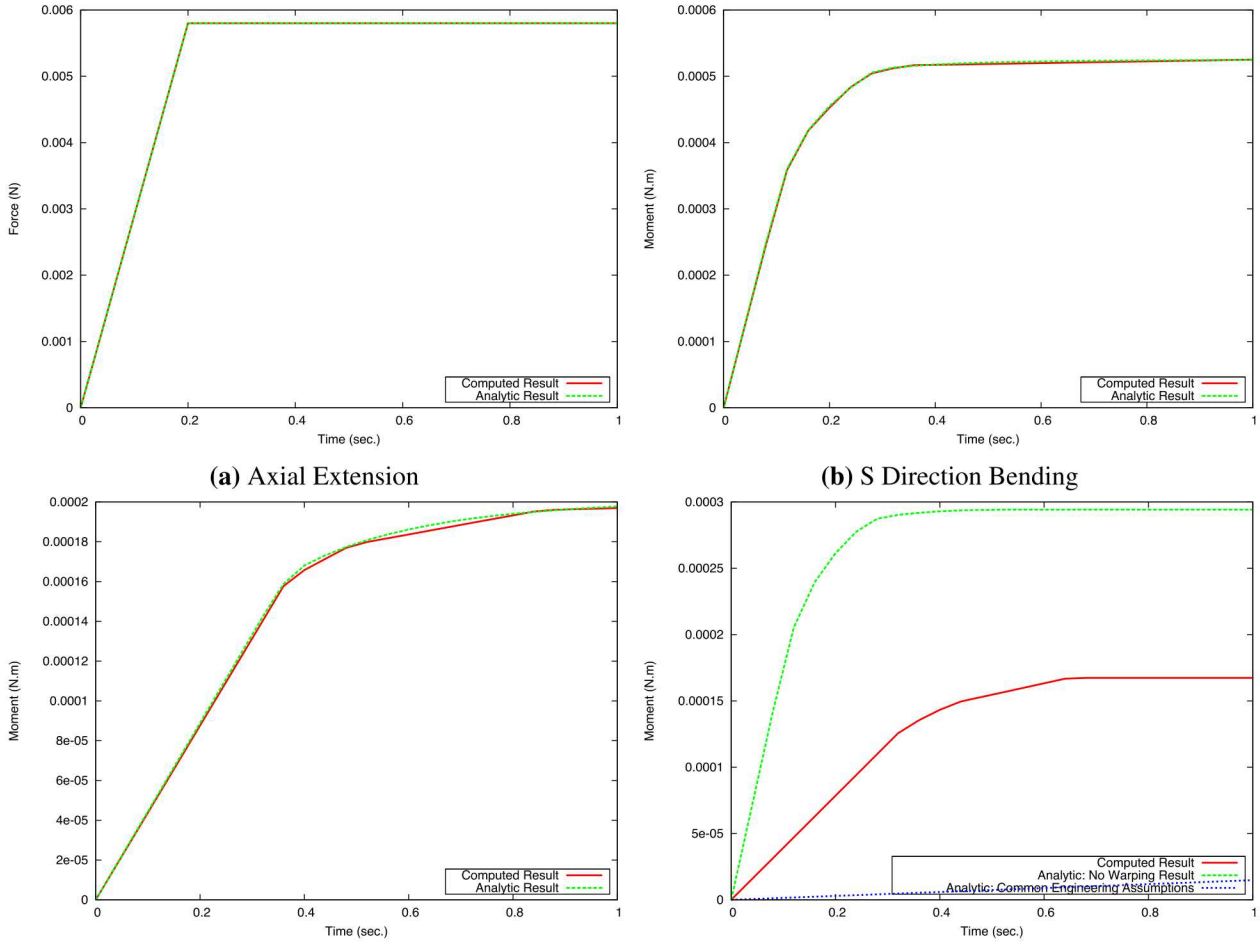
(a) Axial Extension

(b) S Direction Bending

(c) T Direction Bending

(d) R Direction Torsion

Figure 3-45. Box Section



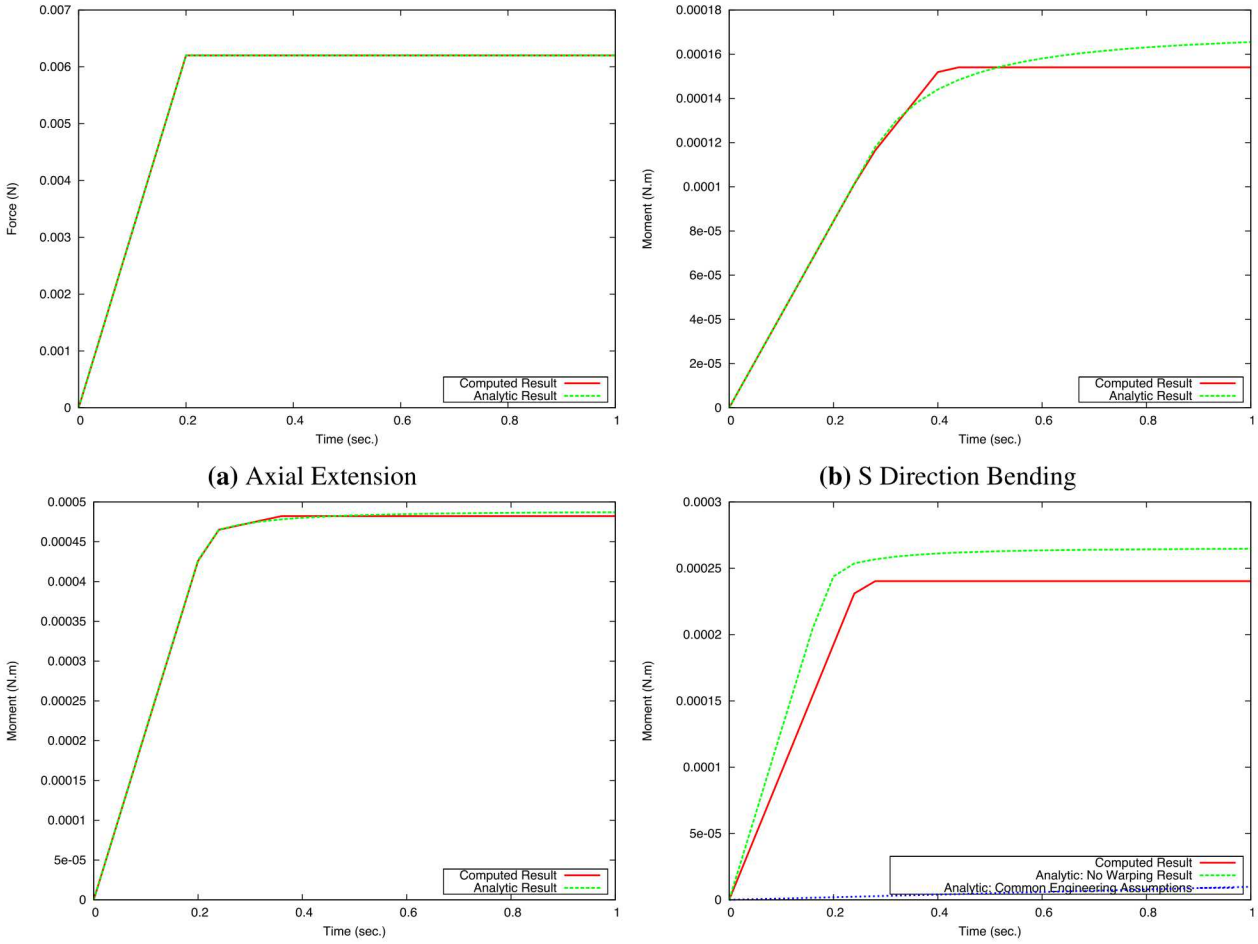
(a) Axial Extension

(b) S Direction Bending

(c) T Direction Bending

(d) R Direction Torsion

Figure 3-46. Hat Section



(a) Axial Extension

(b) S Direction Bending

(c) T Direction Bending

(d) R Direction Torsion

Figure 3-47. I Section

3.13. PRESSURE LOADED LAYERED CANTILEVER BEAM

Analysis Type	Explicit/Implicit Dynamics
Element Type	Shell
Material Model	Elastic
Verification Category	Discretization Error
Verification Quantities	Reaction Moment, Tip Displacement
Number of Tests	4
Keywords	Shell, Layered Shell

3.13.1. Problem Description

This test is a cantilevered beam made of shell elements under uniform pressure loading on the shell surfaces. The load magnitude keeps the problem in small strain, the load is smoothly ramped over time and then held constant to minimize dynamic effects, and the results are compared with Euler-Bernoulli beam theory. The physical problem is a beam of width 1, length 10, thickness 0.125, and applied pressure (at the maximum) of 0.2.

The numerical solution relies on two capabilities: layered shell elements and lofted shells. In the first case a single shell element with two layers is used to model the beam thickness (with two elements across the width and 20 along the length), while in the second case two lofted beam elements, sharing the same nodes, are used to model the beam through the thickness. In each case the elements are layered/lofted to reproduce a shell beam with the thickness given above that is homogeneous through the thickness. The test is whether the two methods give the same result and match classical beam theory.

Note that the individual input files for each specific version run (implicit or explicit, layered or lofted), are included in this manual, but the documentation in this section applies to every version of this test.

3.13.1.1. Boundary Conditions

The problem is of a cantilevered beam, fully built-in at one end with zero displacement and zero rotation boundary conditions. The loading is a uniform pressure applied to one side of the shells.

3.13.1.2. Material

The materials used are all linear elastic with the properties in the accompanying table.

3.13.1.3. Feature Tested

The primary features tested are the layered shell section and the shell lofting capabilities.

Young's Modulus	E	$30.0e6$
Poisson's Ratio	ν	0.3
Density	ρ	$1.0e-1$

Table 3-21. Elastic Material Properties

3.13.2. Assumptions and notes

The major assumption in this test is that the strain is small enough to remain within small strain theory. For the loading and geometry given, the max strain is on the order of 1e-4. This puts the problem well within small strain theory.

Another assumption made is that the loading rate is small enough so that dynamic effects are not significant after the loading reaches its maximum value (and is held constant thereafter). By inspection this appears to be the case, but dynamic effects probably account for most of the observed (small) difference between numerical and analytic solutions.

3.13.3. Verification of Solution

Euler-Bernoulli beam theory provides the analytic solution. Here we have a simple cantilevered beam with a uniform distributed loading along its length. From Euler-Bernoulli beam theory the tip displacement u_{tip} is

$$u_{\text{tip}} = \frac{ql^4}{8EI} \tag{3.64}$$

while the moment at the built-in end M_0 is

$$M_0 = \frac{ql^2}{2}, \tag{3.65}$$

where q is the distributed load along the length l, E is Young's Modulus, and I is the bending moment of inertia perpendicular to the shell plane. For our shell beam with pressure loading p, width w, and thickness t

$$q = pw \tag{3.66}$$

and

$$I = \frac{wt^3}{12}. \tag{3.67}$$

Using the loading and geometry given here, at maximum load $|u_{\text{tip}}| = 0.0512$ and $|M_0| = 10$. The numerical results match these to within 0.2% for the Belytschko-Tsay shell in explicit and implicit calculations. The results match to a more loose tolerance, within 2.0%, for the Belytschko-Leviathan shell in explicit calculations due to extra bending stiffness from automatic hourglass controls. Implicit calculations for the Belytschko-Leviathan shell require a small amount of tangent diagonal shifting to be specified in the full tangent preconditioner to obtain convergence. The results are checked with these tolerances when this test is run using the solution verification capability. For input deck see Appendix B.26.

3.14. LINE WELD FORCE PER UNIT LENGTH

Analysis Type	Explicit Quasi-statics (Presto)
Element Type	Line Weld, Shell4
Strain Incrementation	
Material Model	Elastic
Verification Category	Refinement Consistency
Verification Quantities	Line Weld Force
Number of Tests	1
Keywords	Line Weld Refinement

3.14.1. Problem Description

This test checks that the computed element forces on the line weld are the same with each level of refinement (i.e., they are in units force per unit length). It is composed of two shell blocks with the same dimensions tied together with a line weld to simulate a butt weld connection. The shell containing the line weld is refined whereas the other is held at the same mesh refinement. Note the line weld is refined with the shell block it is attached to.

3.14.1.1. Boundary Conditions

The nodes of one side of the shell geometry (nodeset_1) are fixed in all directions. The nodes on the other side of the shell geometry (nodeset_2) are then displaced in a single component, either translational or rotational. The former tests translational stiffness of the line weld (in the x, y, and z directions) referred to as the "force version," and the latter tests rotational stiffness (about the x-axis) referred to as the "moment version." The displacement is applied in a cosine ramp from 0.0 to 1.0×10^{-4} (in m for force and rad for moment) over 0.8 seconds, and is then held at its final position until the termination time of 1.0 seconds.

3.14.1.2. Material Model

Each block uses an elastic material model. The parameters are shown in Table 3-22 and were chosen for convenience.

Table 3-22. Elastic Material Properties

Young's Modulus	E	2×10^{11} Pa
Poisson's Ratio	ν	0.33
Density	ρ	7830 kg/m^3

The line weld force and moment stiffness in the r, s, and t directions of the line weld are given by the following function. Note, the output of the function is defined as a force density (i.e., force per

unit length).

$$\frac{f}{l} = 12500x \tag{3.68}$$

The stiffness of the line weld is chosen to be much more compliant than the shells so that it achieves a displacement close to 1.0×10^{-4}. At this displacement, a nominal line weld "stress" of 1.25 is expected for both the force and the moment (given in N/m for the force version and N·m/rad for the moment version).

3.14.1.3. Feature Tested

This test verifies the line weld force output remains invariant to mesh refinement.

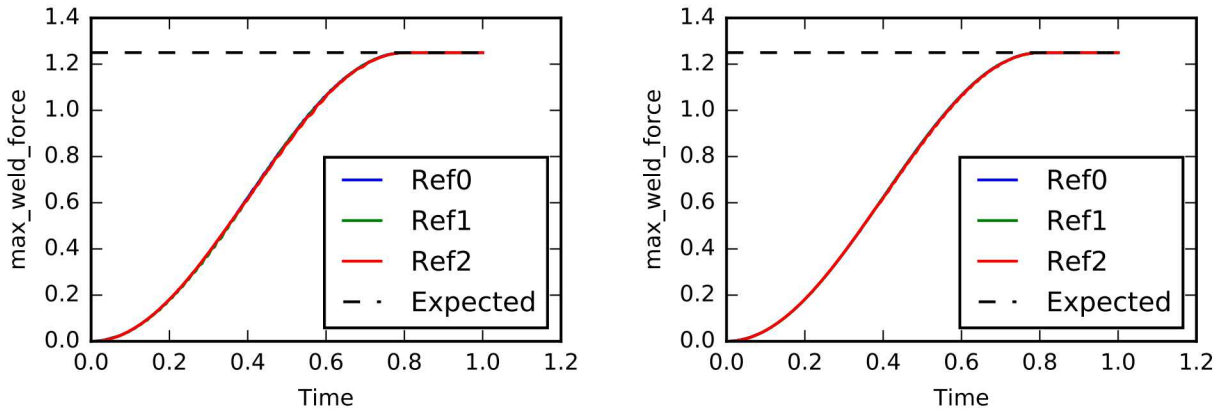
3.14.2. Assumptions and notes

This problem is assumed to be quasistatic in nature. It also assumes that the material is stiff enough to transfer the load through the shell with minimal deformation.

3.14.3. Verification of Solution

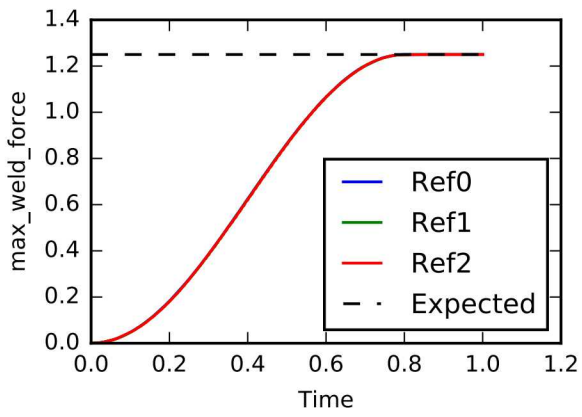
The line weld traction separation law serves as an analytic value. As the shell is displaced, the line weld computes the force to apply by looking up the correct value in the displacement function. The y value of the displacement function input by the user is in units of force/unit length, and thus the element LINE_WELD_force and LINE_WELD_moment values should not change with mesh refinement. The weld forces at each level of refinement can be seen in the Figure 3-48 and are shown to remain constant with refinement.

For input deck see Appendix B.27.

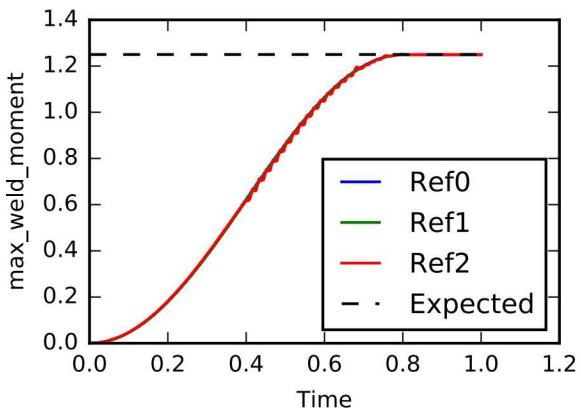


(a) Force along x

(b) Force along y



(c) Force along z



(d) Moment about x

Figure 3-48. Line Weld Force Comparison Under Refinement

3.15. LINE WELD CONVERGENCE

Analysis Type	Explicit Quasi-statics (Presto)
Element Type	Line Weld, Shell4
Strain Incrementation	
Material Model	Elastic
Verification Category	Convergence
Verification Quantities	Reaction Loads
Number of Tests	1
Keywords	Line Weld Refinement

3.15.1. Problem Description

This test checks that the computed reaction loads converge at the expected rate. It is composed of two shell blocks with the same dimensions tied together with a line weld to simulate a butt weld connection. The shell containing the line weld is refined whereas the other is held at the same mesh refinement. Note the line weld is refined with the shell block it is attached to. The shell that does not contain the line weld is converted to a rigid body and the resulting reaction loads on it are monitored for convergence.

3.15.1.1. Boundary Conditions

One side of the shell geometry (block_1) is made into a rigid body (rb_1) and is fixed in all directions. The end nodes on the other side of the shell geometry (nodeset_2) are then displaced linearly as a function of x in two translational component directions y and z. The displacement is applied in a cosine ramp from 0.0 to 1.0×10^{-4} (m) over 0.8 seconds, and is then held at its final position until the termination time of 1.0 seconds.

3.15.1.2. Material Model

Each block uses an elastic material model. The parameters are shown in Table 3-23 and were chosen for convenience.

Table 3-23. Elastic Material Properties

Young's Modulus	E	2×10^{11} Pa
Poisson's Ratio	ν	0.33
Density	ρ	7830 kg/m^3

The line weld force and moment stiffness in the r, s, and t directions of the line weld are given by the following function. Note, the output of the function is defined as a force density (i.e., force per

unit length or moment per unit length).

$$\frac{f}{l} = 12500x \tag{3.69}$$

The stiffness of the line weld is chosen to be much more compliant to exercise the line weld as the main contributor to the solution.

3.15.1.3. Feature Tested

This test verifies the line weld produces convergent results.

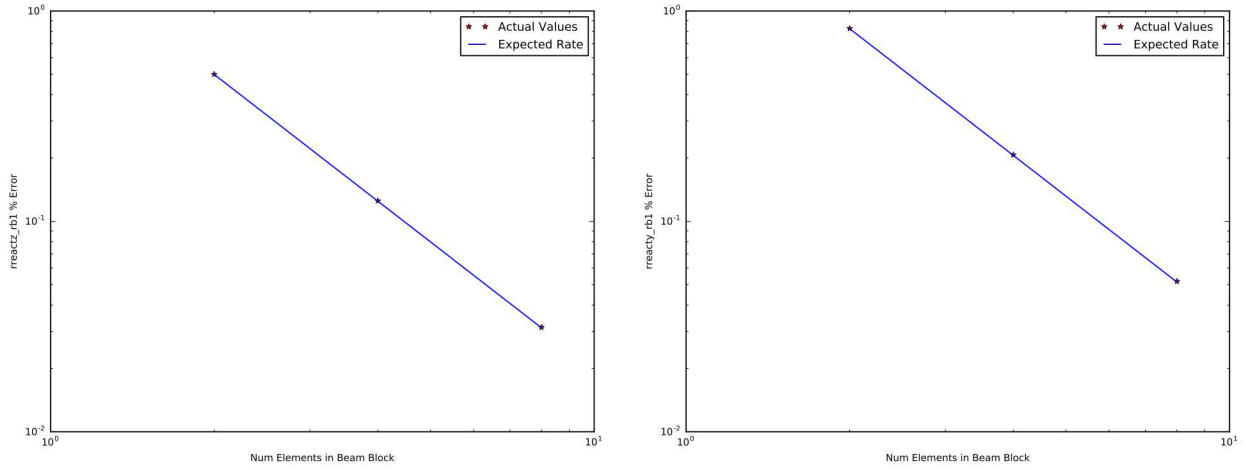
3.15.2. Assumptions and notes

This problem is assumed to be quasistatic in nature.

3.15.3. Verification of Solution

The expected convergence rate is quadratic (2.0). This verification test requires the actual convergence rate to be between 1.9 and 2.1.

The convergence rate of the rotational reactions in the y and z are plotted with the expected rate, shown in Figure 3-49.



(a) Rotational Reaction About Z-Axis **(b)** Rotational Reaction About Y-Axis

Figure 3-49. Reaction Load Convergence

For input deck see Appendix B.28.

3.16. LINE WELD FAILURE

Analysis Type	Explicit Quasi-statics (Presto)
Element Type	Line Weld, Shell4
Strain Incrementation	
Material Model	Elastic
Verification Category	Behavior
Verification Quantities	Line Weld Failure Loads
Number of Tests	1
Keywords	Line Weld Failure

3.16.1. Problem Description

This test checks that the computed element forces on the line weld are as expected during loading and failure. It is composed of two shell blocks with the same dimensions tied together with a line weld to simulate a butt weld connection.

3.16.1.1. Boundary Conditions

The nodes of one side of the shell geometry (nodeset_1) are fixed in all directions. The nodes on the other side of the shell geometry (nodeset_2) are then displaced in a single component, either translational or rotational. The former tests translational stiffness and failure of the line weld (in the x, y, and z directions) referred to as the "force version," and the latter tests rotational stiffness and failure (about the x-axis) referred to as the "moment version." The displacement is applied as a linear ramp.

3.16.1.2. Material Model

Each block uses an elastic material model. The parameters are shown in Table 3-24 and were chosen for convenience.

Table 3-24. Elastic Material Properties

Young's Modulus	E	2×10^{11} Pa
Poisson's Ratio	ν	0.33
Density	ρ	7830 kg/m^3

The line weld force and moment stiffness in the r, s, and t directions of the line weld are given by functions that ramp linearly to a failure load or moment per unit length of 1.25 at a displacement or rotation of 1.0×10^{-4}.

3.16.1.3. Feature Tested

This test verifies the line weld force remains within 5 percent of the expected value throughout loading and failure. The 5 percent error is necessary to allow for the dynamic nature of the problem. Given more steps and a true static solution the tolerance could be reduced to zero.

3.16.2. Assumptions and notes

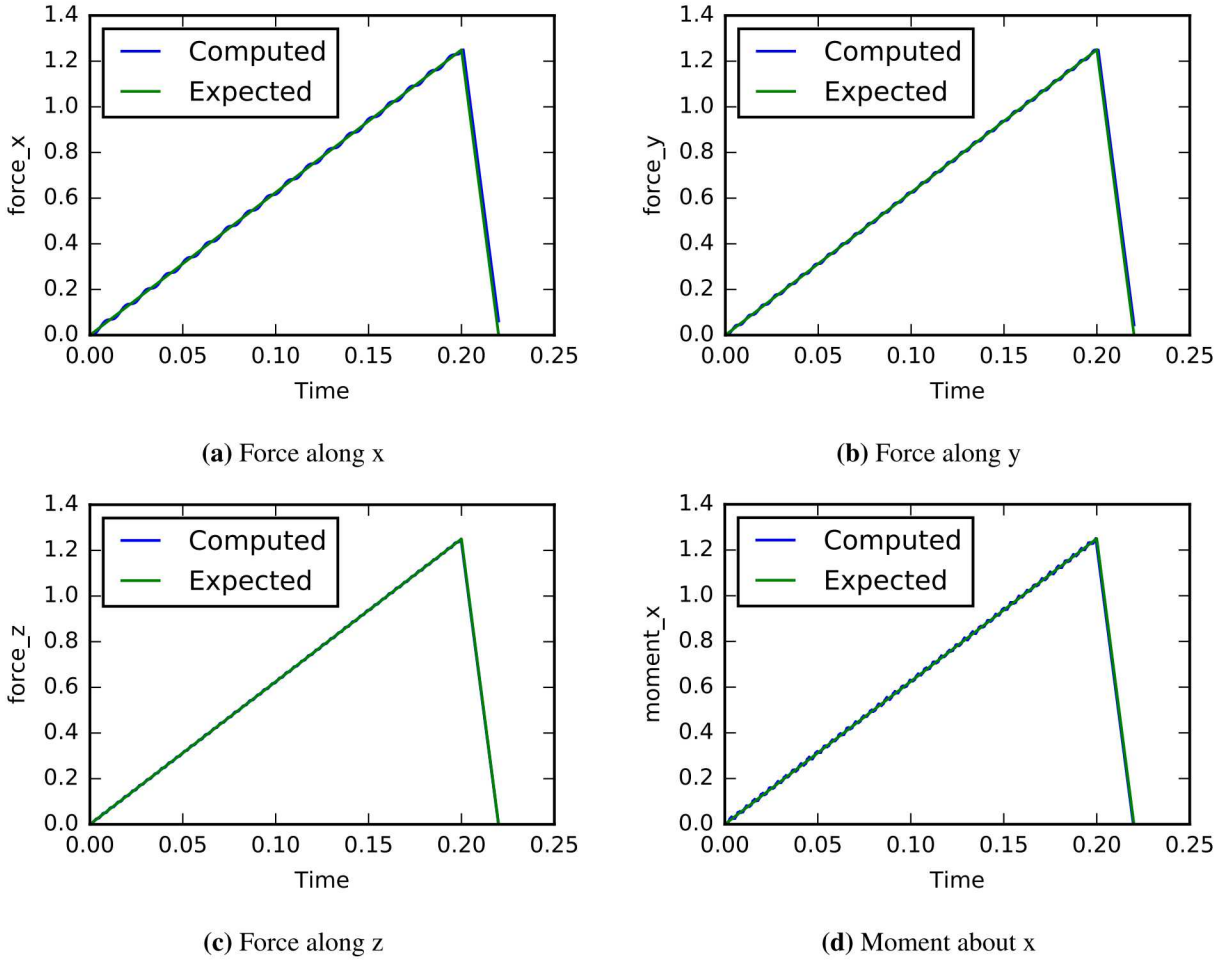
This problem is assumed to be near quasistatic in nature. It also assumes that the material is stiff enough to transfer the load through the shell with minimal deformation.

3.16.3. Verification of Solution

The line weld traction separation law serves as the analytic value. As the shell is displaced, the line weld computes the force to apply by looking up the correct value in the displacement function. The y value of the displacement function input by the user is in units of force/unit length, and thus the element LINE_WELD_force and LINE_WELD_moment values should match the expected values up to the start of failure. After failure the load is expected to decay to zero over the specified 2000 decay steps.

Figure 3-50 shows the individual cases matching the expected values.

For input deck see Appendix B.29.



(a) Force along x

(b) Force along y

(c) Force along z

(d) Moment about x

Figure 3-50. Line Weld Failure Force Verification

4. ENERGY VERIFICATION TESTS

The following tests verify the computation of various model global energy quantities. Tests are included to verify the energy computation is corrector for a given model setups and that errors in those energy quantities converge in the expected ways.

4.1. CONTACT FRICTIONAL ENERGY

Analysis Type	Explicit/Implicit Dynamics
Element Type	Hex8
Strain Incrementation	Strongly Objective
Material Model	Elastic
Keywords	Contact Energy, Friction

4.1.1. Problem Description

This test checks the computation of the contact energy. The test is composed of a unit cube sitting on top of a larger block. The top block has an applied displacement on the top surface while the bottom block has a prescribed displacement in the x-direction.

4.1.1.1. Boundary Conditions

The unit cube has a prescribed displacement of 4e-6 inches on the top surface. It also has a fixed displacement in the x-direction to prevent it from moving along with the block below it. The bottom block is fully prescribed where it has a fixed displacement in the y and z direction and has a final displacement of 0.1 inch in the x-direction.

4.1.1.2. Material Model

Each block uses an elastic material model.

Young's Modulus	E	$30e6$ Psi
Poisson's Ratio	ν	0.0
Density	ρ	$7.4e-4$ slug/in^3

4.1.1.3. *Feature Tested*

The computation of contact energy is tested in explicit dynamics and implicit dynamics.

4.1.2. Assumptions and notes

Poisson's ratio is set to 0 to prevent expansion of the unit cube during compression. The analytic solution does not take into account a change in contact area.

4.1.3. Verification of Solution

Contact energy should be equal to the amount of work done to the system. Work equals the frictional force multiplied by distance. The frictional force is computed by multiplying the coefficient of friction and normal force. A plot showing the theoretical work versus numerical solution is also shown below.

$$F = \mu N = \mu \sigma A = \mu E \epsilon A \tag{4.1}$$
$$F = 0.1 \times 30e6 \times (-4e-6) \times 1 \tag{4.2}$$
$$F = -12lb \tag{4.3}$$

$$W = Fx \tag{4.4}$$
$$W = -12 \times 0.1 \tag{4.5}$$
$$W = -1.2inlb \tag{4.6}$$

The maximum percent error computed for the simulation is less than 2%, where maximum percent error is computed by Equation 4.7.

$$\%\text{Error} = \frac{|\text{Analytic} - \text{Computed}|}{|\text{Analytic}|} * 100.0 \tag{4.7}$$

For input deck see Appendix B.30.

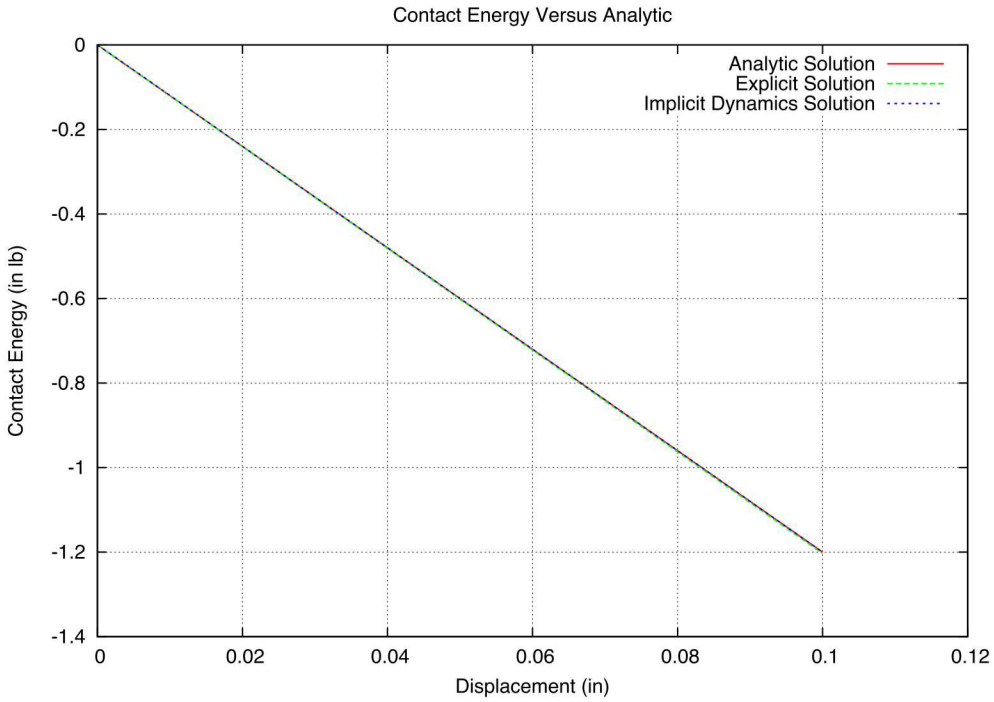


Figure 4-1. Energy-Displacement curve

4.2. CONTACT ENERGY WITHOUT FRICTION

Analysis Type	Explicit/Implicit Dynamics
Element Type	Hex8
Strain Incrementation	Strongly Objective
Material Model	Elastic
Keywords	Contact Energy, Frictionless

4.2.1. Problem Description

This test checks the computation of the contact energy. The test is composed of a unit cube sitting on top of a larger block. The top block has an applied displacement on the top surface while the bottom block has a prescribed displacement in the x-direction.

4.2.1.1. Boundary Conditions

The unit cube has a prescribed displacement of 4e-6 inches on the top surface. It also has a fixed displacement in the x-direction to prevent it from moving along with the block below it. The bottom block is fully prescribed where it has a fixed displacement in the y and z direction and has a final displacement of 0.1 inch in the x-direction.

4.2.1.2. Material Model

Each block uses an elastic material model. .

Young's Modulus	E	$30e6$ Psi
Poisson's Ratio	ν	0.0
Density	ρ	$7.4e-4$ slug/in^3

4.2.1.3. Feature Tested

The computation of contact energy is tested in explicit dynamics and implicit dynamics.

4.2.2. Assumptions and notes

Poisson's ratio is set to 0 to prevent expansion of the unit cube during compression. The analytic solution does not take into account a change in contact area.

4.2.3. Verification of Solution

Since there is no friction between the two blocks, the contact energy should be 0.

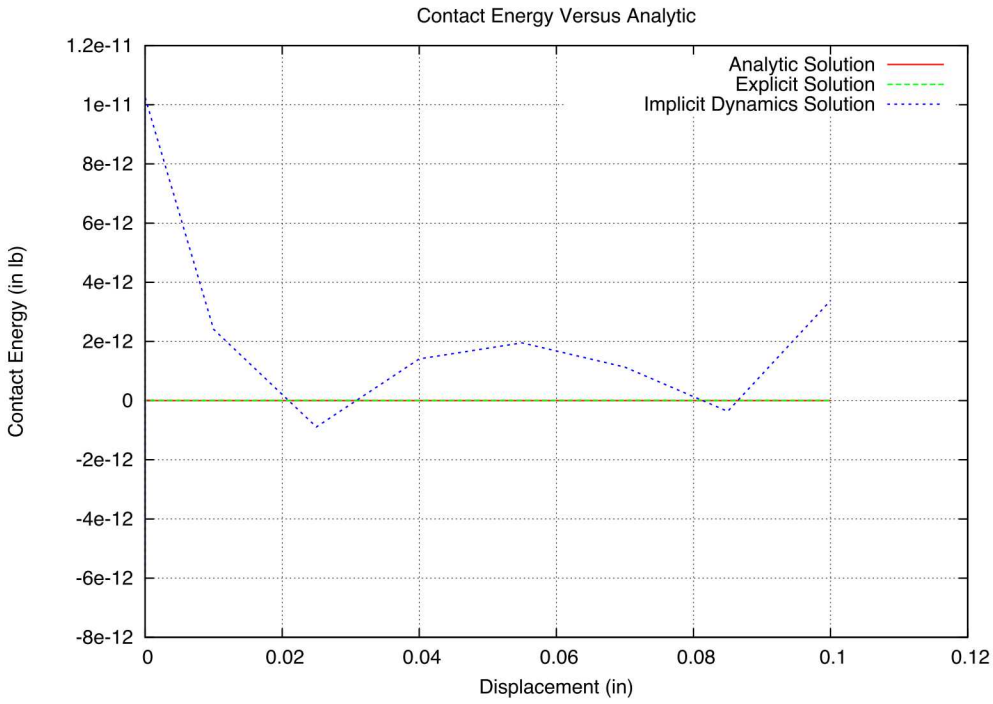


Figure 4-2. Energy-Displacement curve

The maximum value of contact energy computed for the simulation is less than (1.0e-2).

For input deck see Appendix B.31.

4.3. EXTERNAL ENERGY DUE TO APPLIED FORCE

Analysis Type	Explicit/Implicit Dynamics
Element Type	Mean Quadrature
Strain Incrementation	Strongly Objective
Material Model	Elastic
Keywords	External Energy, Force

4.3.1. Problem Description

A force is applied to a nodeset on a single element block.The applied energy due to the force should be equal to outputted external energy.

Element side length	L	1 m
Time	t	0.1 sec
Force	F	9810 N

4.3.1.1. Boundary Conditions

The one element block is not constrained in any way. A nodeset is added to the top of the block and a force is applied to this nodeset. The direction of the applied force is in y-direction (+ or - force will only change the sign of the outputted acceleration and displacement).

4.3.1.2. Material Model

The one element block uses an elastic material model. The density is set to 1000 kg/m^3 for ease of calculations.

Young's Modulus	E	200e+9 Pa
Poisson's Ratio	ν	0.3
Density	ρ	1000 kg/m^3

4.3.1.3. Feature Tested

Computation of external energy is tested.

4.3.2. Verification of Solution

External energy should be equal to the amount of work done to the system. Work equals force multiplied by distance. The derivations for acceleration, distance due to acceleration (from the applied force), and work are shown below. A plot showing the theoretical work versus numerical solution is also shown below.

$$F = ma \tag{4.8}$$
$$a = \frac{F}{m} \tag{4.9}$$
$$a = \frac{9810}{1000} \tag{4.10}$$
$$a = 9.81\frac{m}{s^2} \tag{4.11}$$

$$a = a_o \tag{4.12}$$
$$v = v_o + at \tag{4.13}$$
$$x = x_o + v_o t + \frac{1}{2}at^2 \tag{4.14}$$

$$x = 0 + 0 + \frac{1}{2} \times 9.81 \times (0.1)^2 \tag{4.15}$$
$$x = 0.04905m \tag{4.16}$$

$$W = Fx \tag{4.17}$$
$$W = 9810 \times 0.04905 \tag{4.18}$$
$$W = 481.1805J \tag{4.19}$$

The maximum percent error computed for the explicit and implicit dynamics simulation is less than 0.5%, where maximum percent error is computed by Equation 4.20.

$$\%\text{Error} = \frac{|\text{Analytic} - \text{Computed}|}{|\text{Analytic}|} * 100.0 \tag{4.20}$$

For input deck see Appendix B.32.

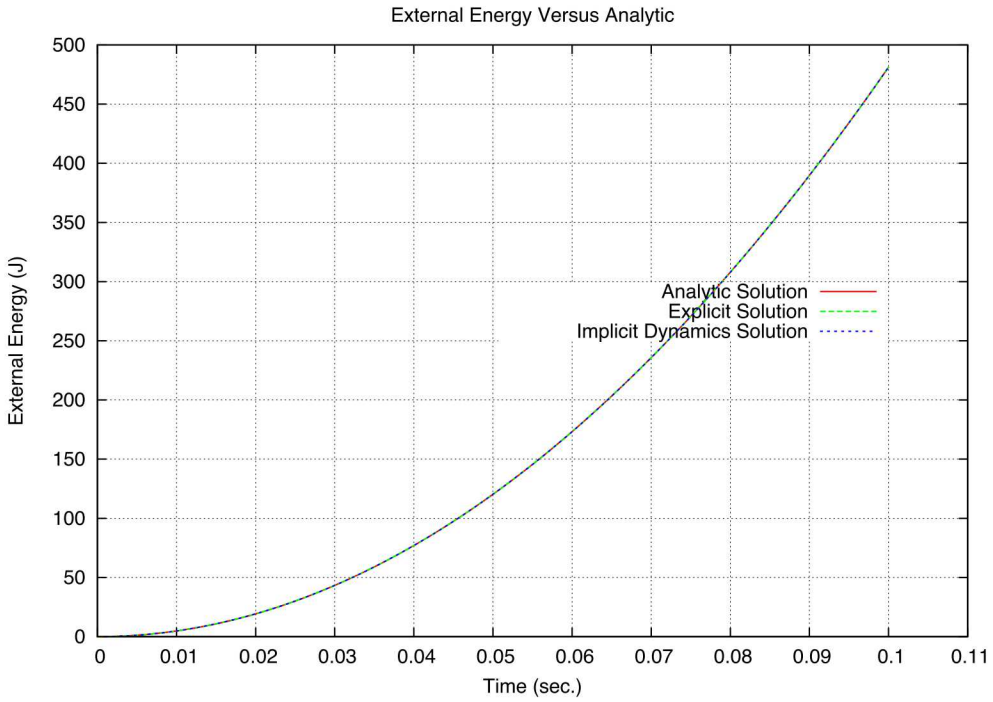


Figure 4-3. Energy-time curve

4.4. EXTERNAL ENERGY DUE TO GRAVITY

Analysis Type	Explicit/Implicit Dynamics
Element Type	Mean Quadrature
Strain Incrementation	Strongly Objective
Material Model	Elastic
Keywords	External Energy, Gravity

4.4.1. Problem Description

Gravity is applied to a single element block.The applied energy due to gravity should be equal to outputted external energy.

Element side length	L	1 m
Time	t	0.1 sec
Gravity	g	9.81 m/s^2

4.4.1.1. Boundary Conditions

The one element block is not constrained in any way. Only gravity effects the block.

4.4.1.2. Material Model

The one element block uses an elastic material model. The density is set to 1000 kg/m^3 for ease of calculations.

Young's Modulus	E	200e+9 Pa
Poisson's Ratio	ν	0.3
Density	ρ	1000 kg/m^3

4.4.1.3. Feature Tested

Computation of external energy is tested.

4.4.2. Verification of Solution

External energy should be equal to the amount of work done to the system. Work equals force multiplied by distance. The derivations for distance due to acceleration (gravity), force, and work are shown below. A plot showing the theoretical work versus numerical solution is also shown below.

$$a = a_o \tag{4.21}$$
$$v = v_o + at \tag{4.22}$$
$$x = x_o + v_o t + \frac{1}{2} a t^2 \tag{4.23}$$

$$x = 0 + 0 + \frac{1}{2} \times 9.81 \times (0.1)^2 \tag{4.24}$$
$$x = 0.04905m \tag{4.25}$$

$$F = ma \tag{4.26}$$
$$F = 1000 \times 9.81 \tag{4.27}$$
$$F = 9810N \tag{4.28}$$

$$W = Fx \tag{4.29}$$
$$W = 9810 \times 0.04905 \tag{4.30}$$
$$W = 481.1805J \tag{4.31}$$

The maximum percent error computed for the explicit and implicit dynamics simulations are less than 0.5%, where maximum percent error is computed by Equation 4.32.

$$\%\text{Error} = \frac{|\text{Analytic} - \text{Computed}|}{|\text{Analytic}|} * 100.0 \tag{4.32}$$

For input deck see Appendix B.33.

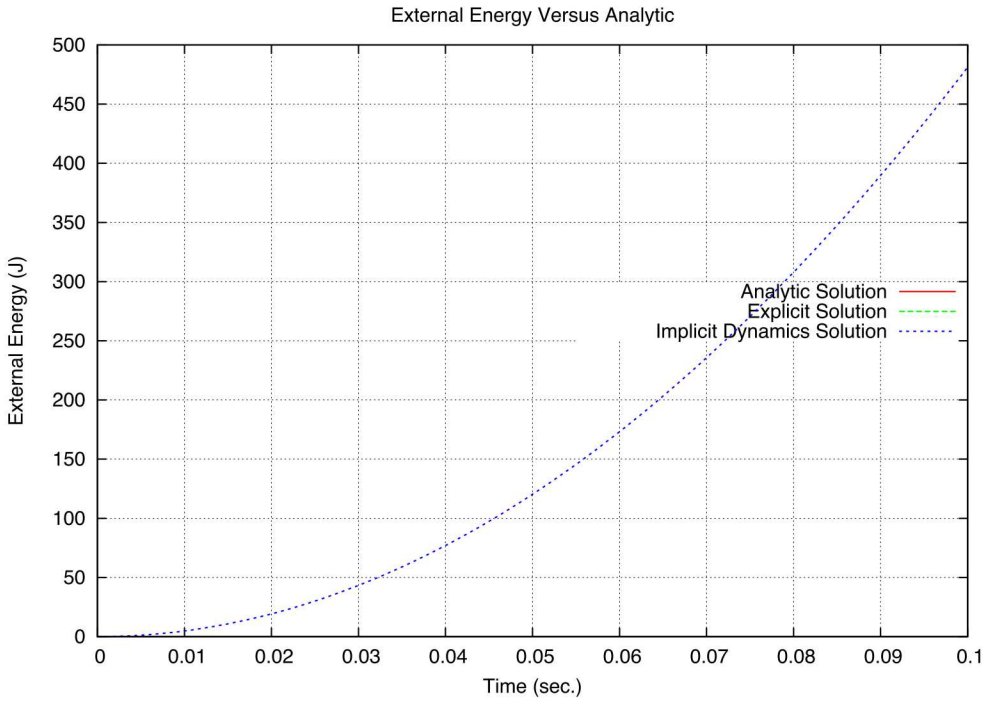


Figure 4-4. Energy-time curve

4.5. HOURGLASS ENERGY FOR UNIFORM GRADIENT HEX ELEMENT WITH MIDPOINT INCREMENT FORMULATION

Analysis Type	Explicit/Implicit Dynamics
Element Type	Hex8
Material Model	Elastic
Keywords	Hourglass Energy, UG Hex8, Midpoint Increment

4.5.1. Problem Description

This test checks the computation of the hourglass energy. The test is composed of a unit cube subjected to hourglass deformation. The same model is run with three levels of mesh refinement; the hourglass energy should converge to zero as the mesh is refined.

4.5.1.1. Boundary Conditions

The cube is fully prescribed with kinematic boundary conditions. Displacement boundary conditions are applied such that the one-element version of the test is in pure hourglass deformation.

4.5.1.2. Material

The only material in the problem is elastic. The elastic values are given below.

Young's Modulus	E	200.0e3 MPa
Poisson's Ratio	ν	0.3
Density	ρ	7900.0 kg/m^3

Table 4-1. Elastic Material Properties

4.5.1.3. Feature Tested

The computation of hourglass energy is tested.

4.5.2. Assumptions and notes

This test assumes accuracy in the material model and applied boundary conditions.

4.5.3. Verification of Solution

This problem is exercised in explicit dynamics and implicit dynamics; hourglass energy should exhibit convergence to zero as the mesh is refined. The rate at which the hourglass energy drops is expected to be 4 but a value between 3 and 4 is accepted in this test.

Figure 4-5 shows the explicit simulation total hourglass energy as the number of elements in the mesh is increased along with upper and lower bounds of expectation.

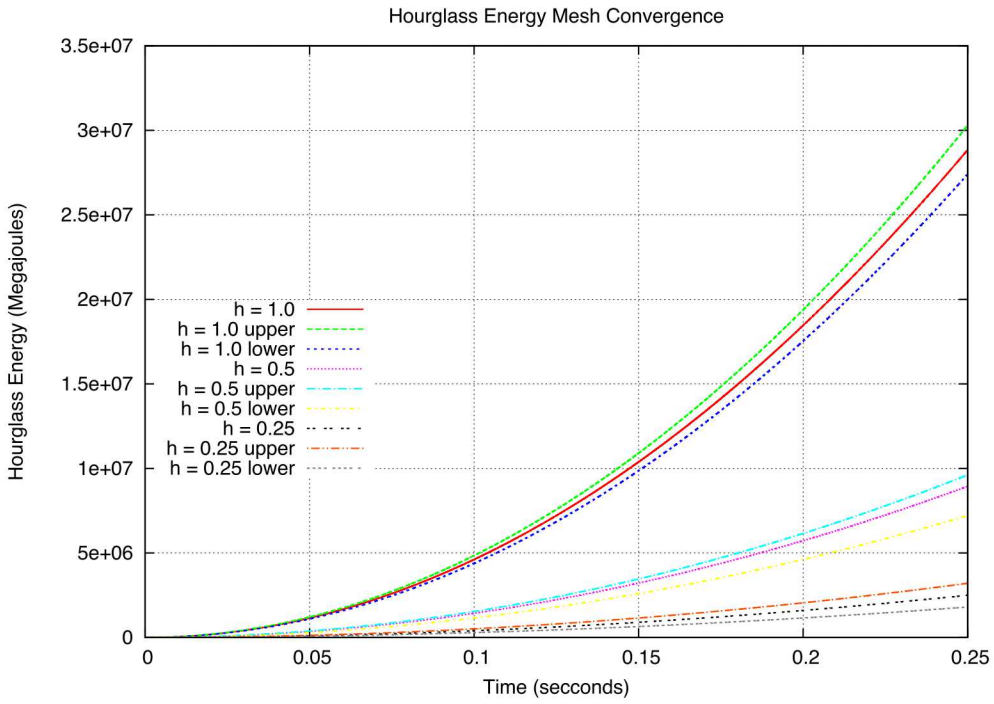


Figure 4-5. Hourglass energy for various levels of mesh refinement: Explicit Dynamics

Figure 4-6 shows the implicit dynamics simulation total hourglass energy as the number of elements in the mesh is increased along with upper and lower bounds of expectation.

Results from the one-element version of the test are expected to change less than 5%.

For input deck see Appendix B.34.

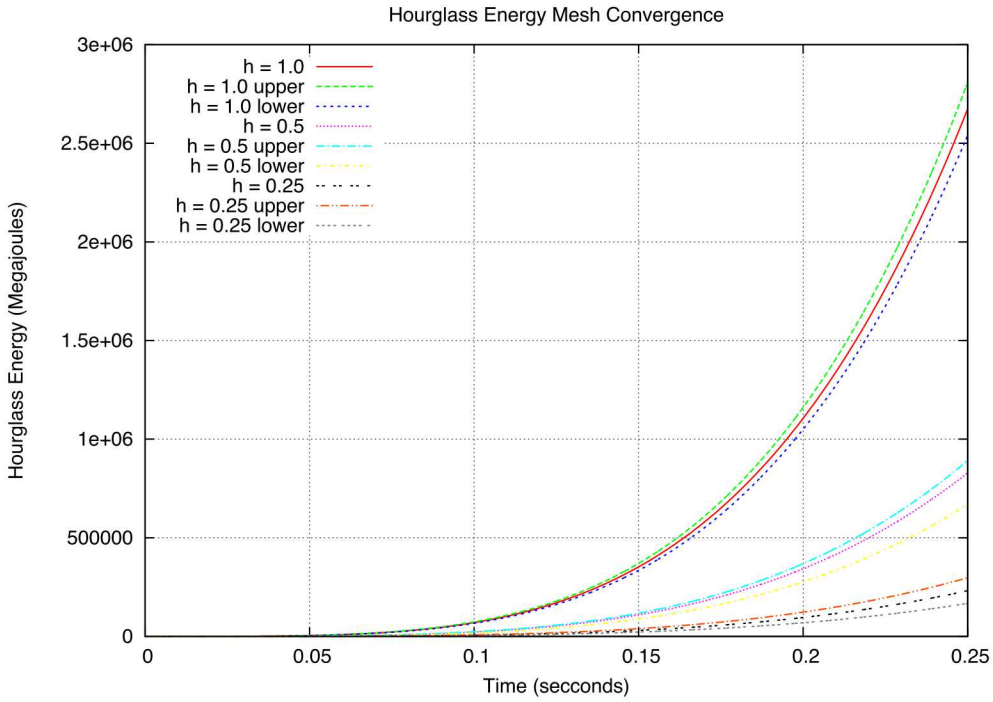


Figure 4-6. Hourglass energy for various levels of mesh refinement: Implicit Dynamics

4.6. HOURGLASS ENERGY FOR UNIFORM GRADIENT HEX ELEMENT WITH STRONGLY OBJECTIVE FORMULATION

Analysis Type	Explicit Dynamics
Element Type	Hex8
Material Model	Elastic
Keywords	Hourglass Energy, UG Hex8, Strongly Objective

4.6.1. Problem Description

This test checks the computation of the hourglass energy. The test is composed of a unit cube subjected to hourglass deformation. The same model is run with three levels of mesh refinement; the hourglass energy should converge to zero as the mesh is refined.

4.6.1.1. Boundary Conditions

The cube is fully prescribed with kinematic boundary conditions. Displacement boundary conditions are applied such that the one-element version of the test is in pure hourglass deformation.

4.6.1.2. Material

The only material in the problem is elastic. The elastic values are given below.

Young's Modulus	E	200.0e3 MPa
Poisson's Ratio	ν	0.3
Density	ρ	7900.0 kg/m^3

Table 4-2. Elastic Material Properties

4.6.1.3. Feature Tested

The computation of hourglass energy is tested.

4.6.2. Assumptions and notes

This test assumes accuracy in the material model and applied boundary conditions.

4.6.3. Verification of Solution

This problem is exercised in explicit dynamics; hourglass energy should exhibit convergence to zero as the mesh is refined.

Figure 4-7 shows the total hourglass energy as the number of elements in the mesh is increased.

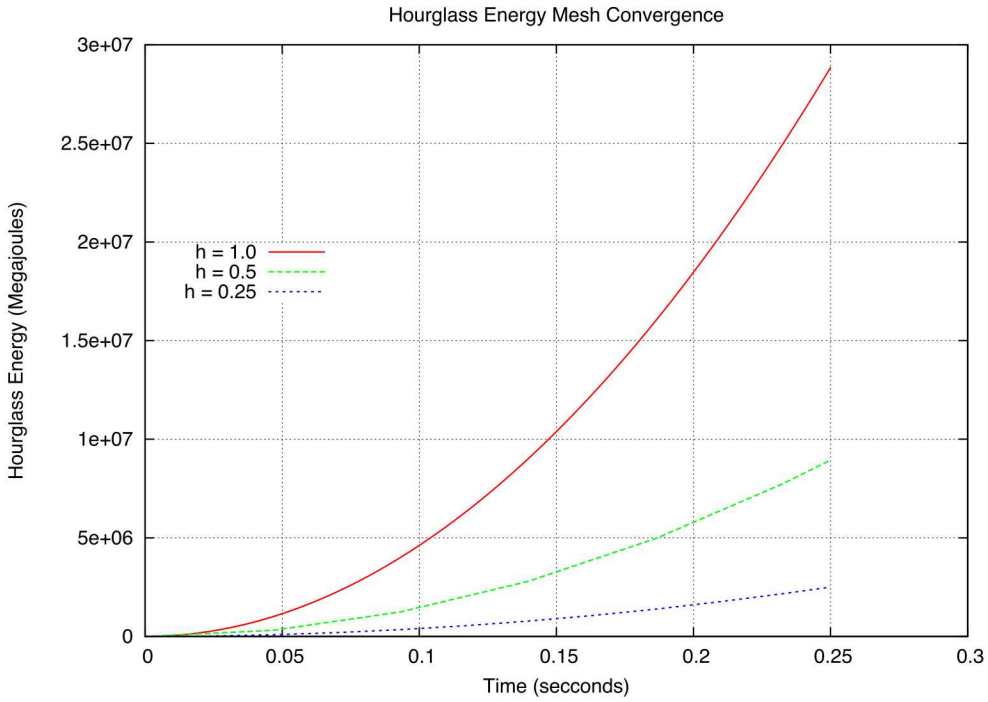


Figure 4-7. Hourglass energy for various levels of mesh refinement

To determine if the test passes or fails, results from the one-element version of the test are compared to a gold solution file.

For input deck see Appendix B.35.

4.7. HOURGLASS ENERGY WITH VISCOSITY CONTROL FOR UNIFORM GRADIENT HEX ELEMENT WITH STRONGLY OBJECTIVE FORMULATION

Analysis Type	Explicit Dynamics
Element Type	Hex8
Material Model	Elastic
Keywords	Hourglass Energy, UG Hex8, Strongly Objective, Viscous

4.7.1. Problem Description

This test checks the computation of the hourglass energy for viscous hourglass control. The test is composed of a unit cube subjected to hourglass deformation (pure hourglass deformation for the one-element model). The same model is run with three levels of mesh refinement; the hourglass energy should converge to zero as the mesh is refined. In addition, the magnitude of the hourglass energy is checked for the case where the hourglass viscosity coefficient is doubled and for the case where the strain rate is doubled. In both cases, the hourglass energy should double as well.

4.7.1.1. Boundary Conditions

The cube is deformed using prescribed kinematic boundary conditions. Displacement boundary conditions are applied such that the one-element version of the test is in pure hourglass deformation.

4.7.1.2. Material

The only material in the problem is elastic. The elastic values are given below.

Young's Modulus	E	200.0e3 MPa
Poisson's Ratio	ν	0.3
Density	ρ	7900.0 kg/m^3

Table 4-3. Elastic Material Properties

4.7.1.3. Feature Tested

Viscous hourglass control is tested.

4.7.2. Assumptions and notes

This test assumes accuracy in the material model and applied boundary conditions.

4.7.3. Verification of Solution

This problem is exercised in explicit dynamics; hourglass energy should exhibit convergence to zero as the mesh is refined. Additionally, the hourglass energy should increase by a factor of two if the viscous hourglass coefficient is doubled or if the strain rate is doubled.

Figure 4-8 shows the total hourglass energy as the number of elements in the mesh is increased.

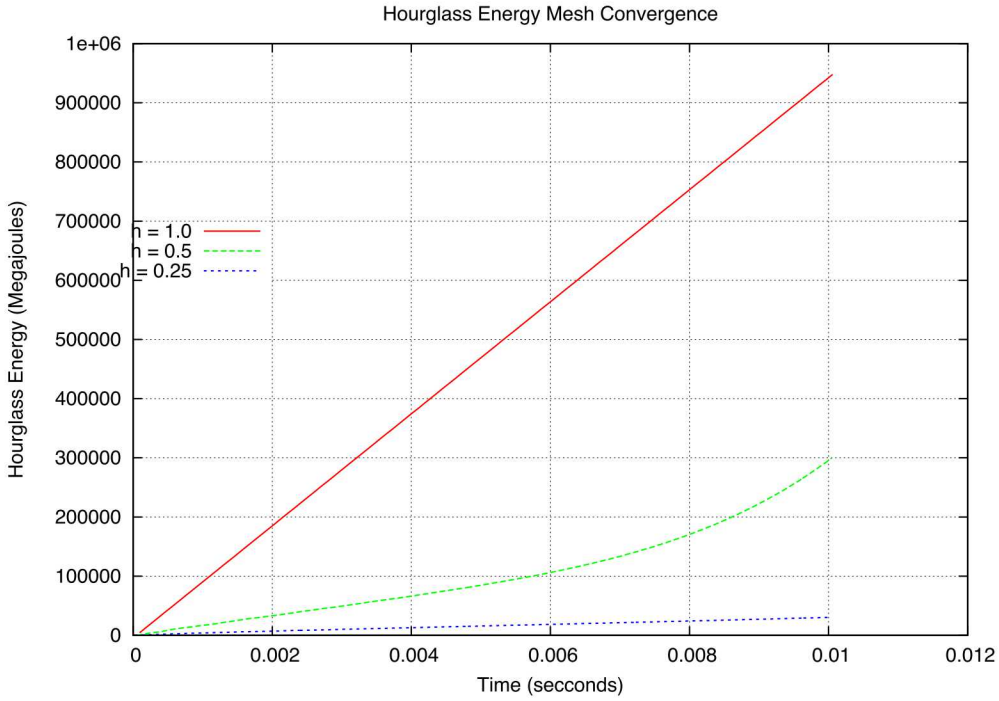


Figure 4-8. Hourglass energy for various levels of mesh refinement

Figure 4-9 shows the total hourglass energy when the viscous hourglass coefficient is doubled.

Figure 4-10 shows the total hourglass energy for two levels of strain rate.

To determine if this test passes or fails, results from the 64-element version of the test are compared to a gold solution file.

For input deck see Appendix B.36.

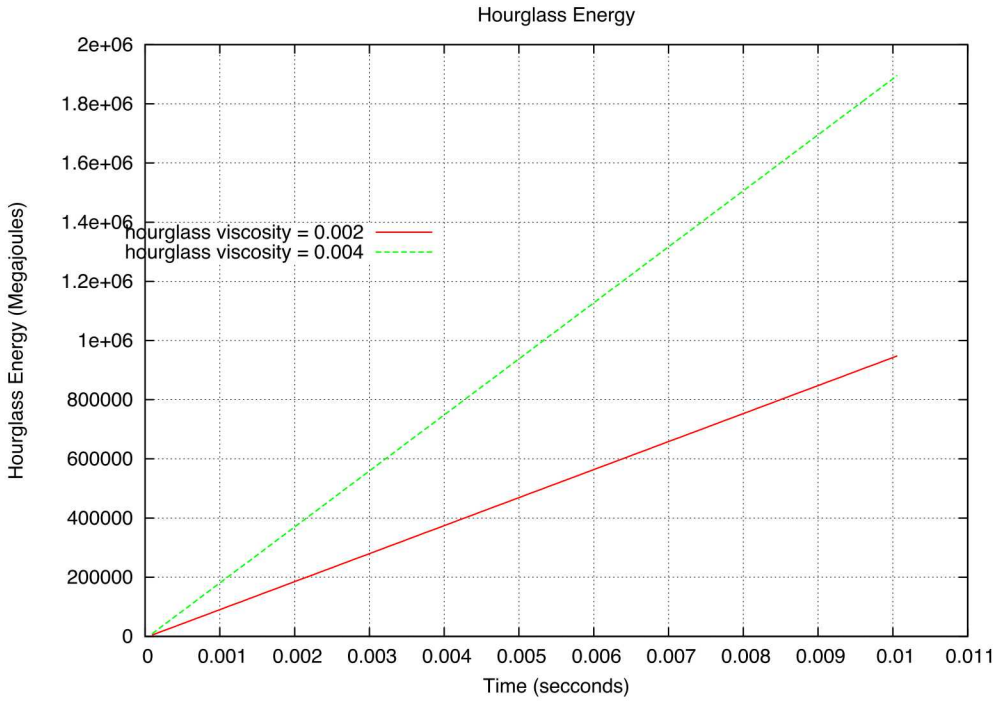


Figure 4-9. Hourglass energy for two levels of viscous hourglass coefficient

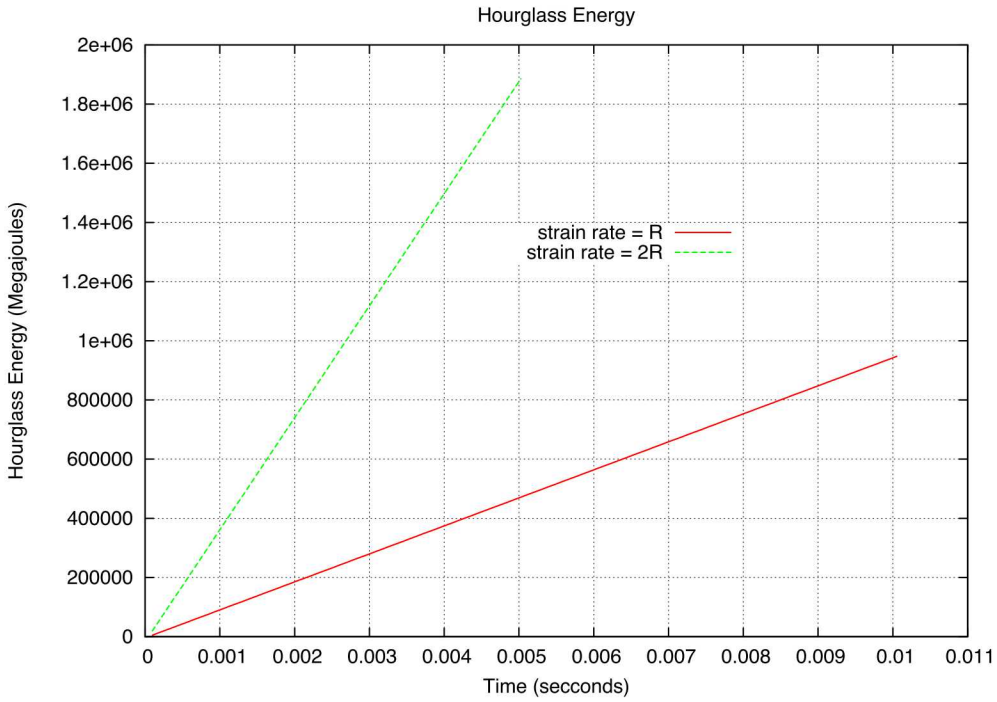


Figure 4-10. Hourglass energy for two levels of strain rate

4.8. INTERNAL ENERGY – EXPLICIT AND IMPLICIT DYNAMICS

Analysis Type	Explicit/Implicit Dynamics
Element Type	Hex8
Material Model	Elastic
Keywords	Internal Energy

4.8.1. Problem Description

This test checks the computation of the internal energy. The test is composed of a unit cube.

4.8.1.1. Boundary Conditions

The cube is fully prescribed with kinematic boundary conditions. All directions are fixed except for the z direction where compressive displacement boundary conditions are applied. The displacement is linear up to 1% strain.

4.8.1.2. Material

The only material in the problem is elastic. The elastic values are given below.

Young's Modulus	E	200.0e3 MPa
Poisson's Ratio	ν	0.3
Density	ρ	7900.0 kg/m^3

Table 4-4. Elastic Material Properties

4.8.1.3. Feature Tested

The computation of internal energy is tested.

4.8.2. Assumptions and notes

This test assumes accuracy in the material model and applied boundary conditions. Also, the loading is assumed to approximate quasistatic conditions.

4.8.3. Verification of Solution

This problem is exercised in explicit dynamics and implicit dynamics. They should match the analytic values of internal energy computed from Equation (4.33). This equation gives the work done during compression, which is equal to the increase in internal energy in this problem.

$$\text{Internal Energy} = 0.5 * \text{area} * \text{Young's modulus} * \text{strain}_{xx}^2. \tag{4.33}$$

Figure 4-11 shows the code computed values for internal energy in the test.

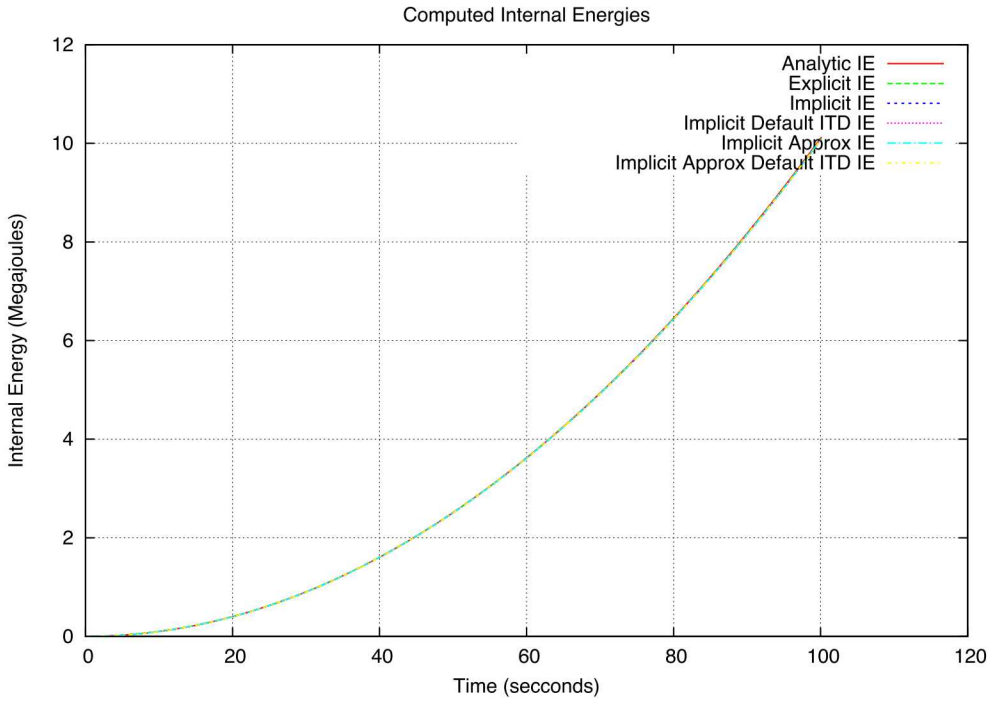


Figure 4-11. Analytic and Computed Values of Internal Energy

The maximum percent error computed for the simulations are less than 0.5% where maximum percent error is computed by Equation 4.34.

$$\%\text{Error} = \frac{|\text{Analytic} - \text{Computed}|}{|\text{Analytic}|} * 100.0 \tag{4.34}$$

For input deck see Appendix B.37.

4.9. INTERNAL (STRAIN) ENERGY – QUASISTATICS

Analysis Type	Quasi-statics
Element Type	Hex8
Material Model	Elastic
Keywords	Internal Energy

4.9.1. Problem Description

This test checks the computation of the strain energy. The test is composed of a unit cube.

4.9.1.1. Boundary Conditions

The cube is fully prescribed with kinematic boundary conditions. All faces are fixed except for one, where compressive displacement boundary conditions are applied initially, followed by shear displacement boundary conditions. The maximum displacement compressive strain is 1% and the maximum shear strain is 0.1%.

4.9.1.2. Material

The only material in the problem is elastic. The elastic values are given below.

Young's Modulus	E	200.0e3 MPa
Poisson's Ratio	ν	0.3
Density	ρ	7900.0 kg/m^3

Table 4-5. Elastic Material Properties

4.9.1.3. Feature Tested

The computation of strain energy is tested.

4.9.2. Assumptions and notes

This test assumes accuracy in the material model and applied boundary conditions. Also, the loading is assumed to approximate quasistatic conditions.

4.9.3. Verification of Solution

This problem is exercised in quasistatics. It should match the analytic values of compressive strain energy computed from Equation (4.35) for the first 100.0 seconds and the values of shear strain energy from Equation (4.36) for the second 100.0 seconds. These equations give the work done during compression and shear, respectively, which are equal to the strain energy in this problem.

$$\text{Strain Energy} = 0.5 * \text{area} * \text{Young's modulus} * \text{strain}_{xx}^2. \tag{4.35}$$

$$\text{Strain Energy} = 0.25 * \text{area} * \frac{\text{Young's modulus}}{1.0 + \text{Poisson's ratio}} * \text{strain}_{xy}^2. \tag{4.36}$$

Figures 4-12 and 4-13 show the analytic and code-computed values of strain energy for this test.

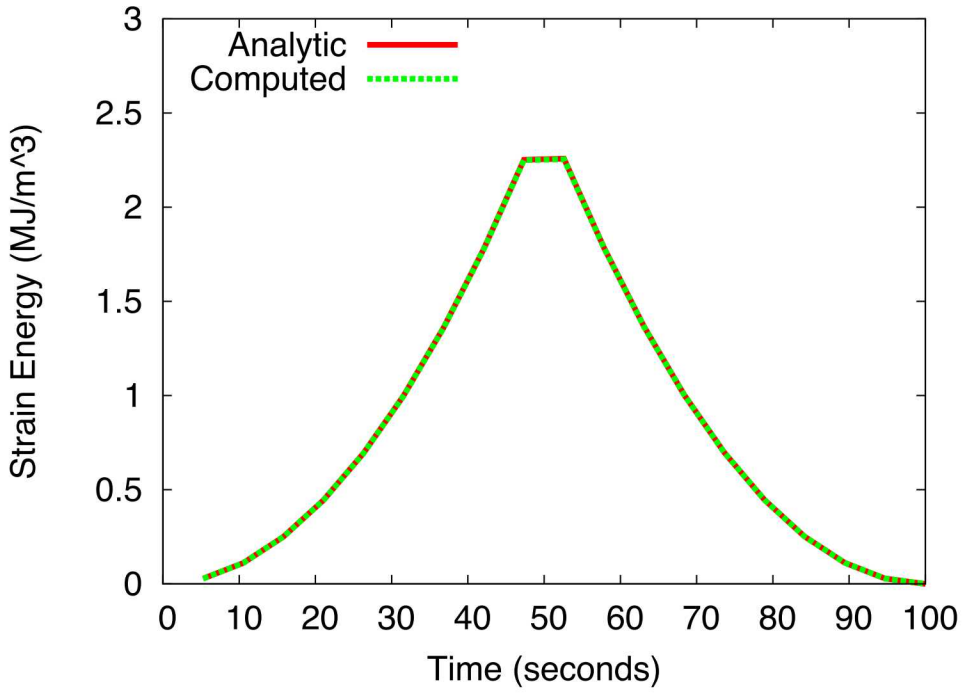


Figure 4-12. Analytic and computed values of strain energy for compressive loading

The maximum percent errors computed for the simulations are less than 0.14% for the compressive strain energy and less than 0.05% for the shear strain energy, where maximum percent error is computed by Equation 4.37.

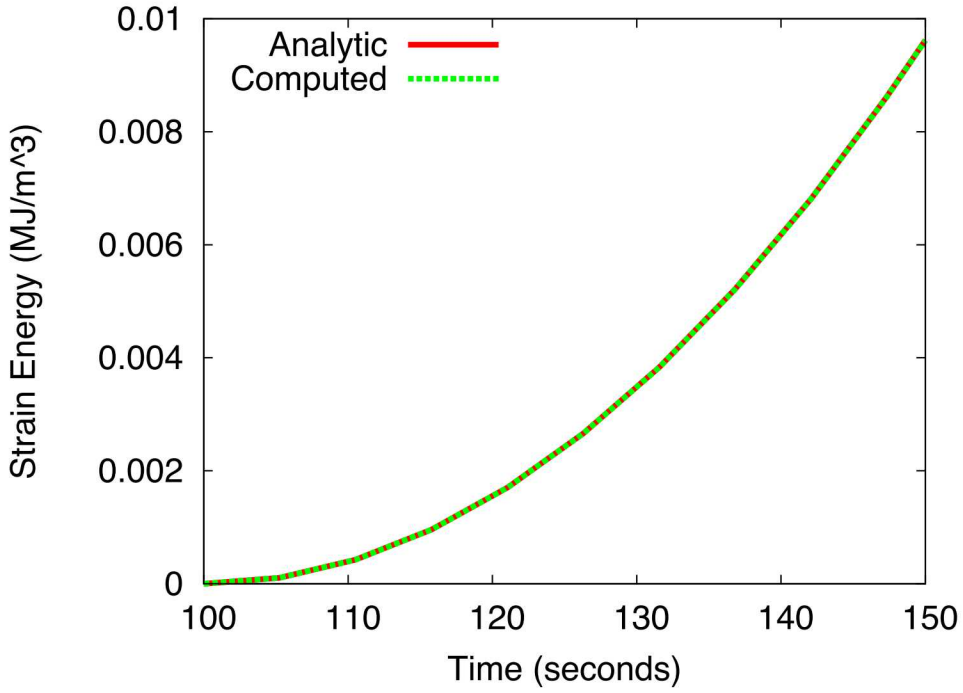


Figure 4-13. Analytic and computed values of strain energy for shear loading

$$\%\text{Error} = \frac{|\text{Analytic} - \text{Computed}|}{|\text{Analytic}|} * 100.0 \tag{4.37}$$

For input deck see Appendix B.38.

4.10. KINETIC ENERGY

Analysis Type	Explicit/Implicit Dynamics
Element Type	Hex8
Strain Incrementation	Mid-Point Incrementation
Material Model	Elastic
Verification Category	Analytic Solution
Verification Quantities	Kinetic Energy
Keywords	Kinetic Energy

4.10.1. Problem Description

This test verifies the computation of the kinetic energy on a unit cube subjected to a prescribed sinusoidal velocity boundary condition.

4.10.1.1. Boundary Conditions

The cube is fully prescribed with kinematic boundary conditions. All directions are fixed except for the one direction where a sinusoidal velocity is prescribed according to Equation 4.38.

$$Velocity = 2.0 * sin(2.0 * \pi * t) \tag{4.38}$$

4.10.1.2. Material

The only material in the problem is elastic. The elastic values were selected for convenience of calculation.

Young's Modulus	E	$30.0e6$
Poisson's Ratio	ν	0.3
Density	ρ	100.0

Table 4-6. Elastic Material Properties

4.10.1.3. Feature Tested

The computation of kinetic energy is tested for explicit and implicit dynamics.

4.10.2. Assumptions and notes

This test assumes the prescribed velocity is correctly enforced.

4.10.3. Verification of Solution

This problem is exercised in explicit and implicit dynamics and should match the analytic values of kinetic energy computed from the Equation 4.39.

$$\text{Kinetic Energy} = 0.5 * mass * velocity^2 \tag{4.39}$$

Figure 4-14 shows the analytic and computed values for kinetic energy in the test.

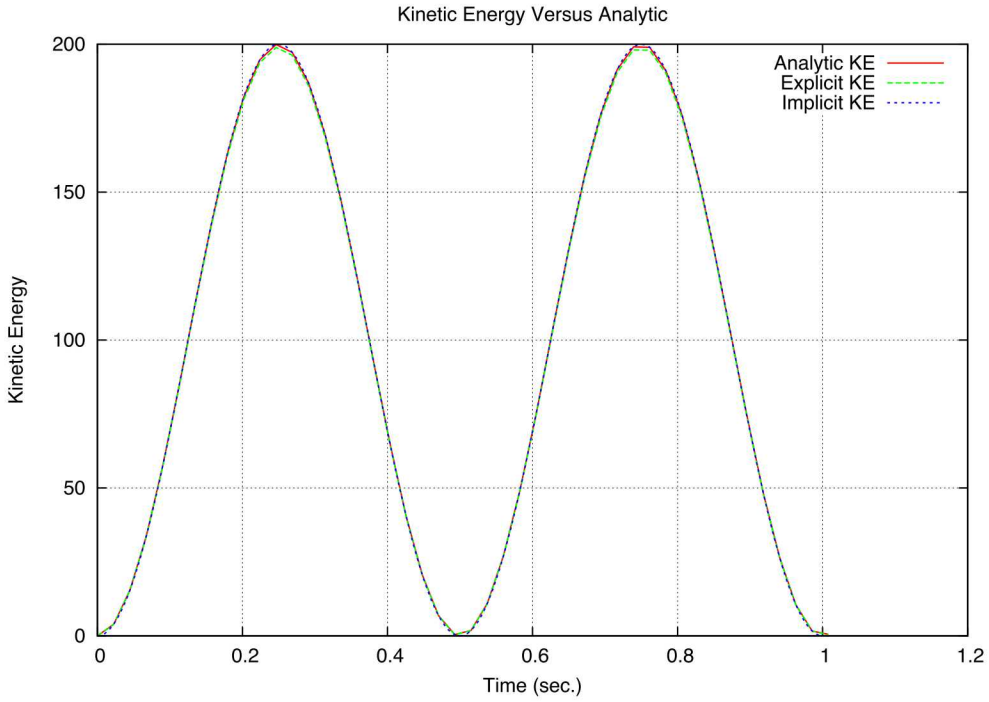


Figure 4-14. Analytic and Computed Values of Kinetic Energy

The maximum percent error computed for the simulations is less than 0.5% (skipping where energy crosses zero) where maximum percent error is computed by Equation 4.40.

$$\%Error = \frac{|Analytic - Computed|}{|Analytic|} * 100.0 \tag{4.40}$$

For input deck see Appendix B.39.

APPENDIX A. OTHER SIERRA/SM VERIFICATION TESTS NOT IN THIS DOCUMENT

The following tables list verification tests that exist in the nightly Sierra/SM test suite, but have not yet been formally added to this document. Each of these tests is run nightly and compares the results of a Sierra/SM code run to an analytic result. Over time, more of these tests will be included in this document with full documentation of code results and analytic solutions.

In addition to the verification tests, several thousand additional regression tests are run nightly with the Sierra/SM code suite. These regression tests ensure that: different capabilities remain functional when used together; the performance of the code does not degrade; all supported input commands are handled correctly; and, code results of analyses that are too complex to have an analytic solution still compare closely with known acceptable results.

Table A-1. Additional Contact Verification Tests

Test Location
adagio_rtest/adagio/contactSimpleFrictionless/
adagio_rtest/adagio/contactTiedWithKinBc/
adagio_rtest/adagio/contact_fixed/
adagio_rtest/adagio/contact_simple/
adagio_rtest/high_velocity_contact/hex_v_hex/
adagio_rtest/high_velocity_contact/sphere_v_hex/
adagio_rtest/presto/chatter_contact/
adagio_rtest/presto/contact_kinbc_slice/
adagio_rtest/presto/ContactGeometryCheck/
adagio_rtest/presto/initial_overlap4/
adagio_rtest/presto/vtest/contact_2block_init_velocity_nogap/
adagio_rtest/presto/vtest/contact_2block_init_velocity_withgap/
adagio_rtest/presto/vtest/contact_3slide_blocks/
adagio_rtest/presto/vtest/contact_fixed/
adagio_rtest/presto/vtest/contact_patch_test_1/
adagio_rtest/presto/vtest/contact_patch_test_2/
adagio_rtest/presto/vtest/contact_patch_test_3/
adagio_rtest/presto/vtest/remove_initial_overlap/
adagio_rtest/verification/contact_tests/shell_edge_test/
adagio_rtest/verification/elastic_bar_impact/

Table A-2. Additional Material Verification Tests

Test Location
adagio_rtest/materials/abaqus_umat_vumat/
adagio_rtest/materials/elastic/
adagio_rtest/materials/ep_power_law/
adagio_rtest/materials/johnson_cook/
adagio_rtest/materials/ml_ep_fail/
adagio_rtest/materials/ml_ep_fail_biaxial/
adagio_rtest/materials/plane_stress_rate_plasticity_vtest/
adagio_rtest/materials/thermoelastic/
adagio_rtest/presto/vtest/ductile_death_disp/
adagio_rtest/presto/vtest/elastic_death_disp/
adagio_rtest/presto/vtest/elastic_death_load/
adagio_rtest/presto/vtest/material_ep_one_elem_disp/
adagio_rtest/presto/vtest/material_ep_one_elem_press/
adagio_rtest/presto/vtest/B61_MLEP_dynamic_compress/
adagio_rtest/presto/vtest/B61_MLEP_quasi_compress/
adagio_rtest/presto/vtest/B61_MLEP_thermal_strain/
adagio_rtest/presto/vtest/B61_MLEP_vpf_quasi_compress/
adagio_rtest/presto/vtest/B61_ORTHO_dynamic_compress/
adagio_rtest/presto/vtest/B61_ORTHO_quasi_compress/
adagio_rtest/presto/vtest/B61_vpf_dynamic_compress/
adagio_rtest/presto/vtest/B61_vpf_thermal_strain/
adagio_rtest/presto/vtest/B61_vpf_quasi_compress/

Table A-3. Additional Solid Element Verification Tests

Test Location
adagio_rtest/adagio/ug3dh8_patch_test_velbc/
adagio_rtest/adagio/ug3dh8_patch_test/
adagio_rtest/adagio/tet10_uni_disp_cube/
adagio_rtest/adagio/tet4_uni_disp_cube/
adagio_rtest/presto/tet_conv0/
adagio_rtest/presto/vtest/tet10_one_elem/
adagio_rtest/presto/vtest/tet4_one_elem/
adagio_rtest/presto/vtest/tet_so_patch_tests/
adagio_rtest/presto/vtest/ug3dh8_mi_conv0/
adagio_rtest/presto/vtest/rotating_ring_off_axis_hex/
adagio_rtest/presto/vtest/rotating_ring_off_axis_hexso/
adagio_rtest/presto/vtest/rotating_ring_on_axis_hex/
adagio_rtest/presto/vtest/rotating_ring_on_axis_hexso/
adagio_rtest/verification/nodal_stress/cantilever_beam_convergence/fully_integrated_hex8/
adagio_rtest/verification/nodal_stress/cantilever_beam_convergence/hex27/
adagio_rtest/verification/nodal_stress/cantilever_beam_convergence/uniform_gradient_hex8/

Table A-4. Additional Shell Element Verification Tests

Test Location
adagio_rtest/presto/vtest/sp3dq4_memb_patch_test_elastobad/
adagio_rtest/presto/vtest/ur3dq4_mi_memb_patch_test/
adagio_rtest/presto/vtest/kinbc_fixedrotationcomp_ur3dq4_so_twisted_beam/
adagio_rtest/presto/vtest/pressurized_ep_cyl_rfnd/
adagio_rtest/presto/vtest/pressurized_ep_cylinder/
adagio_rtest/shell_verification/vp18_pipe-on-pipe-impact/key_hoff_shell/
adagio_rtest/shell_verification/vp18_pipe-on-pipe-impact/mean_quadrature/
adagio_rtest/verification/vp01_beam-straight/enhanced_strain/
adagio_rtest/verification/vp01_beam-straight/key_hoff_shell/
adagio_rtest/verification/vp01_beam-straight/mean_quadrature/
adagio_rtest/verification/vp02_beam-twisted/enhanced_strain/
adagio_rtest/verification/vp02_beam-twisted/key_hoff_shell/
adagio_rtest/verification/vp02_beam-twisted/mean_quadrature/
adagio_rtest/verification/vp02_beam-twisted/seven_parameter/
adagio_rtest/verification/vp03_two-element-bending/mean_quadrature/
adagio_rtest/verification/vp04_hemisphere-diameter-loads/key_hoff_shell/
adagio_rtest/verification/vp05_raasch-hook/key_hoff_shell/
adagio_rtest/verification/vp05_raasch-hook/mean_quadrature/
adagio_rtest/verification/vp06_beam-to-ring/enhanced_strain/
adagio_rtest/verification/vp06_beam-to-ring/key_hoff_shell/
adagio_rtest/verification/vp06_beam-to-ring/mean_quadrature/
adagio_rtest/verification/vp09_ptest-sphere-inflation/enhanced_strain/
adagio_rtest/verification/vp09_ptest-sphere-inflation/key_hoff_shell/
adagio_rtest/verification/vp09_ptest-sphere-inflation/mean_quadrature/
adagio_rtest/verification/vp10_simply-supported-plate/key_hoff_shell/
adagio_rtest/verification/vp13_cylinder-pinched_diaphram/key_hoff_shell/
adagio_rtest/verification/vp15_hogg-plate/key_hoff_shell/
adagio_rtest/verification/vp19_layered-cantilever/

Table A-5. Additional Membrane Element Verification Tests

Test Location
adagio_rtest/adagio/ur3dm4_patch_test/
adagio_rtest/presto/vtest/ur3dm4_so_patch_test/

Table A-6. Additional Line Element Verification Tests

Test Location
adagio_rtest/adagio/truss_cosine_load/
adagio_rtest/presto/beam_timestep_baseline/
adagio_rtest/presto/damper_spring_critdamped/
adagio_rtest/presto/damper_spring_underdamped/
adagio_rtest/presto/truss_cosine_load/
adagio_rtest/presto/vtest/spring_on_axis_01el/
adagio_rtest/presto/vtest/spring_on_axis_05el/

Table A-7. Additional Specialty Element Verification Tests

Test Location
adagio_rtest/presto/spot_weld_test/
adagio_rtest/presto/spotweld_multiple/
adagio_rtest/presto/rbar/
adagio_rtest/presto/rigid_body_ellipsoid/
adagio_rtest/presto/rigid_body_from_attribute/
adagio_rtest/presto/rigid_body_general/
adagio_rtest/presto/rigid_body_kinbc/
adagio_rtest/presto/rigid_body_pmass_rotate/
adagio_rtest/presto/rigid_body_pmass_translate/
adagio_rtest/presto/rigid_body_principal/
adagio_rtest/presto/rigid_body_wedge/
adagio_rtest/presto/vtest/embedded_sph/

Table A-8. Additional Boundary Condition Verification Tests

Test Location
adagio_rtest/adagio/beam_bc_on_off/
adagio_rtest/presto/periodbc_block_offsethole/
adagio_rtest/presto/vtest/kinbc_presaccelcomp_baronaxis_sset/
adagio_rtest/presto/vtest/kinbc_presdisplcomp_baronaxis_sset/
adagio_rtest/presto/vtest/kinbc_presrotation_singleshell_mi/
adagio_rtest/presto/vtest/kinbc_presvelcomp_baronaxis/
adagio_rtest/presto/vtest/kinbc_presveldir_baroffaxis/
adagio_rtest/presto/vtest/kinbc_presvelrad_ringsect/
adagio_rtest/presto/vtest/fext_gravity_bar/
adagio_rtest/presto/vtest/fext_presforcecomp_beam/
adagio_rtest/presto/vtest/fext_presforcedir_beam/
adagio_rtest/presto/vtest/fext_presforcesub_beam/
adagio_rtest/presto/vtest/fext_presmomentcomp_singleshell_mi/
adagio_rtest/presto/vtest/fext_presmomentdir_singleshell_mi/
adagio_rtest/presto/vtest/fext_unifpress_baroffaxis_impulse/
adagio_rtest/presto/vtest/fext_unifpress_baronaxis_impulse/

Table A-9. Additional Miscellaneous Verification Tests

Test Location
adagio_rtest/abnormal_usage/distorted_elem/
adagio_rtest/abnormal_usage/distorted_elem_post_overlap_rem/
adagio_rtest/presto/vtest/mass_property_test1/
adagio_rtest/presto/vtest/mass_property_test2/
adagio_rtest/presto/vtest/vme5/
adagio_rtest/user_sub_lib/cavityVolume/
adagio_rtest/verification/V005/
adagio_rtest/verification/V063/
adagio_rtest/verification/VM18/
adagio_rtest/verification/VM27/
adagio_rtest/verification/VM7/
adagio_rtest/verification/VMC1/
adagio_rtest/verification/VME6/
adagio_rtest/verification/strain_cycle/imp_dynamics/

APPENDIX B. INPUT DECKS FOR VERIFICATION PROBLEMS

B.1. CONTACT FORCE BALANCE

See Section 2.1 for problem description.

```
begin sierra contact_force_balance

  begin function ramp
    type is analytic
    evaluate expression is "x <= 1.0 ? sin((pi/2.0)*x) : 1.0;"
  end

  begin material flubber
    density = 100.0
    begin parameters for model elastic
      youngs modulus = 30.0e6
      poissons ratio = 0.3
    end parameters for model elastic
  end material flubber

  begin finite element model contact_force_balance1
    database name = contact_force_balance.g
    database type = exodusII

    begin parameters for block block_1
      material = flubber
      model = elastic
    end

    begin parameters for block block_2
      material = flubber
      model = elastic
    end

  end finite element model contact_force_balance1

  begin adagio procedure fred

    begin time control
      begin time stepping block p1
        start time = 0.0
        begin parameters for adagio region adagio
          time increment = 0.1
        end
      end
      termination time = 1.0
    end

    begin adagio region adagio
      use finite element model contact_force_balance1

      begin results output fred
```

```
  database name = contact_force_balance.e
  database type = exodusII
  at step 0, increment = 10
  nodal variables = displacement
  nodal variables = velocity
  nodal variables = acceleration
  nodal variables = force_contact
  nodal variables = force_external
  nodal variables = force_internal
  nodal variables = reaction
  element variables = von_mises
  global variables = timestep
  global variables = external_energy
  global variables = internal_energy
  global variables = kinetic_energy
  global variables = momentum
  global variables = cont_press
  global variables = cont_fixed
  global variables = press_force
  global variables = fixreact
end

BEGIN HISTORY OUTPUT OUTPUT_adagio_HIS
  DATABASE NAME = contact_force_balance.h
  DATABASE TYPE = EXODUSII
  AT STEP 0, INCREMENT = 1
  VARIABLE = GLOBAL cont_press
  VARIABLE = GLOBAL cont_fixed
  VARIABLE = GLOBAL press_force
  VARIABLE = GLOBAL fixreact
  VARIABLE = GLOBAL err1
  VARIABLE = GLOBAL err2
  VARIABLE = GLOBAL err3
END HISTORY OUTPUT OUTPUT_adagio_HIS

begin fixed displacement
  block = block_1
  components = xyz
end

begin pressure
  surface = surface_1
  function = ramp
  scale factor = 1000.0
end

begin user output
  block = block_2
  compute global press_force as sum of nodal force_external
  compute at every step
end

begin user output
  block = block_1
  compute global cont_fixed as sum of nodal force_contact
  compute at every step
end

begin user output
  block = block_2
  compute global cont_press as sum of nodal force_contact
  compute at every step
end

begin user output
  block = block_1
  compute global fixreact as sum of nodal reaction
  compute at every step
```

```
      end

      begin user output
        compute global err1 from expression "abs(press_force[1] + cont_press[1]) + abs(press_force[2] + cont_p
        compute global err2 from expression "abs(cont_press [1] + cont_fixed[1]) + abs(cont_press [2] + cont_f
        compute global err3 from expression "abs(cont_fixed [1] + fixreact
[1]) + abs(cont_fixed [2] + fixreact  [2]) + abs(cont_fixed [3] + fixreact  [3])"
      end

      begin solution verification
        completion file = VerifErr
        verify global err1 = 0.0
        verify global err2 = 0.0
        verify global err3 = 0.0
        tolerance = 0.5 ## 0.05 % of Balanced
      end

      begin contact definition frictionless
        search      = dash
        enforcement = al
        contact surface surf_1 contains block_1
        contact surface surf_2 contains block_2
        begin interaction inter_1
          surfaces = surf_1 surf_2
          friction model = fric
        end interaction inter_1
         begin constant friction model fric
          friction coefficient = 0.1
         end
      end contact definition frictionless

      Begin solver
        level 1 predictor = none
        begin control contact
          target relative residual       = 1.0e-3
          Maximum Iterations             = 100
          Minimum Iterations             = 2
        end
        Begin cg
          acceptable relative residual = 1.0e10
          target relative residual     = 1.0e-5
          maximum iterations           = 500
          iteration print              = 50
          begin full tangent preconditioner
            minimum smoothing iterations = 25
            iteration update = 100
          end
        end
      end

    end
  end
end
```

B.2. HERTZ SPHERE-SPHERE CONTACT

See Section 2.2 for problem description.

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
## Parameters
# P       = 50000000 # Prescribed Force
# E       = 6.89e+10 # Young's Modulus
# nu      = 0.33 # Poisson's Ratio
# density = 1.024e-06 # Density
# Rstar   = 1 # Radius of Sphere
# numThetaCollections  = 8 # Number of Collection Points Around Circumference of Contact Patch
# numRadialCollections = 8 # Number of Collection Points Along Radius of Contact Patch

## Variables
# degreeInterval       = 45
# degreeTolerance      = 22.5
# Rmax                 = 0.08164645795 # Analytic
# radiusInterval       = 0.01020580724 # Computed from Rmax
# radiusTolerance      = 0.005102903622 # Computed from Rmax (but not used below)
# radiusErrorMax       = 4.5 # Command line input (percentage)
# radiusErrorMaxTol    = 0.5 # Command line input
# numRadialCollections = 9 # Add 1 for zero radius collection
# deflectErrorMax      = 9.75 # Command line input (percentage)
# deflectErrorMaxTol   = 1 # Command line input
# numSteps             = 4
# constraint           = face_face
# contactAlgorithm     = augmented Lagrange node-Face


begin sierra  Analysis of Hertz-Mindlin-Lubkin contact model

  title  Analysis of Hertz-Mindlin-Lubkin contact model

  define direction y with vector 0.0 1.0 0.0
  define direction x with vector 1.0 0.0 0.0
  define direction z with vector 0.0 0.0 1.0
  define point origin with coordinates 0.0 0.0 0.0

  #---------- Functions ---------

  begin definition for function load
    type is analytic
    evaluate expression = " 50000000 * x "
  end definition for function load

  begin definition for function analytic_radius
    type is analytic
    #expression variable: P = global load
    evaluate expression = " pow((0.75*50000000*x*1/6.89e+10),(1.0/3.0))"
  end

  begin definition for function analytic_compression
    type = analytic
    expression variable: a = global analytic_radius
    evaluate expression = "(a*a)/1"
  end

  begin definition for function ErrorCM
    type = analytic
    expression variable: ac = global analytic_compression
    expression variable: c = global compression
    evaluate expression = " (ac > 0.0) ? (abs( abs(c) - ac )/ac)  * 100.0 : 0.0 "
  end

  begin definition for function XYradius
    type = analytic
```

```
    expression variable: x = nodal model_coordinates(x)
    expression variable: y = nodal model_coordinates(y)
    expression variable: dx = nodal displacement(x)
    expression variable: dy = nodal displacement(y)
    evaluate expression = "sqrt((x+dx)^2+(y+dy)^2)"
  end

  begin definition for function angle
    type = analytic
    expression variable: x = nodal model_coordinates(x)
    expression variable: y = nodal model_coordinates(y)
    expression variable: dx = nodal displacement(x)
    expression variable: dy = nodal displacement(y)
    evaluate expression = " ( atan2((y+dy),(x+dx)) < 0.0 )  ?
atan2((y+dy),(x+dx))*(180.0/pi) + 360.0 : atan2((y+dy),(x+dx))*(180.0/pi) "
  end

  begin definition for function contact_radius
    type = analytic
    expression variable: cs = nodal contact_status
    expression variable: cnt = nodal contact_normal_traction_magnitude
    expression variable: radius = nodal XYradius
    evaluate expression = " ( cs > 0.9 && abs(cnt) > 0.0 && radius < 0.5 ) ? radius : 0.0"
  end

  begin definition for function analytic_pressure
    type is analytic
    expression variable: a = global analytic_radius
    expression variable: r = nodal XYradius
    expression variable: P = global load
    evaluate expression = "(r <= a) ? 1.5*P/(pi*a*a)*sqrt(1-(r/a)*(r/a)) : 0.0"
  end

  begin definition for function contact_pressure
    type is analytic
    expression variable: fx = nodal force_contact(x)
    expression variable: fy = nodal force_contact(y)
    expression variable: fz = nodal force_contact(z)
    expression variable: ca = nodal contact_area
    evaluate expression = " (ca > 0.0) ? (sqrt(fx^2+fy^2+fz^2)/ca) : 0.0"
  end

  #0
  #0
  begin definition for function contactRadius1
    type = analytic
    expression variable: angle = nodal angle
    expression variable: crad = nodal contact_radius
    evaluate expression = "  ( angle >= -22.5 &&  angle < 22.5 ) ? crad : 0.0 "
  end
  begin definition for function ErrorCR1
    type = analytic
    expression variable: acrad = global analytic_radius
    expression variable: crad = global MaxContactRadius1
    evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
  end
  # 45
  begin definition for function contactRadius2
    type = analytic
    expression variable: angle = nodal angle
    expression variable: crad = nodal contact_radius
    evaluate expression = "  ( angle >= 22.5 &&  angle < 67.5 ) ? crad : 0.0 "
  end
  begin definition for function ErrorCR2
    type = analytic
    expression variable: acrad = global analytic_radius
    expression variable: crad = global MaxContactRadius2
    evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
```

```
end
# 90
begin definition for function contactRadius3
  type = analytic
  expression variable: angle = nodal angle
  expression variable: crad = nodal contact_radius
  evaluate expression = "  ( angle >= 67.5 &&  angle < 112.5 ) ? crad : 0.0 "
end
begin definition for function ErrorCR3
  type = analytic
  expression variable: acrad = global analytic_radius
  expression variable: crad = global MaxContactRadius3
  evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
end
# 135
begin definition for function contactRadius4
  type = analytic
  expression variable: angle = nodal angle
  expression variable: crad = nodal contact_radius
  evaluate expression = "  ( angle >= 112.5 &&  angle < 157.5 ) ? crad : 0.0 "
end
begin definition for function ErrorCR4
  type = analytic
  expression variable: acrad = global analytic_radius
  expression variable: crad = global MaxContactRadius4
  evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
end
# 180
begin definition for function contactRadius5
  type = analytic
  expression variable: angle = nodal angle
  expression variable: crad = nodal contact_radius
  evaluate expression = "  ( angle >= 157.5 &&  angle < 202.5 ) ? crad : 0.0 "
end
begin definition for function ErrorCR5
  type = analytic
  expression variable: acrad = global analytic_radius
  expression variable: crad = global MaxContactRadius5
  evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
end
# 225
begin definition for function contactRadius6
  type = analytic
  expression variable: angle = nodal angle
  expression variable: crad = nodal contact_radius
  evaluate expression = "  ( angle >= 202.5 &&  angle < 247.5 ) ? crad : 0.0 "
end
begin definition for function ErrorCR6
  type = analytic
  expression variable: acrad = global analytic_radius
  expression variable: crad = global MaxContactRadius6
  evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
end
# 270
begin definition for function contactRadius7
  type = analytic
  expression variable: angle = nodal angle
  expression variable: crad = nodal contact_radius
  evaluate expression = "  ( angle >= 247.5 &&  angle < 292.5 ) ? crad : 0.0 "
end
begin definition for function ErrorCR7
  type = analytic
  expression variable: acrad = global analytic_radius
  expression variable: crad = global MaxContactRadius7
  evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
end
# 315
begin definition for function contactRadius8
```

```
    type = analytic
    expression variable: angle = nodal angle
    expression variable: crad = nodal contact_radius
    evaluate expression = "  ( angle >= 292.5 &&  angle < 337.5 ) ? crad : 0.0 "
  end
  begin definition for function ErrorCR8
    type = analytic
    expression variable: acrad = global analytic_radius
    expression variable: crad = global MaxContactRadius8
    evaluate expression = " acrad > 0.0 ? ( abs(crad - acrad) / acrad ) * 100.0 : 0.0 "
  end
  # 360


  begin definition for function ErrorCP
    type = analytic
    expression variable: acp = nodal analytic_pressure
    expression variable: cp = nodal contact_pressure
    expression variable: crad = nodal contact_radius
    expression variable: acrad = global analytic_radius
    evaluate expression = " (crad <= 0.75*acrad && cp > 0.0 ) ?  ( abs( cp - acp ) ) : 0.0 "
  end

  #---------- Materials ----------

  begin property specification for material mat_1
    density = 1.024e-06
    begin parameters for model elastic
      youngs modulus = 6.89e+10
      poissons ratio = 0.33
    end parameters for model elastic
  end property specification for material mat_1

#---------- Finite Element Model -----------

  begin finite element model hertz
    Database name = p03_Hertz_contact.g
    Database type = exodusII

    begin parameters for block block_1 block_10 block_1000
      material mat_1
      solid mechanics use model elastic
    end parameters for block block_1 block_10 block_1000

  end finite element model hertz

  begin adagio procedure procedure_1

    #---------- Time Step Control ----------

    begin time control
      begin time stepping block p0
        start time = 0.0
        begin parameters for adagio region region_1
          number of time steps = 4
        end parameters for adagio region region_1
      end time stepping block p0
      termination time = 1.0
    end time control

    begin adagio region region_1

      use finite element model hertz

      #--------- Contact ---------


      begin contact definition cont1
```

```
    search = dash
    contact surface skin_10 contains block_10
    contact surface skin_1 contains block_1
    contact surface skin_1000 contains block_1000
    enforcement = al
    begin interaction int_0
      friction model = frictionless
      master = skin_1000
      slave = skin_1
      constraint formulation = face_face
    end interaction int_0
    begin interaction int_1
      master = skin_10
      slave = skin_1
      friction model = tied
      constraint formulation = face_face
    end interaction int_1
    compute contact variables = on
  end contact definition cont1


  #--------- Boundary Conditions ---------

  begin fixed displacement
    block = block_10
    components = x y z
  end fixed displacement

  begin fixed displacement
    block = block_1000
    components = x y
  end fixed displacement

  begin prescribed force
    surface = surface_2
    component = z
    function = load
    scale factor = -0.25  ## 4 nodes
  end prescribed force

#--------- Results Output ---------

  begin user output
    surface = surface_3
    compute global load as function load
    compute global analytic_radius as function analytic_radius
    compute global analytic_compression as function analytic_compression
    compute global compression as min of nodal displacement(z)
    compute global ErrorCM as function ErrorCM
    compute nodal XYradius as function XYradius
    compute nodal angle as function angle
    compute nodal contact_radius as function contact_radius
    compute nodal analytic_pressure as function analytic_pressure
    compute nodal contact_pressure as function contact_pressure
    compute nodal ErrorCP as function ErrorCP
    compute global L2ErrorCP as l2norm of nodal ErrorCP
    #0
    compute nodal contactRadius1 as function contactRadius1
    compute global MaxContactRadius1 as max of nodal contactRadius1
    compute global ErrorCR1 as function ErrorCR1
    compute nodal contactRadius2 as function contactRadius2
    compute global MaxContactRadius2 as max of nodal contactRadius2
    compute global ErrorCR2 as function ErrorCR2
    compute nodal contactRadius3 as function contactRadius3
    compute global MaxContactRadius3 as max of nodal contactRadius3
    compute global ErrorCR3 as function ErrorCR3
    compute nodal contactRadius4 as function contactRadius4
    compute global MaxContactRadius4 as max of nodal contactRadius4
```

```
  compute global ErrorCR4 as function ErrorCR4
  compute nodal contactRadius5 as function contactRadius5
  compute global MaxContactRadius5 as max of nodal contactRadius5
  compute global ErrorCR5 as function ErrorCR5
  compute nodal contactRadius6 as function contactRadius6
  compute global MaxContactRadius6 as max of nodal contactRadius6
  compute global ErrorCR6 as function ErrorCR6
  compute nodal contactRadius7 as function contactRadius7
  compute global MaxContactRadius7 as max of nodal contactRadius7
  compute global ErrorCR7 as function ErrorCR7
  compute nodal contactRadius8 as function contactRadius8
  compute global MaxContactRadius8 as max of nodal contactRadius8
  compute global ErrorCR8 as function ErrorCR8
  compute at every step
end

begin results output output_1
  database name = p03_Hertz_contact.e
  at step 0 increment = 10
  nodal variables = displacement
  nodal variables = force_contact
  nodal variables = contact_tangential_direction as cdirtan
  nodal variables = contact_normal_direction as cdirnor
  nodal variables = contact_status as celement
  nodal variables = contact_normal_traction_magnitude as cfnor
  nodal variables = contact_tangential_traction_magnitude as cftan
  nodal variables = contact_area as carea
  global variables = total_iter as itotal
  global variables = analytic_radius
  global variables = load
  nodal variables = contact_radius
  nodal variables = analytic_pressure
  nodal variables = contact_pressure
  nodal variables = angle
  nodal variables = XYradius
  nodal variables = ErrorCP
  global variables = L2ErrorCP
  global variables = analytic_compression
  global variables = compression
  global variables = ErrorCM
  #0
  nodal variables =  contactRadius1
  global variables = MaxContactRadius1
  global variables = ErrorCR1
  nodal variables =  contactRadius2
  global variables = MaxContactRadius2
  global variables = ErrorCR2
  nodal variables =  contactRadius3
  global variables = MaxContactRadius3
  global variables = ErrorCR3
  nodal variables =  contactRadius4
  global variables = MaxContactRadius4
  global variables = ErrorCR4
  nodal variables =  contactRadius5
  global variables = MaxContactRadius5
  global variables = ErrorCR5
  nodal variables =  contactRadius6
  global variables = MaxContactRadius6
  global variables = ErrorCR6
  nodal variables =  contactRadius7
  global variables = MaxContactRadius7
  global variables = ErrorCR7
  nodal variables =  contactRadius8
  global variables = MaxContactRadius8
  global variables = ErrorCR8
end results output output_1

begin solution verification
```

```
        completion file = VerifContactRadius
        skip times = 0.0 to 0.999
        #0
        verify global ErrorCR1 = 4.5
        verify global ErrorCR2 = 4.5
        verify global ErrorCR3 = 4.5
        verify global ErrorCR4 = 4.5
        verify global ErrorCR5 = 4.5
        verify global ErrorCR6 = 4.5
        verify global ErrorCR7 = 4.5
        verify global ErrorCR8 = 4.5
        tolerance = 0.5
      end

      begin solution verification
        completion file = VerifContactCompression
        skip times = 0.0 to 0.999
        verify global ErrorCM = 9.75
        tolerance = 1
      end

      #--------- Solver ---------

      begin solver

        begin control contact
          target relative residual = 1.0e-3
          target relative contact residual = 1.0e-3
          maximum iterations = 100
        end
        begin cg
          target relative residual = 1.0e-5
          acceptable relative residual = 1.0e10
          maximum iterations = 100
          iteration print = 10
        end cg


      end solver

    end adagio region region_1

  end adagio procedure procedure_1

end sierra  Analysis of Hertz-Mindlin-Lubkin contact model
```

B.3. DERESIEWICZ SPHERE-SPHERE CONTACT

See Section 2.3 for problem description.

```
# jas2adagio translation from deresiewicz.i done on Wed Jun 25 12:57:52 2008

# ----------- Warning/Error/Information Message Help: -----------
# Numbers in parentheses () refer to JAS input file lines.
# Numbers in brackets [] refer to Adagio input file lines.
# ---------------------------------------------------------------

{include("materialParameters.i")}

begin sierra  Analysis of Hertz-Mindlin-Lubkin contact model

  title  Analysis of Hertz-Mindlin-Lubkin contact model

  define direction y with vector 0.0 1.0 0.0
  define direction x with vector 1.0 0.0 0.0
  define direction z with vector 0.0 0.0 1.0
  define point origin with coordinates 0.0 0.0 0.0

{include("computeTorque.i")}

  #---------- Materials ----------

  begin property specification for material mat_1
    density = 1.024E-6
    begin parameters for model elastic
      youngs modulus = 68900000000.0
      poissons ratio = 0.33
    end parameters for model elastic
  end property specification for material mat_1

  begin solid section solid_1
    strain incrementation = midpoint_increment
    hourglass rotation = scaled
  end solid section solid_1

  begin property specification for material mat_10
    density = 1.0
    begin parameters for model elastic
      youngs modulus = 110000.0
      poissons ratio = 0.3
    end parameters for model elastic
  end property specification for material mat_10

  begin solid section solid_10
    rigid body = 10
  end solid section solid_10

  begin rigid body 10
  end rigid body 10

  begin property specification for material mat_1000
    density = 1.0
    begin parameters for model elastic
      youngs modulus = 110000.0
      poissons ratio = 0.3
    end parameters for model elastic
  end property specification for material mat_1000

  begin solid section solid_1000
    rigid body = 1000
  end solid section solid_1000

  begin rigid body 1000
```

```
end rigid body 1000

#---------- Finite Element Model -----------

begin finite element model deresiewicz
  Database name = deresiewicz.g
  Database type = exodusII
  component separator character = ""

  begin parameters for block block_1
    material mat_1
    solid mechanics use model elastic
    section = solid_1
  end parameters for block block_1

  begin parameters for block block_10
    material mat_10
    solid mechanics use model elastic
    section = solid_10
  end parameters for block block_10

  begin parameters for block block_1000
    material mat_1000
    solid mechanics use model elastic
    section = solid_1000
  end parameters for block block_1000

end finite element model deresiewicz

begin adagio procedure procedure_1

  #---------- Time Step Control ----------

  begin time control
    begin time stepping block p0
      start time = 0.0
      begin parameters for adagio region region_1
        number of time steps = 10
      end parameters for adagio region region_1
    end time stepping block p0
    begin time stepping block p1
      start time = 0.01
      begin parameters for adagio region region_1
        number of time steps = 9
      end parameters for adagio region region_1
    end time stepping block p1
    begin time stepping block p2
      start time = 0.1
      begin parameters for adagio region region_1
        number of time steps = 9
      end parameters for adagio region region_1
    end time stepping block p2
    begin time stepping block p3
      start time = 1.0
      begin parameters for adagio region region_1
        number of time steps = 20
      end parameters for adagio region region_1
    end time stepping block p3
    termination time = 2.0
  end time control

  begin adagio region region_1

    use finite element model deresiewicz

    #--------- Contact ---------

    begin contact definition frictional
```

```
  search      = acme
  enforcement = frictional
  contact surface surf_2 contains surface_2
  contact surface surf_3 contains surface_3

  begin interaction int_0
    master = surf_2
    slave = surf_3
    normal tolerance = 1e-06
    tangential tolerance = 1e-08
    capture tolerance = 1e-06
    tension release = 0.0
    friction coefficient = 0.3
    friction coefficient function = function_1
  end interaction int_0

end contact definition frictional

begin contact definition tied
  search      = acme
  enforcement = tied
  contact surface surf_1 contains surface_1
  contact surface surf_4 contains surface_4

  begin interaction int_0
    master = surf_1
    slave = surf_4
    normal tolerance = 0.0001
    tangential tolerance = 0.0001
    capture tolerance = 0.0001
  end interaction int_0

end contact definition tied

#--------- Boundary Conditions ---------

begin fixed displacement
  block = block_10
  components = x y z
end fixed displacement

begin fixed displacement
  block = block_1000
  components = y
end fixed displacement

begin fixed rotation
  block = block_10
  components = x y z
end fixed rotation

begin fixed rotation
  block = block_1000
  components = x
end fixed rotation

begin fixed rotation
  block = block_1000
  components = y
end fixed rotation

begin prescribed displacement
  block = block_1000
  component = x
  function = function_120
  scale factor = 0.003
end prescribed displacement
```

```
      begin prescribed force
        block = block_1000
        component = z
        function = function_110
        scale factor = -50000000.0
      end prescribed force

      begin prescribed rotation
        block = block_1000
        direction = z
        function = function_1100
        scale factor = 0.05
      end prescribed rotation

      #--------- Results Output ---------

{include("outputTorque.i")}

      begin results output output_1
        database name = deresiewicz.e
        database type = exodusII
        component separator character = ""
        at time 0.0 increment = 1.0
        at step 28 increment = 1
        nodal variables = displacement as displ
        nodal variables = contact_tangential_direction as cdirtan
        nodal variables = contact_normal_direction as cdirnor
        nodal variables = contact_slip_direction_current as cdirslp
        nodal variables = contact_status as celement
        nodal variables = contact_normal_traction_magnitude as cfnor
        nodal variables = contact_tangential_traction_magnitude as cftan
        nodal variables = contact_slip_increment_current as cdtan
        nodal variables = contact_accumulated_slip as cstan
        nodal variables = contact_frictional_energy_density as cetan
        nodal variables = contact_area as carea
        global variables = total_iter as itotal
      end results output output_1

{include("svTorque.i")}

      #--------- Solver ---------

      begin solver
        begin loadstep predictor
          type = scale_factor
          scale factor = 1.0
          slip scale factor = 0.0
        end loadstep predictor

        begin control contact
          level = 1
          target relative residual = 0.00001
          maximum iterations = 99
        end control contact
        begin cg
          target relative residual = 0.000005
          minimum residual improvement = 0.5
          maximum iterations = 9759
          reset limits 293 14639 1000 0.5
          iteration print = 25
          preconditioner = diagonal
        end cg
      end solver
    end adagio region region_1

  end adagio procedure procedure_1

end sierra  Analysis of Hertz-Mindlin-Lubkin contact model
```

B.4. HERTZ CYLINDER-CYLINDER CONTACT – CONVERGENCE TEST

See Section 2.4 for problem description.

```
# ------------------------------------------------------------------------
#
# HertzContactOf2Cylinders Test
#
# ------------------------------------------------------------------------
#
# Aprepro default mesh value to help FCT
#{mesh="2"}
#
# Aprepro variable settings that are passed in from the test script
# shape: {shape}
# mesh: {mesh}
# elem_topo: {elem_topo}
# formulation: {formulation}
# Formulation flags: 0~off, 1~on
#   mean_quad: {mean_quad}
#   selective_dev: {selective_dev}
# strain_incrementation: {strain_incrementation}
# material_model: {material_model}
# delta_tolerance: {delta_tolerance}
# Solver flag: 0~off, 1~on
#   tangent_pre: {tangent_pre}
# Contact algorithm flags: 0~off, 1~on
#   node_face: {node_face}
#   face_face: {face_face}
#
# Aprepro variable settings that are common to all analyses
# termination_time: {termination_time = 0.02}
# Number of steps that works with tangent pre is 10
# number_steps: {number_steps = 5}
# epsilon_time is an offset from the termination_time used to ignore all results
# except those associated with the final time step.
# epsilon_time: {epsilon_time = termination_time/(number_steps*2)}
# Ey: {Ey = 100000.0}
# nu: {nu = 0.2}
# R: {R = 4.0}
# cylinder_length AKA disk thickness
# cylinder_length: {cylinder_length = 0.1/(2^(mesh-1))}
#
#-------------------------------------------------------------------------
#

begin sierra  Hertz2Cylinders

# Functions -----------------------------------------------------------

  # Linear time function for displacement
  begin function delta
    type is analytic
    expression variable: time = global time
    evaluate expression = "time"
  end function delta

# material data -------------------------------------------------------

  begin property specification for material Mat_1
    density = 1.0

    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
```

```
    end parameters for model {material_model}
  end property specification for material Mat_1

  begin property specification for material Mat_2
    density = 1.0

    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
    end parameters for model {material_model}
  end property specification for material Mat_2

# section data (required to create non-default hex elements) ---------

   begin solid section solid_section
     # strain incrementation = {strain_incrementation}
     formulation = {formulation}
     {Ifdef(selective_dev)}
     deviatoric parameter = 0.5
     {Else}
     # Undefined selective_dev
     {Endif}
    end solid section solid_section

# FE model ------------------------------------------------------------

  begin finite element model Hertz_mesh
    Database Name = {shape}_{elem_topo}_{mesh}.g
    Database Type = exodusII

    begin block defaults
      material = Mat_1
      model = {material_model}
      section = solid_section
      {Ifdef(mean_quad)}
      linear bulk viscosity = 0.06
      quadratic bulk viscosity = 1.20
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
      {Else}
      # Undefined mean_quad
      {Endif}
    end block defaults



    begin parameters for block block_1

    end parameters for block block_1

    begin parameters for block block_2
      material = Mat_2
    end parameters for block block_2

  end finite element model Hertz_mesh

# procedure data ------------------------------------------------------

  begin adagio procedure Hertz_Proc

    begin time control
      begin time stepping block linear_time
        start time = 0.0
        begin parameters for adagio region Hertz_Region
          time increment = {termination_time/number_steps}
        end parameters for adagio region Hertz_Region
      end time stepping block linear_time
      termination time = {termination_time}
```

```
    end time control

# region data ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    begin adagio region Hertz_Region
      use finite element model Hertz_mesh

# BC data ..................................................................

      # Prescribed through-plane displacements to give plane-strain conditions
      begin fixed displacement
        surface = plusZ_surface minusZ_surface
        component = z
      end

      # Prescribed vertical displacement on the bottom surface of bottom 1/2 cylinder
      begin prescribed displacement
        node set = btm_flat
        component = Y
        function = delta
        scale factor = 1.0
      end

      # Prescribed vertical displacement on the top surface of top 1/2 cylinder
      begin prescribed displacement
        node set = top_flat
        component = Y
        function = delta
        scale factor = -1.0
      end

      # prescribed displacements to prevent remaining rigid body motions
      # mid-points on top and bottom surfaces
      begin fixed displacement
        node set =  top_z_line  btm_z_line
        component = x
      end fixed displacement

# Contact parameters .......................................................

      begin contact definition
        search = dash
        enforcement = al  #augmented lagrange
        compute contact variables = on

        {Ifdef(face_face)}
        skin all blocks = on
        begin interaction cyl2cyl
          surfaces = btm_block  top_block
          # this is performing frictionless contact by default.
        end interaction cyl2cyl
        {Endif}

        {Ifdef(node_face)}
        contact node set top_node_set contains top_cylinder_nodes
        contact surface btm_surface contains btm_cylinder_sides
        search = dash
        begin interaction cyl2cyl
          master = btm_surface
          slave = top_node_set
        end interaction cyl2cyl
        {Endif}

      end contact definition

# Solver parameters ........................................................

      begin solver
```

```
      begin loadstep predictor
        type = scale_factor
        scale factor = 0.0 0.0
      end
      ${target_relative_residual = 1e-6}
      begin control contact
        target relative residual = {target_relative_residual}
        target residual          = 1.0e-10
        Maximum Iterations       = 100
        acceptable relative residual = 1e-4
        $ KHP: This really helps improve the contact convergence
        $ for this problem (From O(100) iterations down to O(10).
        lagrange adaptive penalty = uniform
      end
      begin cg
        target relative residual = {target_relative_residual/10.0}
        acceptable relative residual = 1.0e5
        {Ifdef(tangent_pre)}
        maximum iterations = 50
        begin full tangent preconditioner
          linear solver = feti
          conditioning  = no_check
          minimum smoothing iterations = 10
          small number of iterations = 45
        end
        {Else}
        maximum iterations = 1000
        orthogonality measure for reset = 0.1
        line search secant
        preconditioner = elastic
        {Endif}
      end cg
    end solver

# Output data ....................................................

    begin user output
      include all blocks
      extrapolate element variable stress to nodal variable nodal_stress
    end

    #begin user variable P
    #  type is global real
    #  global operator = max
    #end
    #begin user variable delta_analyt
    #  type is global real
    #  global operator = max
    #end
    #begin user variable delta_rel_error
    #  type is global real
    #  global operator = max
    #end

    begin user output
      node set = btm_flat
      # use reaction force on bottom of bottom cylinder
      compute global Pt as sum of nodal reaction(y)
      # analytical solution is in terms of force/length (P)
      compute global P from expression "Pt/{cylinder_length}"
      compute global delta_analyt from expression "(((1.0-{nu}^2)*P*(-1.0 + 2.0*Log((2.0*Sqrt(Pi)*{R})/Sqrt(
      compute global delta_rel_error from expression "abs(delta_analyt-time)/delta_analyt"
      # analytical solution for the contact radius -- used mainly initially to judge the adequacy of the mes
      compute global a from expression "(2.0*Sqrt((((1.0-{nu}^2)*P*{R})/{Ey})))/Sqrt(Pi)"
      compute global mesh from expression "{mesh}"
    end

    # heartbeat output does not currently support user defined variables, or does it
```

```
  # This file is for quickly examining the results
  begin heartbeat output convergence_visual_output
    stream name = {shape}_{mesh}_conv.txt
    global time
    global mesh
    global P
    global a
    global delta_analyt
    global delta_rel_error as delta_rel_error
  end heartbeat output convergence_visual_output

  # heartbeat file of displacement BC error for convergence analysis and plotting
  begin heartbeat output delta_convergence_tabular_output
    stream name = {shape}_{mesh}_conv_table.csv
    labels = off
    legend = off
    format = spyhis  # a csv file
    # start time = 0.0
    at step 1 increment = 1
    termination time = {termination_time}
    # global time  -- unnecessary with spyhis format
    global mesh
    global delta_rel_error as delta_rel_error
  end heartbeat output delta_convergence_tabular_output

  # heartbeat file of load and calculated radius for asymptotic convergence analysis and plotting
  begin heartbeat output load-radius_convergence_tabular_output
    stream name = {shape}_{mesh}_PnA_table.csv
    labels = off
    legend = off
    format = spyhis  # a csv file
    # start time = 0.0
    at step 1 increment = 1
    termination time = {termination_time}
    # global time  -- unnecessary with spyhis format
    global mesh
    global P as P
    global a as a
  end heartbeat output load-radius_convergence_tabular_output


  # Outputting Exodus Information
  begin Results Output output_adagio
    Database Name = {shape}_{elem_topo}_{mesh}_Adagio.e
    Database Type = exodusII
    At Time 0.0, Increment = 0.00001
    nodal variables = displacement as displ
    nodal variables = nodal_stress as nodal_stress
    nodal variables = force_contact as fc
    nodal variables = reaction as reaction
    nodal variables = velocity as vel
    element variables = stress as elem_stress
    global Variables = timestep
    global variables = mesh as mesh
    global variables = delta_analyt
    global variables = delta_rel_error
    global variables = P
  end results output output_adagio

  # note that we will probably not use the following if we are able to use a convergence criterion
  begin solution verification
    completion file = delta_verification
    skip times = 0.0 to {termination_time-epsilon_time}
    verify global delta_rel_error = 0.0
    tolerance = {delta_tolerance}
  end

end adagio region Hertz_Region
```

```
  end adagio procedure Hertz_Proc

  begin feti equation solver feti
  end

end sierra Hertz2Cylinders
```

B.5. MINDLIN CYLINDER-CYLINDER CONTACT – CONVERGENCE TEST

See Section 2.5 for problem description.

```
# ------------------------------------------------------------------
#
# MindlinContactOf2Cylinders Test
#
# This first version of the test uses a single procedure but two time blocks.
# It is sufficient if the friction forces developed during the normal relative
# displacment do not have significant tangent components.
#
# ------------------------------------------------------------------
#
# Aprepro default mesh value to help FCT
#{mesh="2"}
#
# Aprepro variable settings that are passed in from the test script
# shape: {shape}
# mesh: {mesh}
# elem_topo: {elem_topo}
# formulation: {formulation}
# Formulation flags: 0~off, 1~on
#   mean_quad: {mean_quad}
#   selective_dev: {selective_dev}
# strain_incrementation: {strain_incrementation}
# material_model: {material_model}
# delta_tolerance: {delta_tolerance}
# Solver flag: 0~off, 1~on
#   tangent_pre: {tangent_pre}
# Contact algorithm flags: 0~off, 1~on
#   node_face: {node_face}
#   face_face: {face_face}
# Torque force flags: 0~off, 1~on
#   reaction_forces: {reaction_forces=0}
#   contact_forces:  {contact_forces=1}
#
# Aprepro variable settings that are common to all analyses
# preload_termination_time: {preload_termination_time = 0.02}
# shear_time_increment: {shear_time_increment = 0.01}
# shear_termination_time: {shear_termination_time =
preload_termination_time + shear_time_increment}
# preload_number_steps: {preload_number_steps = 20}
# shear_number_steps:   {shear_number_steps = 20}
# epsilon_time is an offset from the termination_time used to ignore all results
# except those associated with the final time step.
# epsilon_time: {epsilon_time = shear_time_increment/(shear_number_steps*2)}
#
# Ey: {Ey = 100000.0}
# nu: {nu = 0.2}
# R: {R = 4.0}
# cylinder_length AKA disk thickness
# cylinder_length: {cylinder_length = 0.1/(2^(mesh-1))}
#
#-------------------------------------------------------------------
#

begin sierra  Mindlin2Cylinders

# Functions ---------------------------------------------------------

  # Linear time function (during first time period) for normal displacement
  begin function delta_preload
    type is analytic
    expression variable: time = global time
```

```
    evaluate expression = "(time<{preload_termination_time}) ? time : {preload_termination_time}"
  end function delta_preload

  # Linear time function (during second time period) for lateral (shear) displacement
  begin function delta_shear
    type is analytic
    expression variable: time = global time
    evaluate expression = "(time>{preload_termination_time}) ? time-{preload_termination_time} : 0.0"
  end function delta_shear

# material data ------------------------------------------------------

  begin property specification for material Mat_1
    density = 1.0

    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
    end parameters for model {material_model}
  end property specification for material Mat_1

  begin property specification for material Mat_2
    density = 1.0

    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
    end parameters for model {material_model}
  end property specification for material Mat_2

# section data (required to create non-default hex elements) ---------

   begin solid section solid_section
     # strain incrementation = {strain_incrementation}
     formulation = {formulation}
     {Ifdef(selective_dev)}
     deviatoric parameter = 0.5
     {Else}
     # Undefined selective_dev
     {Endif}
    end solid section solid_section

# FE model -----------------------------------------------------------

  begin finite element model Mindlin_mesh
    Database Name = {shape}_{elem_topo}_{mesh}.g
    Database Type = exodusII

    begin parameters for block block_1
      material Mat_1
      solid mechanics use model {material_model}
      section = solid_section
      {Ifdef(mean_quad)}
      linear bulk viscosity = 0.06
      quadratic bulk viscosity = 1.20
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
      {Else}
      # Undefined mean_quad
      {Endif}
    end parameters for block block_1

    begin parameters for block block_2
      material Mat_2
      solid mechanics use model {material_model}
      section = solid_section
      {Ifdef(mean_quad)}
      linear bulk viscosity = 0.06
```

```
      quadratic bulk viscosity = 1.20
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
      {Else}
      # Undefined mean_quad
      {Endif}
    end parameters for block block_2
  end finite element model Mindlin_mesh

# procedure data ------------------------------------------------------

  begin adagio procedure Mindlin_Proc

    begin time control
      begin time stepping block preload_time
        start time = 0.0
        begin parameters for adagio region Mindlin_Region
          time increment = {preload_termination_time/preload_number_steps}
        end parameters for adagio region Mindlin_Region
      end time stepping block preload_time
      begin time stepping block shear_time
        start time = {preload_termination_time}
        begin parameters for adagio region Mindlin_Region
          time increment = {(shear_termination_time-preload_termination_time)/shear_number_steps}
        end parameters for adagio region Mindlin_Region
      end time stepping block shear_time
      termination time = {shear_termination_time}
    end time control

# region data ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    begin adagio region Mindlin_Region
      use finite element model Mindlin_mesh

# BC data ..............................................................

      # Prescribed through-plane displacements to give plane-strain conditions
      begin fixed displacement
        surface = plusZ_surface minusZ_surface
        component = z
      end

      # Prescribed vertical displacement on the bottom surface of bottom 1/2 cylinder
      begin prescribed displacement
        node set = btm_flat
        component = Y
        function = delta_preload
        scale factor = 1.0
      end

      # Prescribed vertical displacement on the top surface of top 1/2 cylinder
      begin prescribed displacement
        node set = top_flat
        component = Y
        function = delta_preload
        scale factor = -1.0
      end

      # prescribed displacements to prevent remaining rigid body motions
      # mid-points on top and bottom surfaces
      begin fixed displacement
        active periods = preload_time
        node set =  top_z_line  btm_z_line
        component = x
      end fixed displacement

      # apply horizontal (shear) displacements to top and bottom surfaces
      # Prescribed horizontal displacement on the bottom surface of bottom 1/2 cylinder
```

```
      begin prescribed displacement
        node set = btm_flat
        component = x
        function = delta_shear
        scale factor = -1.0
      end

      # Prescribed horizontal displacement on the top surface of top 1/2 cylinder
      begin prescribed displacement
        node set = top_flat
        component = x
        function = delta_shear
        scale factor = 1.0
      end

# Contact parameters ..............................................

      begin contact definition
        search = dash
        enforcement = al  #augmented lagrange
        compute contact variables = on

         begin constant friction model a_friction
          friction coefficient = 0.3
        end constant friction model a_friction

        {Ifdef(face_face)}
        skin all blocks = on
        begin interaction cyl2cyl
          surfaces = btm_block  top_block
          # this is performing frictionless contact by default.
          friction model = a_friction
        end interaction cyl2cyl
        {Endif}

        {Ifdef(node_face)}
        contact node set top_node_set contains top_cylinder_nodes
        contact surface btm_surface contains btm_cylinder_sides
        search = dash
        begin interaction cyl2cyl
          master = btm_surface
          slave = top_node_set
          friction model = a_friction
        end interaction cyl2cyl
        {Endif}

      end contact definition

# Solver parameters ...............................................

      begin solver
        begin loadstep predictor
          type = scale_factor
          scale factor = 0.0
        end
        ${target_relative_residual = 1.0e-7}
        begin control contact
          target relative residual     = {target_relative_residual}
          acceptable relative residual = 0.01
          target residual          = 1.0e-10
          Maximum Iterations       = 100
        end
        begin cg
          target relative residual = {target_relative_residual / 10.0 }
          {Ifdef(tangent_pre)}
          maximum iterations = 50
          begin full tangent preconditioner
            linear solver = feti
```

```
            conditioning  = no_check
            minimum smoothing iterations = 10
            small number of iterations = 45
          end
          {Else}
          maximum iterations = 1000
          orthogonality measure for reset = 0.1
          line search secant
          preconditioner = elastic
          {Endif}
        end cg
      end solver

# Output data ....................................................

      begin user output
        include all blocks
        extrapolate element variable stress to nodal variable nodal_stress
      end

      #begin user variable P
      #  type is global real
      #  global operator = max
      #end
      #begin user variable delta_analyt
      #  type is global real
      #  global operator = max
      #end
      #begin user variable delta_rel_error
      #  type is global real
      #  global operator = max
      #end

      begin user output
        node set = btm_flat
        # use reaction force on bottom of bottom cylinder
        compute global Qt as sum of nodal reaction(x)
        compute global Pt as sum of nodal reaction(y)
        # analytical solution is in terms of force/length (q)
        compute global P from expression "Pt/{cylinder_length}"
        compute global Q from expression "Qt/{cylinder_length}"
        #compute global delta_analyt from expression "(((1.0-{nu}^2)*P*(-1.0 + 2.0*Log((2.0*Sqrt(Pi)*{R})/Sqrt
        #compute global delta_rel_error from expression "abs(delta_analyt-time)/delta_analyt"
        # analytical solution for the contact radius -- used mainly initially to judge the adequacy of the mes
        #compute global a from expression "(2.0*Sqrt((((1.0-{nu}^2)*P*{R})/{Ey})))/Sqrt(Pi)"
        compute global mesh from expression "{mesh}"
      end

      # heartbeat output does not currently support user defined variables, or does it
      # This file is for quickly examining the results
      begin heartbeat output convergence_visual_output
        stream name = {shape}_{mesh}_conv.txt
#        precision = 9
        global time
        global mesh
        global P
        global Q
        #global a
        #global delta_analyt
        #global delta_rel_error as delta_rel_error
      end heartbeat output convergence_visual_output

      # heartbeat file of displacement BC error for convergence analysis and plotting
      #begin heartbeat output delta_convergence_tabular_output
      # stream name = {shape}_{mesh}_conv_table.csv
      #  precision = 16
      #  labels = off
      #  legend = off
```

```
      #  format = spyhis  # a csv file
      #  start time = {preload_termination_time}
      #  at step 1 increment = 1
      #  termination time = {shear_termination_time}
        # global time  -- unnecessary with spyhis format
      #  global mesh
        #global delta_rel_error as delta_rel_error
      #end heartbeat output delta_convergence_tabular_output

      # heartbeat file of load and calculated radius for asymptotic convergence analysis and plotting
      begin heartbeat output load-radius_convergence_tabular_output
        stream name = {shape}_{mesh}_Q_table.csv
#        precision = 8
        labels = off
        legend = off
        format = spyhis  # a csv file
        start time = {preload_termination_time}
        at step 1 increment = 1
        termination time = {shear_termination_time}
        # global time  -- unnecessary with spyhis format
        global mesh
        global Q as Q
        #global a as a
      end heartbeat output load-radius_convergence_tabular_output


      # Outputting Exodus Information
      begin Results Output output_adagio
        Database Name = {shape}_{elem_topo}_{mesh}_Adagio.e
        Database Type = exodusII
        At Time 0.0, Increment = 0.00001
        nodal variables = displacement as displ
        nodal variables = nodal_stress as nodal_stress
        nodal variables = force_contact as fc
        nodal variables = reaction as reaction
        nodal variables = velocity as vel
        element variables = stress as elem_stress
        global Variables = timestep
        global variables = mesh as mesh
        global variables = delta_analyt
        global variables = delta_rel_error
        global variables = P
        global variables = Q
      end results output output_adagio

      # note that we will probably not use the following if we are able to use a convergence criterion
      #begin solution verification
      #  completion file = delta_verification
      #  skip times = 0.0 to {shear_termination_time-epsilon_time}
      #  verify global delta_rel_error = 0.0
      #  tolerance = {delta_tolerance}
      #end

    end adagio region Mindlin_Region
  end adagio procedure Mindlin_Proc

  begin feti equation solver feti
  end

end sierra Mindlin2Cylinders
```

B.6. HERTZ SPHERE-SPHERE CONTACT – CONVERGENCE TEST

See Section 2.6 for problem description.

```
# ------------------------------------------------------------------
#
# Hertz contact of 2 half-spheres test
#
# ------------------------------------------------------------------
#
# Aprepro default mesh value to help FCT
#
# Aprepro variable settings that are passed in from the test script
# shape: {shape}
# mesh: {mesh}
# elem_topo: {elem_topo}
# formulation: {formulation}
# Formulation flags: 0~off, 1~on
#   mean_quad: {mean_quad}
#   selective_dev: {selective_dev}
# strain_incrementation: {strain_incrementation}
# material_model: {material_model}
# Solver flag: 0~off, 1~on
#   tangent_pre: {tangent_pre}
# Contact algorithm flags: 0~off, 1~on
#   node_face: {node_face}
#   face_face: {face_face}
# Torque force flags: 0~off, 1~on
#   reaction_forces: {reaction_forces=0}
#   contact_forces:  {contact_forces=1}
#
# Aprepro variable settings that are common to all analyses
# Original load level for preload was 0.0020 -- insufficient to use contact region
# For mesh 1: first "circular" mesh ring is in contact at a load level of 0.018 (to 2 digits)
# For mesh 2: first "circular" mesh ring is in contact at a load level of 0.022 (to 2 digits)
# For mesh 3: first "circular" mesh ring is in contact at a load level of 0.02? (to 2 digits)
# Load level of 0.023 could give consistent contact with all three meshes -- not checked yet.
# preload_termination_time: {preload_termination_time = 0.024}
# preload_number_steps: {preload_number_steps = 30}
# Ey: {Ey = 100000.0}
# nu: {nu = 0.2}
# R: {R = 4.0}
# G: {G = 41666.66667}
#------------------------------------------------------------------

begin sierra  Hertz

# Geometric entities used to define a "rotational displacement field"--

  define point origin with coordinates 0.0  0.0  0.0
  define direction y_direction with vector 0.0  1.0  0.0

# Functions ---------------------------------------------------------

  # Linear time function (during first time period) for normal displacement
  begin function delta_preload
    type is analytic
    expression variable: time = global time
    evaluate expression = "(time<{preload_termination_time}) ? time : {preload_termination_time}"
  end function delta_preload

# material data -----------------------------------------------------

  begin property specification for material Mat_1
    density = 1.0
```

```
    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
    end parameters for model {material_model}
  end property specification for material Mat_1

  begin property specification for material Mat_2
    density = 1.0
    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
    end parameters for model {material_model}
  end property specification for material Mat_2

# section data (required to create non-default hex elements) ---------

   begin solid section solid_section
     # strain incrementation = {strain_incrementation}
     formulation = {formulation}
     {Ifdef(selective_dev)}
     deviatoric parameter = 0.5
     {Else}
     # Undefined selective_dev
     {Endif}
    end solid section solid_section

# FE model ------------------------------------------------------------

  begin finite element model Half-spheres_mesh
    Database Name = {shape}_{elem_topo}_{mesh}.g
    Database Type = exodusII

    begin parameters for block block_1
      material Mat_1
      solid mechanics use model {material_model}
      section = solid_section
      {Ifdef(mean_quad)}
      linear bulk viscosity = 0.06
      quadratic bulk viscosity = 1.20
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
      {Else}
      # Undefined mean_quad
      {Endif}
    end parameters for block block_1

    begin parameters for block block_2
      material Mat_2
      solid mechanics use model {material_model}
      section = solid_section
      {Ifdef(mean_quad)}
      linear bulk viscosity = 0.06
      quadratic bulk viscosity = 1.20
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
      {Else}
      # Undefined mean_quad
      {Endif}
    end parameters for block block_2

  end finite element model Half-spheres_mesh

# procedure data for initial (Hertz) compression ----------------------------------------

  begin adagio procedure Compression_Proc

    begin time control
      begin time stepping block preload_time
```

```
    start time = 0.0
    begin parameters for adagio region Compression_Region
      time increment = {preload_termination_time/preload_number_steps}
    end parameters for adagio region Compression_Region
  end time stepping block preload_time
  termination time = {preload_termination_time}
end time control

# region data :::::::::::::::::::::::::::::::::::::::::::::::::::::::::

begin adagio region Compression_Region
  use finite element model Half-spheres_mesh

  # BC data ..................................................................

  # Fix the x-component of these two points which are along the
  # x=0 lines of the flat faces, and at z=-4
  # constrains the displacement to be radial
  begin fixed displacement
    active periods = preload_time
    node set = NodeCircumferencet NodeCircumferenceb
    component = x
  end

  # Prescribed vertical displacement on the face of the top hemisphere
  begin prescribed displacement
    node set = NodeFacet
    component = y
    function = delta_preload
    scale factor = -1.0
  end

  # Prescribed vertical displacement on the face of the bottom hemisphere
  begin prescribed displacement
    node set = NodeFaceb
    component = y
    function = delta_preload
    scale factor = 1.0
  end

  # Fix the x and z components of the center node of each hemisphere to
  # constrain the center point to displace normal to the plane
  begin fixed displacement
    node set =  NodeCentert NodeCenterb
    components = x z
  end fixed displacement

  # Additional fixed displacement placed on the inital contact nodes to prevent planar movement
  begin fixed displacement
    node set  = ContactNodet ContactNodeb
    components = x z
  end fixed displacement

  # Contact parameters .....................................................

  begin contact definition
    search      = dash
    enforcement = al   #augmented lagrange
    compute contact variables = on

    # We originally used time dependent friction model here, but that did not work in Adagio
    # thus the need for two procedures in this version of the tests.

    begin frictionless model hertz_friction_free
    end frictionless model hertz_friction_free

    {Ifdef(face_face)}
    skin all blocks = on
```

```
  begin interaction sphere2sphere
    surfaces = block_1  block_2
    normal tolerance = 1e-06
    friction model = hertz_friction_free
  end interaction sphere2sphere
  {Endif}

  {Ifdef(node_face)}
  contact node set top_node_set contains NodeContactt
  contact surface btm_surface contains SurfContactb
  begin interaction sphere2sphere
    master = btm_surface
    slave =  top_node_set
    normal tolerance = 1e-06
    friction model = hertz_friction_free
  end interaction sphere2sphere
  {Endif}

end contact definition

# Solver parameters ..............................................

begin solver
  begin control contact
    target relative residual = 1.0e-6
    target residual          = 1.0e-6
    acceptable relative residual = 1.0e-3
  end
  begin cg
    target relative residual = 1.0e-7
    target residual          = 1.0e-8
    maximum iterations       = 50
    acceptable relative residual = 1.0
    begin full tangent preconditioner
      linear solver = feti
      conditioning  = no_check
      minimum smoothing iterations = 10
      small number of iterations = 45
    end
  end
end solver

# Output data ....................................................

{ifdef(extrapolate)}
begin user output
  include all blocks
  extrapolate element variable stress to nodal variable nodal_stress
end
{endif}

begin user output
  node set = NodeFacet
  # use reaction force on top of top sphere
  compute global P as sum of nodal reaction(y)
  # First set of user outputs is for the Hertz part of the solution -- the preload
  compute global delta_analyt from expression "3^(2/3)*((-1+{nu}^2)^2*P^2/({Ey}^2*{R}))^(1/3)/(2^(1/3))"
  compute global delta_rel_error from expression "abs(delta_analyt-2*time)/delta_analyt"
  # analytical solution for the contact radius -- used mainly initially to judge the adequacy of the mes
  compute global a from expression "3^(1/3)*((-1+{nu}^2)*P*{R}/{Ey})^(1/3)/(2^(2/3))"
  compute global mesh from expression "{mesh}"
end

begin heartbeat output convergence_visual_output
  stream name = {shape}_{mesh}_conv.txt
  global P
  global a
  global delta_analyt
```

```
      global delta_rel_error
    end heartbeat output convergence_visual_output

    begin heartbeat output convergence_tabular_output
      stream name = {shape}_{mesh}_P_table.csv
      labels = off
      legend = off
      format = spyhis  # a csv file
      At Step 1, increment = 1
      termination time = {preload_termination_time}
      global mesh
      global P
      global a
    end heartbeat output convergence_tabular_output

    # heartbeat file for plotting or convergence calculations
    begin heartbeat output convergence_tabular_output
      stream name = {shape}_{mesh}_conv_table.csv
      labels = off
      legend = off
      format = spyhis  # a csv file
       At Step 1, increment = 1
      termination time = {preload_termination_time}
      global mesh
      global delta_rel_error
    end heartbeat output convergence_tabular_output

    # Outputting Exodus Information
    begin Results Output output_adagio
      Database Name = {shape}_{elem_topo}_{mesh}_compress.e
      Database Type = exodusII
      At Step 0, Increment = 1
      nodal variables = coordinates
      nodal variables = contact_status
      nodal variables = contact_incremental_slip_magnitude as contact_magnitude
      nodal variables = contact_area
      nodal variables = displacement as displ
      nodal variables = nodal_stress
      nodal variables = force_contact as fc
      nodal variables = reaction
      nodal variables = force_internal
      nodal variables = force_external
      element variables = stress as elem_stress
      global Variables = timestep
      global variables = mesh
      global variables = delta_analyt
      global variables = delta_rel_error
      global variables = P
      global variables = a
    end results output output_adagio

  end adagio region Compression_Region
end adagio procedure Compression_Proc

begin feti equation solver feti
end

end sierra Hertz
```

B.7. LUBKIN SPHERE-SPHERE CONTACT – CONVERGENCE TEST

See Section 2.7 for problem description.

```
# ----------------------------------------------------------------------
#
# Lubkin contact of 2 half-spheres test
#
# ----------------------------------------------------------------------
#
# Aprepro default mesh value to help FCT
#{mesh="2"}
#
# Aprepro variable settings that are passed in from the test script
# shape: {shape}
# mesh: {mesh}
# elem_topo: {elem_topo}
# formulation: {formulation}
# Formulation flags: 0~off, 1~on
#   mean_quad: {mean_quad}
#   selective_dev: {selective_dev}
# strain_incrementation: {strain_incrementation}
# material_model: {material_model}
# torque_tolerance: {torque_tolerance}
# Solver flag: 0~off, 1~on
#   tangent_pre: {tangent_pre}
# Contact algorithm flags: 0~off, 1~on
#   node_face: {node_face}
#   face_face: {face_face}
# Torque force flags: 0~off, 1~on
#   reaction_forces: {reaction_forces=0}
#   contact_forces:  {contact_forces=1}
#
# Aprepro variable settings that are common to all analyses
# Original load level for preload was 0.0020 -- insufficient to use contact region
# For mesh 1: first "circular" mesh ring is in contact at a load level of 0.018 (to 2 digits)
# For mesh 2: first "circular" mesh ring is in contact at a load level of 0.022 (to 2 digits)
# For mesh 3: first "circular" mesh ring is in contact at a load level of 0.02? (to 2 digits)
# Load level of 0.023 could give consistent contact with all three meshes -- not checked yet.
# preload_termination_time: {preload_termination_time = 0.024}
# Use increment of 0.01 to examine transition from stick to slip
# twist_time_increment: {twist_time_increment = 0.01}
# twist_termination_time: {twist_termination_time =
preload_termination_time + twist_time_increment}
# For node-face contact (using either a fully integrated or mean-quadrature integrated element)
# we could obtain convergence in 20 step per time block. Face-face contact with the fully-integrated
# element forced the smaller time stepping. 40 steps worked for face-face mean-quadrature.
# With 60 steps for preload, face-full-mesh2 run only made it to step 45.
# preload_number_steps: {preload_number_steps = 30}
# 60 steps are not sufficient for face-face contact with fully-integrated hex
# twist_number_steps:   {twist_number_steps = 30}
# epsilon_time is an offset from the termination_time used to ignore all results
# except those associated with the final time step.
# epsilon_time: {epsilon_time = twist_time_increment/(twist_number_steps*2)}
# Ey: {Ey = 100000.0}
# nu: {nu = 0.2}
# R: {R = 4.0}
# G: {G = 41666.66667}
# Constants used in Pades Expression
# a0: {a0 = 0}
# a1: {a1 = 5.3333}
# a2: {a2 = 6.0327}
# a3: {a3 = 19.6951}
# a4: {a4 = 42.5359}
# b0: {b0 = 1}
```

```
# b1: {b1 = 5.1193}
# b2: {b2 = 15.6833}
# b3: {b3 = 30.8099}
# b4: {b4 = 72.2111}
# Absolute rotations of each half sphere, dummy values for now
# at: {at = 0}
# ab: {ab = 0}
# Frictions coefficient used when contact occurs
# mu: {mu = 0.30}
# Rigid body option
# rigid_body: {rigid_body = 0}
# Procedure type -- initially only applied to twist loading and only
# defined in the input file for preliminary parameter studies
# proc_type: {proc_type = "adagio"}
# Turning on the next variable reveals a code error as of 7/24/13.
# Once resolved this variable and corresponding aprepro variables
# can be removed.
# extrapolate: {extrapolate = 0}

#-----------------------------------------------------------------------

begin sierra  Lubkin

# Geometric entities used to define a "rotational displacement field"--

  define point origin with coordinates 0.0  0.0  0.0
  define direction y_direction with vector 0.0  1.0  0.0
  define axis cylindrical_z_axis with point origin direction y_direction

# Functions -----------------------------------------------------------

  # Linear time function (during first time period) for normal displacement
  begin function delta_preload
    type is analytic
    expression variable: time = global time
    evaluate expression = "(time<{preload_termination_time}) ? time : {preload_termination_time}"
  end function delta_preload

  # Linear time function (during second time period) for rotational (torsional) displacement
  begin function theta_twist
    type is analytic
    expression variable: time = global time
    evaluate expression = "(time>{preload_termination_time}) ? time-{preload_termination_time} : 0.0"
  end function theta_twist

  # This approach did not work in contact.
  # Shifted Heaviside function to turn friction on after the normal preloading
  # begin function friction_switch
  #   type is analytic
  #   expression variable: time = global time
  #   evaluate expression = "(time>{preload_termination_time}) ? 1.0 : 0.0"
  # end function friction_switch

  # y-component of Torque due to one node on the top sphere cut-plane (actually any y-plane)
  # y-component of the vector cross product that yields Torque (F cross x).
  # Use of coordinates (not model_coordinates) => correctly attaining torque in the deformed configuration.
  # This funciton assumes that the center of the cut plane is at x,z=0.
  # Aprepro variable torque_force defines which forces are used to calculate the torque.
  begin function node_torque
    type is analytic
    expression variable: x1 = nodal coordinates(x)
    expression variable: x3 = nodal coordinates(z)
    {Ifdef(reaction_forces)}
    expression variable: F1 = nodal reaction(x)
    expression variable: F3 = nodal reaction(z)
    # assumes reaction force on the body not on the "restraining plane"
    evaluate expression = "(x3*F1-x1*F3)"
    {Endif}
```

```
    {Ifdef(contact_forces)}
    expression variable: F1 = nodal force_contact(x)
    expression variable: F3 = nodal force_contact(z)
    # using top sphere => sign change but usable with sphere on plane case
    evaluate expression = "(x3*F1-x1*F3)"
    {Endif}
  end function node_torque

# material data ------------------------------------------------------

  begin property specification for material Mat_1
    density = 1.0
    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
    end parameters for model {material_model}
  end property specification for material Mat_1

  begin property specification for material Mat_2
    density = 1.0
    begin parameters for model {material_model}
      youngs modulus = {Ey}
      poissons ratio = {nu}
    end parameters for model {material_model}
  end property specification for material Mat_2

# section data (required to create non-default hex elements) ---------

   begin solid section solid_section
     # strain incrementation = {strain_incrementation}
     formulation = {formulation}
     {Ifdef(selective_dev)}
     deviatoric parameter = 0.5
     {Else}
     # Undefined selective_dev
     {Endif}
    end solid section solid_section

    {Ifdef(rigid_body)}
    begin rigid body rb_1
      include nodes in surface_3
    end rigid body rb_1

    begin rigid body rb_2
      include nodes in surface_4
    end rigid body rb_2
    {Endif}

# FE model -----------------------------------------------------------

  begin finite element model Half-spheres_mesh
    Database Name = {shape}_{elem_topo}_{mesh}.g
    Database Type = exodusII

    begin parameters for block block_1
      material Mat_1
      solid mechanics use model {material_model}
      section = solid_section
      {Ifdef(mean_quad)}
      linear bulk viscosity = 0.06
      quadratic bulk viscosity = 1.20
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
      {Else}
      # Undefined mean_quad
      {Endif}
    end parameters for block block_1
```

```
    begin parameters for block block_2
      material Mat_2
      solid mechanics use model {material_model}
      section = solid_section
      {Ifdef(mean_quad)}
      linear bulk viscosity = 0.06
      quadratic bulk viscosity = 1.20
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
      {Else}
      # Undefined mean_quad
      {Endif}
    end parameters for block block_2

  end finite element model Half-spheres_mesh

# procedure data for initial (Hertz) compression ----------------------------------------

  begin adagio procedure Compression_Proc

    begin time control
      begin time stepping block preload_time
        start time = 0.0
        begin parameters for adagio region Compression_Region
          time increment = {preload_termination_time/preload_number_steps}
        end parameters for adagio region Compression_Region
      end time stepping block preload_time
      termination time = {preload_termination_time}
    end time control

    # region data ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    begin adagio region Compression_Region
      use finite element model Half-spheres_mesh

      # BC data ..................................................................

      # Fix the x-component of these two points which are along the
      # x=0 lines of the flat faces, and at z=-4
      # constrains the displacement to be radial
      begin fixed displacement
        active periods = preload_time
        node set = NodeCircumferencet NodeCircumferenceb
        component = x
      end

      # Prescribed vertical displacement on the face of the top hemisphere
      begin prescribed displacement
        node set = NodeFacet
        component = y
        function = delta_preload
        scale factor = -1.0
      end

      # Prescribed vertical displacement on the face of the bottom hemisphere
      begin prescribed displacement
        node set = NodeFaceb
        component = y
        function = delta_preload
        scale factor = 1.0
      end

      # Fix the x and z components of the center node of each hemisphere to
      # constrain the center point to displace normal to the plane
      begin fixed displacement
        node set =  NodeCentert NodeCenterb
        components = x z
      end fixed displacement
```

```
# Additional fixed displacement placed on the inital contact nodes to prevent planar movement
begin fixed displacement
  node set  = ContactNodet ContactNodeb
  components = x z
end fixed displacement

# Contact parameters ..............................................

begin contact definition
  search      = dash
  enforcement = al   #augmented lagrange
  $compute contact variables = on

  # We originally used time dependent friction model here, but that did not work in Adagio
  # thus the need for two procedures in this version of the tests.

  begin frictionless model hertz_friction_free
  end frictionless model hertz_friction_free

  {Ifdef(face_face)}
  skin all blocks = on
  begin interaction sphere2sphere
    surfaces = block_1  block_2
    friction model = hertz_friction_free
    # The following combination does not work and should flag a warning
    #   friction coefficient = 0.3
    #   friction coefficient function = friction_func
  end interaction sphere2sphere
  {Endif}

  {Ifdef(node_face)}
  contact node set top_node_set contains NodeContactt
  contact surface btm_surface contains SurfContactb
  begin interaction sphere2sphere
    master = btm_surface
    slave =  top_node_set
    friction model = hertz_friction_free
    # The following combination does not work and should flag a warning
    #   friction coefficient = 0.3
    #   friction coefficient function = friction_func
  end interaction sphere2sphere
  {Endif}

end contact definition

# Solver parameters ...............................................

begin solver
  begin loadstep predictor
    type = scale_factor
    scale factor = 0.0 0.0
  end
  begin control contact
    target relative residual = 1.0e-5
    acceptable relative residual = 5.0e-3
    target residual          = 1.0e-8
    maximum iterations       = 100
  end
  begin cg
    target relative residual = 1.0e-6
    target residual          = 1.0e-9
    maximum iterations       = 50
    begin full tangent preconditioner
      linear solver              = feti
      small number of iterations = 10
      conditioning               = no_check
    end
```

```
  end
end solver

# Output data ....................................................

{ifdef(extrapolate)}
begin user output
  include all blocks
  extrapolate element variable stress to nodal variable nodal_stress
end
{endif}

{ifdef(contact_forces)}
begin user output
  node set = nodecontactb
  compute nodal torque as function node_torque
  compute global Ty_top as sum of nodal torque
end
{endif}

begin user output
  node set = NodeFacet
  # use reaction force on top of top sphere
  compute global P as sum of nodal reaction(y)
  # First set of user outputs is for the Hertz part of the solution -- the preload
  compute global delta_analyt from expression "3^(2/3)*((-1+{nu}^2)^2*P^2/({Ey}^2*{R}))^(1/3)/(2^(1/3))"
  compute global delta_rel_error from expression "abs(delta_analyt-2*time)/delta_analyt"
  # analytical solution for the contact radius -- used mainly initially to judge the adequacy of the mes
  compute global a from expression "3^(1/3)*((-1+{nu}^2)*P*{R}/{Ey})^(1/3)/(2^(2/3))"
  compute global mesh from expression "{mesh}"
  # Second set of user outputs is for the Lubkin part of the solution -- the twist
  {ifdef(reaction_forces)}
  # Reaction torque on the top surface. First each nodal contribution is determined.
  compute nodal torque as function node_torque
  compute global Ty_top as sum of nodal torque
  {endif}
  # Torque Data obtained from the simulation. The error is based off the pade solution.
  # compute global Beta as function theta_twist
  # compute global theta_r from expression "abs(Beta*{G}*(a^2)/({mu}*P))"
  # compute global torque_sim from expression "abs(Ty_top/({mu}*P*a))"
  # compute global torque_analyt_pade from expression "({a0}+{a1}*(theta_r)+{a2}*(theta_r^2)+{a3}*(theta
  # compute global Torque_y_rel_error from expression "abs(torque_analyt_pade-torque_sim)/torque_analyt_
end

# Heartbeat output does not currently support user defined variables, or does it?
# This file is for quickly examining the results.
begin heartbeat output convergence_visual_output
  stream name = {shape}_{mesh}_compress_conv.txt
  global a
  # global theta_r
  # global torque_sim
  # global torque_analyt_pade
  # global Torque_y_rel_error
end heartbeat output convergence_visual_output

# Outputting Exodus Information
begin Results Output output_adagio
  Database Name = {shape}_{elem_topo}_{mesh}_compress.e
  Database Type = exodusII
  At Step 0, Increment = 1
  nodal variables = coordinates
  nodal variables = contact_status
  nodal variables = contact_incremental_slip_magnitude as contact_magnitude
  nodal variables = contact_area
  nodal variables = displacement as displ
  nodal variables = nodal_stress
  nodal variables = force_contact as fc
  nodal variables = reaction
```

```
        nodal variables = force_internal
        nodal variables = force_external
        element variables = stress as elem_stress
        global Variables = timestep
        global variables = mesh
        global variables = delta_analyt
        global variables = delta_rel_error
        global variables = P
        global variables = a
        # Variables used when spheres go into torsion
        {ifndef(rigid_body)}
        nodal variables = torque as torque_node
        global variables = Ty_top
        {endif}
        # global variables = torque_sim
        # global variables = torque_analyt_pade
        # global variables = Torque_y_rel_error
        {ifdef(rigid_body)}
        # global variables = rreacty_rb_1 as torque_reaction
        # global variables = rreacty_rb_2 as torque_reaction2
        {Endif}
        # Extra variables used to analyze energy dissipation effects
        # nodal variables = contact_frictional_energy as node_energy_dissipation
      end results output output_adagio

    end adagio region Compression_Region
  end adagio procedure Compression_Proc

# procedure data for final loading stage twist ----------------------------------------

  begin {proc_type} procedure Twist_Proc

    begin procedural transfer preload_to_twist
      include all blocks
    end procedural transfer preload_to_twist

    begin time control
      begin time stepping block twist_time
        start time = {preload_termination_time}
        begin parameters for {proc_type} region Twist_Region
          time increment = {(twist_termination_time-preload_termination_time)/twist_number_steps}
        end parameters for {proc_type} region Twist_Region
      end time stepping block twist_time
      termination time = {twist_termination_time}
    end time control

    # region data ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

    begin {proc_type} region Twist_Region
      use finite element model Half-spheres_mesh

      # BC data ..........................................................

      # Alternatively, we could just fix the values for this time period,
      # since they do not vary with time.

      # Prescribed vertical displacement on the face of the top hemisphere
      begin prescribed displacement
        node set = NodeFacet
        component = y
        function = delta_preload
        scale factor = -1.0
      end

      # Prescribed vertical displacement on the face of the bottom hemisphere
      begin prescribed displacement
        node set = NodeFaceb
        component = y
```

```
  function = delta_preload
  scale factor = 1.0
end

# Fix the x and z components of the center node of each hemisphere to
# constrain the center point to displace normal to the plane
begin fixed displacement
  node set =  NodeCentert NodeCenterb
  components = x z
end fixed displacement

# Prescribed rotation of the face on the top block
begin prescribed displacement
  active periods = twist_time
  node set = NodeFacet
  cylindrical axis = cylindrical_z_axis
  function = theta_twist
  scale factor = 1.0
end prescribed displacement

# Prescribed rotation of the face on the bottom block
begin prescribed displacement
  active periods = twist_time
  node set = NodeFaceb
  cylindrical axis = cylindrical_z_axis
  function = theta_twist
  scale factor = -1.0
end prescribed displacement

# Additional fixed displacement placed on the inital contact nodes to prevent planar movement
begin fixed displacement
  node set  = ContactNodet ContactNodeb
  components = x z
end fixed displacement

# Contact parameters ..............................................

begin contact definition
  search      = dash
  enforcement = al   #augmented lagrange
  compute contact variables = on

  begin constant friction model twist_friction
    friction coefficient = 0.3
  end constant friction model twist_friction

  {Ifdef(face_face)}
  skin all blocks = on
  begin interaction sphere2sphere
    surfaces = block_1  block_2
    friction model = twist_friction
    # The following combination does not work and should flag a warning
    #   friction coefficient = 0.3
    #   friction coefficient function = friction_func
  end interaction sphere2sphere
  {Endif}

  {Ifdef(node_face)}
  contact node set top_node_set contains NodeContactt
  contact surface btm_surface contains SurfContactb
  begin interaction sphere2sphere
    master = btm_surface
    slave =  top_node_set
    friction model = twist_friction
    # The following combination does not work and should flag a warning
    #   friction coefficient = 0.3
    #   friction coefficient function = friction_func
  end interaction sphere2sphere
```

```
    {Endif}

  end contact definition

# Solver parameters ..............................................

  begin solver
    begin loadstep predictor
      type = scale_factor
      scale factor = 0.0 0.0
    end
    begin control contact
      target relative residual = 1.0e-5
      acceptable relative residual = 5.0e-3
      target residual          = 1.0e-8
      maximum iterations       = 100
    end
    begin cg
      target relative residual = 1.0e-6
      target residual          = 1.0e-9
      maximum iterations       = 50
      begin full tangent preconditioner
        linear solver              = feti
        small number of iterations = 10
        conditioning               = no_check
      end
    end
  end solver

# Output data ....................................................

  begin user output
    include all blocks
    extrapolate element variable stress to nodal variable nodal_stress
  end

  {ifdef(contact_forces)}
  begin user output
    node set = nodecontactb
    compute nodal torque as function node_torque
    compute global Ty_top as sum of nodal torque
  end
  {endif}

  begin user output
    node set = NodeFacet
    # use reaction force on top of top sphere
    compute global P as sum of nodal reaction(y)
    # First set of user outputs is for the Hertz part of the solution -- the preload
    compute global delta_analyt from expression "3^(2/3)*((-1+{nu}^2)^2*P^2/({Ey}^2*{R}))^(1/3)/(2^(1/3))"
    compute global delta_rel_error from expression "abs(delta_analyt-2*time)/delta_analyt"
    # analytical solution for the contact radius -- used mainly initially to judge the adequacy of the mes
    compute global a from expression "3^(1/3)*((-1+{nu}^2)*P*{R}/{Ey})^(1/3)/(2^(2/3))"
    compute global mesh from expression "{mesh}"
    # Second set of user outputs is for the Lubkin part of the solution -- the twist
    {ifdef(reaction_forces)}
    # Reaction torque on the top surface. First each nodal contribution is determined.
    compute nodal torque as function node_torque
    compute global Ty_top as sum of nodal torque
    {endif}
    # Torque Data obtained from the simulation. The error is based off the pade solution.
    compute global Beta as function theta_twist
    compute global theta_r from expression "abs(Beta*{G}*(a^2)/({mu}*P))"
    compute global torque_sim from expression "abs(Ty_top/({mu}*P*a))"
    compute global torque_analyt_pade from expression "({a0}+{a1}*(theta_r)+{a2}*(theta_r^2)+{a3}*(theta_r
    compute global Torque_y_rel_error from expression "abs(torque_analyt_pade-torque_sim)/torque_analyt_pa
  end
```

```
# Heartbeat output does not currently support user defined variables, or does it?
# This file is for quickly examining the results.
begin heartbeat output convergence_visual_output
  stream name = {shape}_{mesh}_twist_conv.txt
  global a
  global theta_r
  global torque_sim
  global torque_analyt_pade
  global Torque_y_rel_error
end heartbeat output convergence_visual_output

# heartbeat file for plotting or convergence calculations
begin heartbeat output convergence_tabular_output
  stream name = {shape}_{mesh}_twist_conv_table.csv
  labels = off
  legend = off
  format = spyhis  # a csv file
  start time = {preload_termination_time}
  at step 1 increment = 1
  termination time = {twist_termination_time}
  # global time  -- unnecessary with spyhis format
  global mesh
  global Torque_y_rel_error
end heartbeat output convergence_tabular_output

# heartbeat file of load and calculated radius for asymptotic convergence analysis and plotting
begin heartbeat output load-radius_convergence_tabular_output
  stream name = {shape}_{mesh}_T_table.csv
  labels = off
  legend = off
  format = spyhis  # a csv file
  # start time = 0.0
  at step 1 increment = 1
  termination time = {twist_termination_time}
  # global time  -- unnecessary with spyhis format
  global mesh
  global Ty_top as Torque
end heartbeat output load-radius_convergence_tabular_output

# Outputting Exodus Information
begin Results Output output_adagio
  Database Name = {shape}_{elem_topo}_{mesh}_twist.e
  Database Type = exodusII
  At Step 0, Increment = 1
  nodal variables = coordinates
  nodal variables = contact_status
  nodal variables = contact_incremental_slip_magnitude as contact_magnitude
  nodal variables = contact_area
  nodal variables = displacement as displ
  nodal variables = nodal_stress
  nodal variables = force_contact as fc
  nodal variables = reaction
  nodal variables = force_internal
  nodal variables = force_external
  element variables = stress as elem_stress
  global Variables = timestep
  global variables = mesh
  global variables = delta_analyt
  global variables = delta_rel_error
  global variables = P
  global variables = a
  # Variables used when spheres go into torsion
  {ifndef(rigid_body)}
  nodal variables = torque as torque_node
  global variables = Ty_top
  {endif}
  global variables = torque_sim
  global variables = torque_analyt_pade
```

```
        global variables = Torque_y_rel_error
        {ifdef(rigid_body)}
        global variables = rreacty_rb_1 as torque_reaction
        global variables = rreacty_rb_2 as torque_reaction2
        {Endif}
        # Extra variables used to analyze energy dissipation effects
        # nodal variables = contact_frictional_energy as node_energy_dissipation
      end results output output_adagio

      # note that we will probably not use the following if we are able to use a convergence criterion
      begin solution verification
        completion file = torque_verification
        skip times = 0.0 to {twist_termination_time-epsilon_time}
        verify global Torque_y_rel_error = 0.0
        tolerance = {torque_tolerance}
      end

    end {proc_type} region Twist_Region
  end {proc_type} procedure Twist_Proc

  Begin feti equation solver feti
    $ Default is 1e-3, changed for efficiency
    residual norm tolerance = 1e-2
  End

end sierra Lubkin
```

B.8. STICKING-SLIPPING BLOCK AND SPRING - EXPLICIT DYNAMICS

See Section 2.8 for problem description.

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
begin sierra sticking_block_spring

  begin function one
    type is constant
    begin values
      1.0
    end
  end function

  begin function vert_force
    type is analytic
    evaluate expression = "(-1.0*cos(x*3.14159/10.0)+1.0)/2.0"
  end function

  begin function horiz_force_2
    type is analytic
    evaluate expression = "sin((x-10.0)*3.14159/20.0)"
  end

  begin function spring_react_check
    type is analytic
    evaluate expression = "-10.0*sin((x-10.0)*3.14159/20.0)+(0.5*10.0)"
  end

  begin function spring_stiffness
    type is piecewise linear
    begin values
      0.0  0.0
      0.001  1.0
    end
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin material linear_elastic
    density = 1.0e3
    begin parameters for model elastic
      youngs modulus  = 1.0e8
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic

  begin material linear_elastic_soft
    density = 1.0e3
    begin parameters for model elastic
      youngs modulus  = 1.0e7
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic_soft


  begin solid section blocks
  end

  begin truss section spring
    area = 0.01
  end
```

```
begin finite element model mesh1
  Database Name = blocks.g
  Database Type = exodusII

  begin parameters for block block_1 block_2
    material = linear_elastic
    model = elastic
    section = blocks
  end

  begin parameters for block block_3
    material = linear_elastic_soft
    model = elastic
    section = spring
  end

end finite element model mesh1


begin presto procedure Presto_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region presto_region
      end
    end
    begin time stepping block p2
      start time = 10.0
      begin parameters for presto region presto_region
      end
    end
    termination time = 15.0
  end

  begin presto region presto_region

    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = blocks.e
      Database Type = exodusII
      At time 0, Increment = 0.001
      nodal Variables = displacement as displ
      nodal Variables = velocity as vel
      nodal variables = reaction
      nodal variables = force_external
      nodal variables = force_contact
      nodal variables = contact_tangential_force_magnitude as ctfm
      nodal variables = contact_normal_force_magnitude as cnfm
      nodal variables = contact_incremental_slip_direction as cisd
      nodal variables = contact_accumulated_slip_vector as casv
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
      global variables = spring_react
      global variables = spring_react_check
    end

    begin history output
      Database Name = blocks.h
      Database Type = exodusII
      At step 0, increment = 1
      global spring_react
      global spring_react_check
```

```
end

begin user output
  node = 172
  compute global spring_react as max of nodal reaction(y)
  compute global spring_react_check as function spring_react_check
  compute at every step
end
### definition of BCs ###

begin fixed displacement
  node set = nodelist_1
  components = x y z
end

begin fixed displacement
  surface = surface_1
  components = x y z
end

begin fixed displacement
  block = block_2
  components = x
end

begin gravity
  active periods = p1
  block = block_2
  gravitational constant = 1.0
  function = vert_force
  scale factor = -0.01
  direction = z
end

begin gravity
  active periods = p2
  block = block_2
  gravitational constant = 1.0
  function = one
  scale factor = -0.01
  direction = z
end

begin traction
  active periods = p2
  surface = surface_3
  function = horiz_force_2
  scale factor = 10.0
  direction = y
end



begin contact definition friction
  search = dash
  contact surface surf_1 contains block_1
  contact surface surf_2 contains block_2
  begin interaction inter_1
    surfaces = surf_1 surf_2
    friction model = fric
  end interaction inter_1
  begin constant friction model fric
    friction coefficient = 0.5
  end
end contact definition friction

begin viscous damping
  include all blocks
```

```
        mass damping coefficient = 10.0
      end

      begin solution verification
        completion file = verif_react_zero
        skip times = 13.0 to 20.0
        verify global spring_react = 0.0
        tolerance = 0.25 # 5% of max tangential friction force
      end

      begin solution verification
        completion file = verif_react
        skip times = 0 to 13.6
        verify global spring_react = function spring_react_check
        tolerance = 0.25 # 5% of max tangential friction force

      end

    end presto region presto_region
  end presto procedure Presto_Procedure
end sierra sticking_block_spring
```

B.9. STICKING-SLIPPING BLOCK AND SPRING - IMPLICIT DYNAMICS

See Section 2.9 for problem description.

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
begin sierra sticking_block_spring

  begin function one
    type is constant
    begin values
      1.0
    end
  end function

  begin function vert_force
    type is analytic
    evaluate expression = "(-1.0*cos(x*3.14159/10.0)+1.0)/2.0"
  end function

  begin function horiz_force_2
    type is analytic
    evaluate expression = "sin((x-10.0)*3.14159/20.0)"
  end

  begin function spring_react_check
    type is analytic
    evaluate expression = "-10.0*sin((x-10.0)*3.14159/20.0)+(0.5*10.0)"
  end

  begin function spring_stiffness
    type is piecewise linear
    begin values
      0.0  0.0
      0.001  1.0
    end
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin material linear_elastic
    density = 1.0e3
    begin parameters for model elastic
      youngs modulus  = 1.0e8
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic

  begin material linear_elastic_soft
    density = 1.0e3
    begin parameters for model elastic
      youngs modulus  = 1.0e7
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic_soft


  begin solid section blocks
  end

  begin truss section spring
    area = 0.01
  end
```

```
begin finite element model mesh1
  Database Name = blocks.g
  Database Type = exodusII

  begin parameters for block block_1 block_2
    material = linear_elastic
    model = elastic
    section = blocks
  end

  begin parameters for block block_3
    material = linear_elastic_soft
    model = elastic
    section = spring
  end

end finite element model mesh1


begin adagio procedure Adagio_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for adagio region adagio_region
        number of time steps = 5
      end
    end
    begin time stepping block p2
      start time = 10.0
      begin parameters for adagio region adagio_region
        number of time steps = 30
      end
    end
    termination time = 15.0
  end

  begin adagio region adagio_region

    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = blocks.e
      Database Type = exodusII
      At time 0, Increment = 0.001
      nodal Variables = displacement as displ
      nodal Variables = velocity as vel
      nodal variables = reaction
      nodal variables = force_external
      nodal variables = force_contact
      nodal variables = contact_tangential_force_magnitude as ctfm
      nodal variables = contact_normal_force_magnitude as cnfm
      nodal variables = contact_incremental_slip_direction as cisd
      nodal variables = contact_accumulated_slip_vector as casv
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
      global variables = spring_react
      global variables = spring_react_check
    end

    begin history output
      Database Name = blocks.h
      Database Type = exodusII
      At step 0, increment = 1
```

```
  global spring_react
  global spring_react_check
end

begin user output
  node = 172
  compute global spring_react as max of nodal reaction(y)
  compute global spring_react_check as function spring_react_check
  compute at every step
end
### definition of BCs ###

begin fixed displacement
  node set = nodelist_1
  components = x y z
end

begin fixed displacement
  surface = surface_1
  components = x y z
end

begin fixed displacement
  block = block_2
  components = x
end

begin gravity
  active periods = p1
  block = block_2
  gravitational constant = 1.0
  function = vert_force
  scale factor = -0.01
  direction = z
end

begin gravity
  active periods = p2
  block = block_2
  gravitational constant = 1.0
  function = one
  scale factor = -0.01
  direction = z
end

begin traction
  active periods = p2
  surface = surface_3
  function = horiz_force_2
  scale factor = 10.0
  direction = y
end



begin implicit dynamics
end

begin contact definition friction
  search      = dash
  enforcement = al
  contact surface surf_1 contains block_1
  contact surface surf_2 contains block_2
  begin interaction inter_1
    surfaces = surf_1 surf_2
    friction model = fric
    al penalty = 2.0
  end interaction inter_1
```

```
        begin constant friction model fric
          friction coefficient = 0.5
        end
      end contact definition friction

      Begin solver
        begin loadstep predictor
          type = scale_factor
          scale factor = 0.0 0.0
        end
        begin control contact
          target relative residual       = 1.0e-5
          acceptable relative residual   = 1.0e-3
          Maximum Iterations             = 1000
          Minimum Iterations             = 3
          lagrange initialize            = none
          lagrange adaptive penalty      = off
        end
        Begin cg
          target relative residual     = 1.0e-6
          maximum iterations           = 125
          iteration print              = 1
          begin full tangent preconditioner
            conditioning = no_check
            tangent diagonal scale = 1.0e-3
            small number of iterations = 25
          end
        end
      end

      begin solution verification
        completion file = verif_react_zero
        skip times = 13.0 to 20.0
        verify global spring_react = 0.0
        tolerance = 0.1 # 2% of max tangential friction force
      end

      begin solution verification
        completion file = verif_react
        skip times = 0 to 13.6
        verify global spring_react = function spring_react_check
        tolerance = 0.1 # 2% of max tangential friction force
      end

    end adagio region adagio_region
  end adagio procedure Adagio_Procedure
end sierra sticking_block_spring
```

B.10. STICKING-SLIPPING BLOCK AND SPRING - IMPLICIT STATICS

See Section 2.10 for problem description.

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
begin sierra sticking_block_spring

  begin function one
    type is constant
    begin values
      1.0
    end
  end function

  begin function vert_force
    type is analytic
    evaluate expression = "(-1.0*cos(x*3.14159/10.0)+1.0)/2.0"
  end function

  begin function horiz_force_2
    type is analytic
    evaluate expression = "sin((x-10.0)*3.14159/20.0)"
  end

  begin function spring_react_check
    type is analytic
    evaluate expression = "-10.0*sin((x-10.0)*3.14159/20.0)+(0.5*10.0)"
  end

  begin function spring_stiffness
    type is piecewise linear
    begin values
      0.0  0.0
      0.001  1.0
    end
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin material linear_elastic
    density = 1.0e3
    begin parameters for model elastic
      youngs modulus  = 1.0e8
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic

  begin material linear_elastic_soft
    density = 1.0e3
    begin parameters for model elastic
      youngs modulus  = 1.0e7
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic_soft


  begin solid section blocks
  end

  begin truss section spring
    area = 0.01
  end
```

```
begin finite element model mesh1
  Database Name = blocks.g
  Database Type = exodusII

  begin parameters for block block_1 block_2
    material = linear_elastic
    model = elastic
    section = blocks
  end

  begin parameters for block block_3
    material = linear_elastic_soft
    model = elastic
    section = spring
  end

end finite element model mesh1


begin adagio procedure Adagio_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for adagio region adagio_region
        number of time steps = 5
      end
    end
    begin time stepping block p2
      start time = 10.0
      begin parameters for adagio region adagio_region
        number of time steps = 25
      end
    end
    termination time = 15.0
  end

  begin adagio region adagio_region

    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = blocks.e
      Database Type = exodusII
      At time 0, Increment = 0.001
      nodal Variables = displacement as displ
      nodal Variables = velocity as vel
      nodal variables = reaction
      nodal variables = force_external
      nodal variables = force_contact
      nodal variables = contact_tangential_force_magnitude as ctfm
      nodal variables = contact_normal_force_magnitude as cnfm
      nodal variables = contact_incremental_slip_direction as cisd
      nodal variables = contact_accumulated_slip_vector as casv
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
      global variables = spring_react
      global variables = spring_react_check
    end

    begin history output
      Database Name = blocks.h
      Database Type = exodusII
      At step 0, increment = 1
```

```
  global spring_react
  global spring_react_check
end

begin user output
  node = 172
  compute global spring_react as max of nodal reaction(y)
  compute global spring_react_check as function spring_react_check
  compute at every step
end
### definition of BCs ###

begin fixed displacement
  node set = nodelist_1
  components = x y z
end

begin fixed displacement
  surface = surface_1
  components = x y z
end

begin fixed displacement
  block = block_2
  components = x
end

begin gravity
  active periods = p1
  block = block_2
  gravitational constant = 1.0
  function = vert_force
  scale factor = -0.01
  direction = z
end

begin gravity
  active periods = p2
  block = block_2
  gravitational constant = 1.0
  function = one
  scale factor = -0.01
  direction = z
end

begin traction
  active periods = p2
  surface = surface_3
  function = horiz_force_2
  scale factor = 10.0
  direction = y
end



begin contact definition friction
  search      = dash
  enforcement = al
  contact surface surf_1 contains block_1
  contact surface surf_2 contains block_2
  begin interaction inter_1
    surfaces = surf_1 surf_2
    friction model = fric
  end interaction inter_1
  begin constant friction model fric
    friction coefficient = 0.5
  end
end contact definition friction
```

```
      Begin solver
        begin loadstep predictor
          type = scale_factor
          scale factor = 0.0 0.0
        end
        begin control contact
          target relative residual       = 1.0e-5
          acceptable relative residual   = 1.0e-3
          Maximum Iterations             = 1000
        end
        Begin cg
          acceptable relative residual = 1.0e10
          target relative residual     = 1.0e-6
          maximum iterations           = 100
          iteration print              = 10
          begin full tangent preconditioner
            tangent diagonal scale = 1.0e-4
          end
        end
      end

      begin solution verification
        completion file = verif_react_zero
        skip times = 13.0 to 20.0
        verify global spring_react = 0.0
        tolerance = 0.05 # 1% of max tangential friction force
      end

      begin solution verification
        completion file = verif_react
        skip times = 0 to 13.6
        verify global spring_react = function spring_react_check
        tolerance = 0.05 # 1% of max tangential friction force
      end

    end adagio region adagio_region
  end adagio procedure Adagio_Procedure
end sierra sticking_block_spring
```

B.11. COULOMB FRICTION WITH SLIDING [EXPLICIT DYNAMICS, FACE/FACE CONTACT]

See Section 2.11 for problem description.

```
begin sierra coulombSlide

  {include("MaterialsAndFunction.inp")}

  begin adagio procedure Adagio_Procedure

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for adagio region region_1
          number of time steps = 250
        end parameters for adagio region region_1
      end time stepping block time_control_1
      termination time = 0.05
    end time control

    begin adagio region region_1
      {include("BCAndOutput.inp")}


      begin user output
        node = 211
        compute global contactStiff as max of nodal scalar_stiffness
        compute global dynamicStiff from expression "0.0"
        compute global staticStiff from expression "contactStiff"
        compute at every step
      end

#
#  Solution verification
#
#    Verify that the solution was computed accuratly.
Verification is by total error integral, the integral of the distance between
#
the analytic curve and the analysis curve divided by the total area under the analytic curve.
Note, due to various reasons
#
the solution is somewhat off, thus ensure that the solution maintains the current known properties.
#
      begin solution verification
        skip times = 0.0 to 0.049
        completion file = VerifErr
        verify global relErrFX = 0.0085
        verify global relErrFY = 0.0146
        verify global relErrFZ = 0.0
        verify global relErrDz = 0.002
        tolerance = 0.01
      end
#
#  Iter count too variable on different platforms, test unstable.....,
#  removing for now
#
      begin solution verification
        skip times = 0.0 to 0.049
#        verify global total_iter = 86269
#        tolerance = 1000
        completion file = VerifIter
      end

    end adagio region region_1
```

```
  end adagio procedure Adagio_Procedure

end sierra coulombSlide
```

B.12. OSCILLATING BLOCK SPRING WITH FRICTION

See Section 2.12 for problem description.

```
begin sierra oscillating_block_spring_with_friction

# Aprepro macros for problem parameters
# mass in kg
# {m = 4}
# Natural frequency prescribed as 2Pi radians/sec (i.e., 1 hz)
# {omegan = 2*PI}
# spring stiffness in N/m to yield prescribed omegan: m*omegan^2
# {k = m*omegan^2}
# gravitational constant, 9.81 m/sec^2
# {g = 9.81}
# Uinitial ~ offeset of the block, 1 m to the left
# {Uinitial = -1.0}
# Coefficient of friction -- defined to produce stick after two cycles of oscillation
# mu = -k*Uinitial/(9*g*m) = 400 Pi^2/8829 ~ 4.47144836216268e-1
# {mu = -k*Uinitial/(9*g*m)}
# Ud ~ a measure of the decrease in peak displacement that occurs linearly in time: mu*g/omegan^2
# {Ud = mu*g/omegan^2}

  {include("func_matl_femodel.i")}

  begin presto procedure Presto_Procedure

    begin time control
      begin time stepping block preload
        start time = 0.0
        begin parameters for presto region presto_region
          time step scale factor = 0.95
        end
      end
      begin time stepping block free_vibration
        start time = 10.0
        begin parameters for presto region presto_region
          time step scale factor = 0.95
        end
      end
      termination time = 20
    end

    begin presto region presto_region

      {include("region_shared.i")}

      begin contact definition friction
        search = dash

        contact surface surf_1 contains block_1
        contact surface surf_2 contains block_2

        begin interaction inter_1
          surfaces = surf_1 surf_2
          friction model = mod
        end interaction inter_1

        begin time variant model mod
          model = frictionless during periods preload
          model = fric during periods free_vibration
        end


        begin constant friction model fric
          # coefficient defined such that stick occurs after 2 cycles of vibration
          friction coefficient = {mu}
```

```
        end

        begin enforcement options
          momentum balance iterations = 5
        end

        compute contact variables = on
      end contact definition friction

      begin solution verification
         completion file = v1
         skip times = 0.0 to 19.9
         verify global relativeError = 0.0123779 plus or minus 0.002
         verify global fric_energy_sum = {fricEnergyEnd} plus or minus {abs(fricEnergyEnd)/100}
      end

    end presto region presto_region
  end presto procedure Presto_Procedure
end sierra oscillating_block_spring_with_friction
```

B.13. FRICTION WEDGE

See Section 2.13 for problem description.

```
##############################################################################

BEGIN SIERRA friction_wedge

  TITLE frictional wedge with rigid body contact

#  Pedigree: This test is related to two other verification/performance tests.
#    (1) rigid_body_wedge (path: adagio_rtest/presto/rigid_body_wedge).
#        This test was used as the initial template for friction_wedge. It shares
#        the same mesh and much of the same input, but the emphasis of friction_wedge
#        is upon examining the stick-slip behavior. This one uses rigid body edge
#        wedges.
#    (2) p015_wedge_friction (path: adagio_rtest/performance/p015_wedge_friction)
#        This test shares the same geometry as this test, but the mesh is much finer.
#        Both had been used to study the frictional contact algorithms in the past.
#        This one uses elastic edge wedges.

##############################################################################

# Aprepro macros
# Offset from the critical coefficient of friction of 0.2.
# Original offset was 0.001 but up to 0.1 was also examined.
# {fric_offset = 0.001}
# time to transition to a constant velocity. Used as the interval for deacceleration too.
# {time_Vconst = 0.001}
# time to start transition to zero velocity, i.e., where constant velocity ends
# {time_Deaccel = 0.002}
# time to end analysis, end of constant position time interval, std value is 0.004
# {time_end = 0.05}
# maximum velocity magnitude. standard value = 25 in/sec
# {velocity_max = 5}
# Number of momentum balance iterations
#   momentum_bal_iter now passed in via the command line

##### INITIALIZE DIRECTIONS
  DEFINE DIRECTION X WITH VECTOR 1.0 0.0 0.0
  DEFINE DIRECTION Y WITH VECTOR 0.0 1.0 0.0
  DEFINE DIRECTION Z WITH VECTOR 0.0 0.0 1.0

  # The following function is used to displace the top wedge downward and the bottom block upward
  # with a velocity boundary condition. It consists of a cosine transtion from zero
  # to unity, an interval of unity, a cosine transition back down to zero
  # velocity (fixed position), and then a fixed position for the remainder of time.
  begin function cosine2unity2zero
    type is piecewise analytic
    begin expressions
       0.0                        "(1.0-cos(t*PI/{time_Vconst}))/2.0"
       {time_Vconst}              "1.0"
       {time_Deaccel}              "(cos((t-{time_Deaccel})*PI/{time_Vconst})+1.0)/2.0"
       {time_Deaccel+time_Vconst} "0.0"
    end
  end function

##############################################################################

##### DEFINE MATERIALS

  BEGIN PROPERTY SPECIFICATION FOR MATERIAL edge_wedge_mat
    DENSITY = 7.4e-5
    BEGIN PARAMETERS FOR MODEL ELASTIC
      YOUNGS MODULUS = 1.0e4
      POISSONS RATIO = 0.00
    END PARAMETERS FOR MODEL ELASTIC
```

```
  END PROPERTY SPECIFICATION FOR MATERIAL edge_wedge_mat

  BEGIN PROPERTY SPECIFICATION FOR MATERIAL center_wedge_mat
    DENSITY = 7.4e-5
    BEGIN PARAMETERS FOR MODEL ELASTIC
      YOUNGS MODULUS = 1.0E4
      POISSONS RATIO = 0.00
    END PARAMETERS FOR MODEL ELASTIC
  END PROPERTY SPECIFICATION FOR MATERIAL center_wedge_mat

#  begin rigid body 1
#  end rigid body 1

#  begin solid section rigid_1
#    rigid body = 1
#  end

  # begin solid section solid_2
  #   formulation = fully_integrated
  # end solid section solid_2

#  begin rigid body 3
#  end rigid body 3

#  begin solid section rigid_3
#    rigid body = 3
#  end

###############################################################################

##### DEFINE FEM MODEL
  BEGIN FINITE ELEMENT MODEL BLOCKS

    ##### FILE NAMES & TYPES
    {if(surface_blocks=="on")}
    DATABASE NAME = friction_wedge.g
    {Endif}
    {if(surface_sidesets=="on")}
    # Load the one with side sets for this permutation of the problem.
    DATABASE NAME = friction_wedge_ss.g
    {endif}
    DATABASE TYPE = EXODUSII
    # Blocks
    #   1 ~ bottom wedge
    #   2 ~ middle wedge
    #   3 ~ top    wedge
    # Node sets
    #  10 ~ all x-surface nodes (+/-) of top & bottom wedges
    #  21 ~ bottom surface nodes of bottom wedge
    #  22 ~ top surface nodes of top wedge
    # 101 ~ all nodes for 3 wedges on +z surfaces
    # 102 ~ all nodes for 3 wedges on -z surfaces
    # Side sets
    #  20 ~ top surface of top wedge and bottom surface of bottom wedge

##### DEFINE BLOCK 1
    BEGIN PARAMETERS FOR BLOCK block_1
      MATERIAL edge_wedge_mat
      SOLID MECHANICS USE MODEL ELASTIC
#      section = rigid_1
    END PARAMETERS FOR BLOCK block_1

##### DEFINE BLOCK 2
    BEGIN PARAMETERS FOR BLOCK block_2
      HOURGLASS VISCOSITY = 1.0E-3
      MATERIAL center_wedge_mat
      SOLID MECHANICS USE MODEL ELASTIC
      # section = solid_2
```

```
    END PARAMETERS FOR BLOCK block_2

##### DEFINE BLOCK 3
    BEGIN PARAMETERS FOR BLOCK block_3
      MATERIAL edge_wedge_mat
      SOLID MECHANICS USE MODEL ELASTIC
#      section = rigid_3
    END PARAMETERS FOR BLOCK block_3

  END FINITE ELEMENT MODEL BLOCKS

################################################################################

  BEGIN PRESTO PROCEDURE PRESTO_CALCULATIONS

    ##### DEFINE PROBLEM TIME AND TIME STEP PARAMETERS
    BEGIN TIME CONTROL
      TERMINATION TIME = {time_end}
      BEGIN TIME STEPPING BLOCK TIMESTEPING
        START TIME = 0.0
        BEGIN PARAMETERS FOR PRESTO REGION PROBLEM
          STEP INTERVAL = 100
          time step scale factor = 1.0
        END PARAMETERS FOR PRESTO REGION PROBLEM
      END TIME STEPPING BLOCK TIMESTEPING
    END TIME CONTROL

##### DEFINE PROBLEM
    BEGIN PRESTO REGION PROBLEM
      USE FINITE ELEMENT MODEL BLOCKS

##### DEFINE OUTPUT
      BEGIN RESULTS OUTPUT OUTPUT_PRESTO
        DATABASE NAME = friction_wedge_iter{momentum_bal_iter}.e
        DATABASE TYPE = EXODUSII
        AT TIME 0, INCREMENT = 1.0E-3
        NODAL VARIABLES = ACCELERATION AS ACCL
        NODAL VARIABLES = VELOCITY AS VEL
        NODAL VARIABLES = DISPLACEMENT AS DISPL
        nodal variables = force_contact
        nodal variables = contact_accumulated_slip as slip
        ELEMENT VARIABLES = VON_MISES AS VONMISES
        global variables = external_energy as ExternalEnergy
        global variables = internal_energy as InternalEnergy
        global variables = kinetic_energy as KineticEnergy
        global variables = momentum as Momentum
        global variables = timestep as TIMESTEP
      END RESULTS OUTPUT OUTPUT_PRESTO

      begin user output
        block = block_2
        compute global Uxave as average of nodal displacement(x)
        compute global UxBlk2ave from expression "abs(Uxave)"
        compute global SlipAveAllNodes as average of nodal contact_accumulated_slip
        compute global SlipBlk2Ave from expression "2.0*SlipAveAllNodes"
        compute at every step
      end

      begin user output
        surface = mid_blk_top
        compute global totConForceYa as sum of nodal force_contact(y)
      end

      begin user output
        surface = mid_blk_btm
        compute global totConForceYb as sum of nodal force_contact(y)
      end
```

```
      #
      #  Get slip rate metrics for use in verification
      #
      begin user variable t0
        type = global real
        global operator = max
      end
      begin user variable t1
        type = global real
        global operator = max
      end
      begin user variable s0
        type = global real
        global operator = max
      end
      begin user variable s1
        type = global real
        global operator = max
      end
      begin user variable f0
        type = global real
        global operator = max
      end
      begin user variable f1
        type = global real
        global operator = max
      end

      begin user output
        compute global totConForceY from expression "abs(totConForceYb - totConForceYa)"

        compute global t0 from expression "(time < 0.025) ? time         : t0 "
        compute global s0 from expression "(time < 0.025) ? SlipBlk2Ave  : s0 "
        compute global f0 from expression "(time < 0.025) ? totConForceY : f0 "
        compute global t1 from expression "(time < 0.050) ? time         : t1 "
        compute global s1 from expression "(time < 0.050) ? SlipBlk2Ave  : s1 "
        compute global f1 from expression "(time < 0.050) ? totConForceY : f1 "

        compute global finErr from expression "((s1-s0)/(t1-t0))/(0.5*(f1+f0))"


      end

# {expectedError}
{Ifdef(expectedError)}
      begin solution verification
        skip times = 0.0 to 0.0499
        completion file = v{momentum_bal_iter}
        verify global finErr = {expectedError} plus or minus {expectedError * 0.05}
      end
{Endif}


      begin history output
        Database Name = friction_wedge_iter{momentum_bal_iter}.h
        Database Type = exodusII
        At step 0, increment = 1
        # node 55 ~ bottom left node on z-face of middle wedge
        nodal contact_accumulated_slip at node 55 as slip_55
        # node 91 ~ bottom left node on z-face of middle wedge
        nodal contact_accumulated_slip at node 91 as slip_91
        # node    ~ bottom left node on z-face of bottom wedge
        #  used to monitor the boundary condition
        node displacement(y) at node 1 as disp_y_1
        node velocity(y) at node 1 as velo_y_1
        # nodal displacement at node 55
        node displacement(x) at node 55 as disp_x_55
```

```
        # user variables
        global UxBlk2ave
        global SlipBlk2Ave
        global totConForceY

        global t0
        global t1
        global s0
        global s1
        global f0
        global f1
        global finErr
      end history output




##### DEFINE BOUNDARY CONDITIONS

      begin fixed displacement
        block = block_1
        components = x z
      end fixed displacement

      begin fixed rotation
        block = block_1
        components = x y z
      end fixed rotation

      BEGIN PRESCRIBED VELOCITY
        block = block_1
        DIRECTION = Y
        FUNCTION = cosine2unity2zero
        SCALE FACTOR = {velocity_max}
      END PRESCRIBED VELOCITY

      begin fixed displacement
        block = block_2
        components = z
      end fixed displacement

      begin fixed displacement
        block = block_3
        components = x z
      end fixed displacement

      begin fixed rotation
        block = block_3
        components = x y z
      end fixed rotation

      BEGIN PRESCRIBED VELOCITY
        block = block_3
        DIRECTION = Y
        FUNCTION = cosine2unity2zero
        SCALE FACTOR = {-velocity_max}
      END PRESCRIBED VELOCITY

##### DEFINE CONTACT
      BEGIN CONTACT DEFINITION frictional

        search = {search_method}

        {if(surface_blocks=="on")}
        CONTACT SURFACE SS1  CONTAINS block_1
        CONTACT SURFACE SS2  CONTAINS block_2
        CONTACT SURFACE SS3  CONTAINS block_3
```

```
{endif}
{if(surface_sidesets=="on")}
CONTACT SURFACE SS1  CONTAINS sideset_2
CONTACT SURFACE SS2a CONTAINS sideset_3
CONTACT SURFACE SS2b CONTAINS sideset_6
CONTACT SURFACE SS3  CONTAINS sideset_9
{endif}
{if(surface_nodesets=="on")}
CONTACT SURFACE SS1  CONTAINS block_1
CONTACT NODE SET NS2  CONTAINS block_2
CONTACT SURFACE SS3  CONTAINS block_3
{endif}


compute contact variables = on

begin enforcement options
  momentum balance iterations = {momentum_bal_iter}
  num geometry update iterations = 1
end

BEGIN CONSTANT FRICTION MODEL F1
  FRICTION COEFFICIENT = {0.20+fric_offset}
END CONSTANT FRICTION MODEL F1

{if(surface_blocks=="on")}
BEGIN INTERACTION S1
  SURFACES = SS1 SS2
  KINEMATIC PARTITION = 0.0
  NORMAL TOLERANCE = 0.001
  TANGENTIAL TOLERANCE = 0.001
  FRICTION MODEL = F1
END INTERACTION S1

BEGIN INTERACTION S2
  SURFACES = SS3 SS2
  KINEMATIC PARTITION = 0.0
  NORMAL TOLERANCE = 0.001
  TANGENTIAL TOLERANCE = 0.001
  FRICTION MODEL = F1
END INTERACTION S2
{endif}
{if(surface_sidesets=="on")}
BEGIN INTERACTION S1
  SURFACES = SS1 SS2a
  KINEMATIC PARTITION = 0.0
  NORMAL TOLERANCE = 0.001
  TANGENTIAL TOLERANCE = 0.001
  FRICTION MODEL = F1
END INTERACTION S1

BEGIN INTERACTION S2
  SURFACES = SS3 SS2b
  KINEMATIC PARTITION = 0.0
  NORMAL TOLERANCE = 0.001
  TANGENTIAL TOLERANCE = 0.001
  FRICTION MODEL = F1
END INTERACTION S2
{endif}

{if(surface_nodesets=="on")}
BEGIN INTERACTION S1
  master = SS1
  slave = NS2
  NORMAL TOLERANCE = 0.001
  TANGENTIAL TOLERANCE = 0.001
  FRICTION MODEL = F1
END INTERACTION S1
```

```
        BEGIN INTERACTION S2
          master = SS3
          slave = NS2
          NORMAL TOLERANCE = 0.001
          TANGENTIAL TOLERANCE = 0.001
          FRICTION MODEL = F1
        END INTERACTION S2
        {endif}


      END CONTACT DEFINITION frictional

    END PRESTO REGION PROBLEM
  END PRESTO PROCEDURE PRESTO_CALCULATIONS
END SIERRA friction_wedge

##############################################################################
```

B.14. HEX PATCH TESTS – QUASI-STATIC, LINEAR ELASTIC

See Section 3.1 for problem description.

```
begin sierra Hex_patch

  # Units
  # length: inches
  # force: lbs
  # stress: psi
  # time: seconds (pseudo-time for these quasistatic runs)

  # Aprepro macros ..............
  # {u_factor = 1.0E-4}
  # {final_time = 1.0}
  # {number_steps = 2}
  # Expected normal and shear stresses expected from the patch test -- a linear elastic approximation
  # Note that these could be obtained more symbolically to generalize the test.
  # {sigNormal = 400.0}
  # {sigShear =   80.0}
  # epsilon_time is an off set from the final_time used to ignore all results
  # except those associated with the final time step.
  # { epsilon_time = final_time/(number_steps*2)}

  # Required externally provided macros with string values:
  #   normal_stress_tolerance
  #   shear_stress_tolerance
  #   strain_incrementation
  #   formulation
  #   meshFile
  #   material model
  # Required externally provided macros "on" or "off":
  #   mean_quad
  #   selective_dev

  # Ux displacement component as a function of space and time
  begin definition for function u
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(2.0*x+y+z)"
  end definition for function u

  # Uy displacement component as a function of space and time
  begin definition for function v
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(x+2.0*y+z)"
  end definition for function v

  # Uz displacement component as a function of space and time
  begin definition for function w
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(x+y+2.0*z)"
  end definition for function w
```

```
define direction x with vector 1.0 0.0 0.0
define direction y with vector 0.0 1.0 0.0
define direction z with vector 0.0 0.0 1.0

begin material block_mat
  density         = 2.61e-4
  begin parameters for model {material_model}
    youngs modulus  = 1.0e6
    poissons ratio  = 0.25
  end parameters for model {material_model}
end material block_mat

begin solid section solid_1
  strain incrementation = {strain_incrementation}
  formulation           = {formulation}
  {Ifdef(selective_dev)}
  deviatoric parameter = 0.5
  {Else}
  # selective_dev undefined
  {Endif}
end

begin finite element model mesh1
  Database Name = {meshFile}
  Database Type = exodusII

  begin parameters for block block_1
    material = block_mat
    model = {material_model}
    section = solid_1
    {Ifdef(mean_quad)}
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_1

  begin parameters for block block_2
    material = block_mat
    model = {material_model}
    section = solid_1
    {Ifdef(mean_quad)}
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_2

  begin parameters for block block_3
    material = block_mat
    model = {material_model}
    section = solid_1
    {Ifdef(mean_quad)}
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_3

  begin parameters for block block_4
    material = block_mat
    model = {material_model}
    section = solid_1
    {Ifdef(mean_quad)}
    linear bulk viscosity = 0.0
```

```
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_4

  begin parameters for block block_5
    material = block_mat
    model = {material_model}
    section = solid_1
    {Ifdef(mean_quad)}
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_5

  begin parameters for block block_6
    material = block_mat
    model = {material_model}
    section = solid_1
    {Ifdef(mean_quad)}
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_6

  begin parameters for block block_7
    material = block_mat
    model = {material_model}
    section = solid_1
    {Ifdef(mean_quad)}
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_7

end finite element model mesh1

begin adagio procedure adagio_patch_procedure

  begin time control
    begin time stepping block ramp_segment
      start time = 0.0
      begin parameters for adagio region adagio_patch_region
        time increment = {final_time/number_steps}
      end parameters for adagio region adagio_patch_region
    end time stepping block ramp_segment
    termination time = {final_time}
  end time control

  begin adagio region adagio_patch_region
    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = hex_patch.e
      Database Type = exodusII
      At Time 0.0, Increment = {final_time}
      nodal Variables = force_external as f_ext
      nodal Variables = force_internal as f_int
      nodal Variables = displacement
      element Variables = stress as stress_el
```

```
  element Variables = cauchy_stress
  element Variables = log_strain as strain_el
  element variables = timestep as elem_time_step
  global Variables = timestep as timestep
  global variables = external_energy as ExternalEnergy
  global variables = internal_energy as InternalEnergy
end results output output_adagio

### definition of BCs ###

# For Hex8 mesh the union of the following nodesets gives all the boundary nodes:
#    nodelist_1 ... nodelist_8
# For the Hex27 mesh the union of the following nodesets gives all the boundary nodes::
#   nodelist_1 ... nodelist_26
# To simplify the BC spectification for now use the 6 surfaces of the block.

# X displacements

begin prescribed displacement
  # node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6 nodelist_7 nodelist_8
  surface = surface_100 surface_200 surface_300 surface_400 surface_500 surface_600
  # block = block_1 -- for debugging
  component = X
  function = u
  scale factor = 1.0
end

# Y displacements

begin prescribed displacement
  # node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6 nodelist_7 nodelist_8
  surface = surface_100 surface_200 surface_300 surface_400 surface_500 surface_600
  # block = block_1 -- for debugging
  component = Y
  function = v
  scale factor = 1.0
end

# Z displacements

begin prescribed displacement
  # node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6 nodelist_7 nodelist_8
  surface = surface_100 surface_200 surface_300 surface_400 surface_500 surface_600
  # block = block_1 -- for debugging
  component = Z
  function = w
  scale factor = 1.0
end

# Solver parameters

begin solver
  begin cg
    target relative residual = 1.0e-11
    maximum iterations = 1000
    begin full tangent preconditioner
    end
  end cg
end solver

# Solution verification
# Examine the bounds of the "expected stresses". Note that these stresses are base on
# linear elastic bodies, and thus are not exact for the finite strain formulation.

begin user output
  block = block_1
  compute global sig11max as max of element stress(xx)
  compute global sig11min as min of element stress(xx)
```

```
  compute global sig22max as max of element stress(yy)
  compute global sig22min as min of element stress(yy)
  compute global sig33max as max of element stress(zz)
  compute global sig33min as min of element stress(zz)
  compute global sig12max as max of element stress(xy)
  compute global sig12min as min of element stress(xy)
  compute global sig13max as max of element stress(xz)
  compute global sig13min as min of element stress(xz)
  compute global sig23max as max of element stress(yz)
  compute global sig23min as min of element stress(yz)

  compute global usig11max as max of element cauchy_stress(xx)
  compute global usig11min as min of element cauchy_stress(xx)
  compute global usig22max as max of element cauchy_stress(yy)
  compute global usig22min as min of element cauchy_stress(yy)
  compute global usig33max as max of element cauchy_stress(zz)
  compute global usig33min as min of element cauchy_stress(zz)
  compute global usig12max as max of element cauchy_stress(xy)
  compute global usig12min as min of element cauchy_stress(xy)
  compute global usig13max as max of element cauchy_stress(xz)
  compute global usig13min as min of element cauchy_stress(xz)
  compute global usig23max as max of element cauchy_stress(yz)
  compute global usig23min as min of element cauchy_stress(yz)


  # Relative percent error in each stress component
  compute global sig11err from expression "max(abs(sig11max-{sigNormal}),abs(sig11min-{sigNormal}))*100/
  compute global sig22err from expression "max(abs(sig22max-{sigNormal}),abs(sig22min-{sigNormal}))*100/
  compute global sig33err from expression "max(abs(sig33max-{sigNormal}),abs(sig33min-{sigNormal}))*100/
  compute global sig12err from expression "max(abs(sig12max-{sigShear} ),abs(sig12min-{sigShear} ))*100/
  compute global sig13err from expression "max(abs(sig13max-{sigShear} ),abs(sig13min-{sigShear} ))*100/
  compute global sig23err from expression "max(abs(sig23max-{sigShear} ),abs(sig23min-{sigShear} ))*100/

  compute global usig11err from expression "max(abs(usig11max-{sigNormal}),abs(usig11min-{sigNormal}))*1
  compute global usig22err from expression "max(abs(usig22max-{sigNormal}),abs(usig22min-{sigNormal}))*1
  compute global usig33err from expression "max(abs(usig33max-{sigNormal}),abs(usig33min-{sigNormal}))*1
  compute global usig12err from expression "max(abs(usig12max-{sigShear} ),abs(usig12min-{sigShear} ))*1
  compute global usig13err from expression "max(abs(usig13max-{sigShear} ),abs(usig13min-{sigShear} ))*1
  compute global usig23err from expression "max(abs(usig23max-{sigShear} ),abs(usig23min-{sigShear} ))*1
end


begin solution verification
  skip times = 0.0 to {final_time-epsilon_time}
  completion file = VerifNormal
  verify global sig11err = 0.0
  verify global sig22err = 0.0
  verify global sig33err = 0.0
  tolerance = {normal_stress_tolerance}
end

begin solution verification
  skip times = 0.0 to {final_time-epsilon_time}
  completion file = VerifShear
  verify global sig12err = 0.0
  verify global sig13err = 0.0
  verify global sig23err = 0.0
  tolerance = {shear_stress_tolerance}
end

{if(formulation=="q1p0")}

{Else}

begin solution verification
  skip times = 0.0 to {final_time-epsilon_time}
  completion file = VerifNormalU
  verify global usig11err = 0.0
```

```
        verify global usig22err = 0.0
        verify global usig33err = 0.0
        tolerance = {normal_stress_tolerance}
      end

      begin solution verification
        skip times = 0.0 to {final_time-epsilon_time}
        completion file = VerifShearU
        verify global usig12err = 0.0
        verify global usig13err = 0.0
        verify global usig23err = 0.0
        tolerance = {shear_stress_tolerance}
      end

      {Endif}

    end adagio region adagio_patch_region
  end adagio procedure adagio_patch_procedure

end sierra Hex_patch
```

B.15. HEX PATCH TESTS – QUASI-STATIC, FINITE DEFORMATION

See Section 3.2 for problem description.

```
# ----------------------------------------------------------------------
#
# HexPatchQsFd1Ua-c Tests -- 1Ua,b,c ~ O(1%,10%,100%) strain versions of the FD test
#  The 10% case was not used in the verification problem.
#
# ----------------------------------------------------------------------
#
# Aprepro variable settings that are passed in from the test script....
# meshFile: {meshFile}
# formulation: {formulation}
# strain_incrementation: {strain_incrementation}
# material_model: {material_model}
# normal_stress_tolerance:  {normal_stress_tolerance}
# shear_stress_tolerance:   {shear_stress_tolerance}
# normal_stretch_tolerance: {normal_stretch_tolerance}
# shear_stretch_tolerance:  {shear_stretch_tolerance}
# Note that u_factor_temp is used so u_factor can be defined here to over-ride the test file values.
#  u_factor = 1.0e-1
# -- possible over-ride assigned here -- just add brackets here & remove below.
# u_factor: {u_factor = u_factor_temp}

# Aprepro macros set within this file for all analyses..............
# Format for double precision: {_FORMAT = "%.16g"}
# youngs modulus: {E_young = 1.0e6}
# poissons ratio: {poissons_ratio = 0.25}
# Lame constants
# lambda: {lambda = E_young*poissons_ratio/((1.0+poissons_ratio)*(1.0-2.0*poissons_ratio))}
# mu: {mu = E_young/(2.0*(1.0+poissons_ratio))}
# prescribed displacement debug: {prescribed_displacement_debug = 'off'}

# final_time: {final_time = 1.0}
# hourglass stiffness parameter, originally 0, now set to default (0.05)
# hourglass_stiffness: {hourglass_stiffness = 0.05}
# Expected stretch values
# stretchNormal: {stretchNormal = 1.0 + 2.0*final_time*u_factor}
# stretchShear:  {stretchShear =  final_time*u_factor}
# Expected normal and shear stresses expected from the patch test
# These forms could be simplified, but in this "pass" we are pasting them from Matematica and minor changes
# (with the hope of not introducing editing errors).
{if(material_model=="elastic")}
  # Hypoelastic model: Elastic -- path dependent => steps affect solution and integration accuracy
  # sigNormal: {sigNormal = (3.0*lambda + 2.0*mu)*(2.0*ln(1.0+final_time*u_factor) + ln(1.0+4.0*final_time*u_f
  # sigShear: {sigShear  =
(2.0*mu*(-ln(1.0+final_time*u_factor) + ln(1.0+4.0*final_time*u_factor)))/3.0}
  {if(u_factor==1.0E-2)}
    {if(strain_incrementation=="strongly_objective")}
      # number_steps: {number_steps = 10}
    {else}
      # number_steps: {number_steps = 200}
    {endif}
  {else}  # O(100%) strain case
    {if(strain_incrementation=="strongly_objective")}
      # number_steps: {number_steps = 100}
    {else}
      # number_steps: {number_steps = 10000}
    {endif}
  {endif}
{elseif(material_model=="neo_hookean")}
  # Hyperelastic model: Neo_Hookean -- path independent => using steps to aid solver
  # sigNormal: {sigNormal = (final_time*(3.0*lambda + 2.0*mu)*u_factor*(12.0 + 54.0*final_time*u_factor + 116.
```

```
          129*final_time**3.0*u_factor**3.0 + 72.0*final_time**4*u_factor**4 +
          16*final_time**5*u_factor**5))/(6.*(1 + final_time*u_factor)**2*(1 + 4*final_time*u_factor))}
  # sigShear: {sigShear  =
(final_time*mu*u_factor*(2 + 5*final_time*u_factor))/((1 + final_time*u_factor)**2*(1 + 4*final_time*u_factor)
  {if(u_factor==1.0E-2)}
    {if(strain_incrementation=="strongly_objective")}
      # number_steps: {number_steps = 2}
    {else}
      # number_steps: {number_steps = 200}
    {endif}
  {else}  # O(100%) strain case
    {if(strain_incrementation=="strongly_objective")}
      # number_steps: {number_steps = 100}
    {else}
      # number_steps: {number_steps = 10000}
    {endif}
  {endif}
{else}
  # Test group is not implemented for material model {material_model}.
{Endif}
# epsilon_time is an off set from the final_time used to ignore all results
# except those associated with the final time step.
# epsilon_time: {epsilon_time = final_time/(number_steps*2)}

#-----------------------------------------------------------------------

begin sierra Hex_patch

  # Units
  # length: inches
  # force: lbs
  # stress: psi
  # time: seconds (pseudo-time for these quasistatic runs)

  # Ux displacement component as a function of space and time
  begin definition for function u
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(2.0*x+y+z)"
  end definition for function u

  # Uy displacement component as a function of space and time
  begin definition for function v
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(x+2.0*y+z)"
  end definition for function v

  # Uz displacement component as a function of space and time
  begin definition for function w
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(x+y+2.0*z)"
  end definition for function w

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0
```

```
begin material block_mat
  density         = 1.0e-4
  begin parameters for model {material_model}
    youngs modulus  = {E_young}
    poissons ratio  = {poissons_ratio}
  end parameters for model {material_model}
end material block_mat

begin solid section solid_1
  strain incrementation = {strain_incrementation}
  formulation           = {formulation}
  {if(formulation=="selective_deviatoric")}
    deviatoric parameter = 0.5
  {Else}
    # not a selective_deviatoric formulation
  {Endif}
end

begin finite element model mesh1
  Database Name = {meshFile}
  Database Type = exodusII

  begin parameters for block block_1
    material = block_mat
    model = {material_model}
    section = solid_1
    {if(formulation=="mean_quadrature")}
      linear bulk viscosity = 0.0
      quadratic bulk viscosity = 0.0
      hourglass stiffness = {hourglass_stiffness}
      hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_1

  begin parameters for block block_2
    material = block_mat
    model = {material_model}
    section = solid_1
    {if(formulation=="mean_quadrature")}
      linear bulk viscosity = 0.0
      quadratic bulk viscosity = 0.0
      hourglass stiffness = {hourglass_stiffness}
      hourglass viscosity = 0.0
    {Endif}
   end parameters for block block_2

  begin parameters for block block_3
    material = block_mat
    model = {material_model}
    section = solid_1
    {if(formulation=="mean_quadrature")}
      linear bulk viscosity = 0.0
      quadratic bulk viscosity = 0.0
      hourglass stiffness = {hourglass_stiffness}
      hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_3

  begin parameters for block block_4
    material = block_mat
    model = {material_model}
    section = solid_1
    {if(formulation=="mean_quadrature")}
      linear bulk viscosity = 0.0
      quadratic bulk viscosity = 0.0
      hourglass stiffness = {hourglass_stiffness}
      hourglass viscosity = 0.0
    {Endif}
```

```
  end parameters for block block_4

  begin parameters for block block_5
    material = block_mat
    model = {material_model}
    section = solid_1
    {if(formulation=="mean_quadrature")}
      linear bulk viscosity = 0.0
      quadratic bulk viscosity = 0.0
      hourglass stiffness = {hourglass_stiffness}
      hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_5

  begin parameters for block block_6
    material = block_mat
    model = {material_model}
    section = solid_1
    {if(formulation=="mean_quadrature")}
      linear bulk viscosity = 0.0
      quadratic bulk viscosity = 0.0
      hourglass stiffness = {hourglass_stiffness}
      hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_6

  begin parameters for block block_7
    material = block_mat
    model = {material_model}
    section = solid_1
    {if(formulation=="mean_quadrature")}
      linear bulk viscosity = 0.0
      quadratic bulk viscosity = 0.0
      hourglass stiffness = {hourglass_stiffness}
      hourglass viscosity = 0.0
    {Endif}
  end parameters for block block_7

end finite element model mesh1

begin feti equation solver feti
end

begin adagio procedure adagio_patch_procedure

  begin time control
    begin time stepping block ramp_segment
      start time = 0.0
      begin parameters for adagio region adagio_patch_region
        time increment = {final_time/number_steps}
      end parameters for adagio region adagio_patch_region
    end time stepping block ramp_segment
    termination time = {final_time}
  end time control

  begin adagio region adagio_patch_region
    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = hex_patch.e
      Database Type = exodusII
      At Time 0.0, Increment = {final_time}
      nodal Variables = force_external as f_ext
      nodal Variables = force_internal as f_int
      nodal Variables = displacement

      element Variables = stress as stress_el
```

```
  element Variables = log_strain as log_strain_el
  element variables = min_principal_log_strain as principal_log_strain_min
  element variables = intermediate_principal_log_strain as principal_log_strain_int
  element variables = max_principal_log_strain as principal_log_strain_max
  element variables = unrotated_stress
  element variables = cauchy_stress
  element variables = rotation
  element variables = left_stretch as V_stretch
  element variables = timestep as elem_time_step
  global Variables = timestep as timestep
  global variables = external_energy as ExternalEnergy
  global variables = internal_energy as InternalEnergy
end results output output_adagio

### definition of BCs ###

# For Hex8 mesh the union of the following nodesets gives all the boundary nodes:
#    nodelist_1 ... nodelist_8
# For the Hex27 mesh the union of the following nodesets gives all the boundary nodes::
#   nodelist_1 ... nodelist_26
# To simplify the BC spectification for now use the 6 surfaces of the block.

# X displacements

begin prescribed displacement
  # node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6 nodelist_7 nodelist_8
  {if(prescribed_displacement_debug == 'on')}
    block = block_1 # -- for debugging
  {else}
    surface = surface_100 surface_200 surface_300 surface_400 surface_500 surface_600
  {endif}
  component = X
  function = u
  scale factor = 1.0
end

# Y displacements

begin prescribed displacement
  # node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6 nodelist_7 nodelist_8
  {if(prescribed_displacement_debug == 'on')}
    block = block_1 # -- for debugging
  {else}
    surface = surface_100 surface_200 surface_300 surface_400 surface_500 surface_600
  {endif}
  component = Y
  function = v
  scale factor = 1.0
end

# Z displacements

begin prescribed displacement
  # node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6 nodelist_7 nodelist_8
  {if(prescribed_displacement_debug == 'on')}
    block = block_1 # -- for debugging
  {else}
    surface = surface_100 surface_200 surface_300 surface_400 surface_500 surface_600
  {endif}
  component = Z
  function = w
  scale factor = 1.0
end

# Solver parameters

begin solver
  {if(formulation=="q1p0")}
```

```
  begin loadstep predictor
    type = scale_factor
    scale factor = 0.0
  end
  {else}
  begin loadstep predictor
    type = scale_factor
    scale factor = 1.0 0.0
  end
  {endif}
  begin cg
    # value of usable tolerance depends on the number of steps used
    # The value of 1e-14 was controled by the hyperelastic model using 10 step
    {if(formulation=="q1p0")}
      # don't push this one, since it does not "pass" the patch test anyway
      target relative residual     = 1.0e-9
      acceptable relative residual = 1.0e1
      maximum iterations = 25
    {else}
      target relative residual = 1.0e-14
      # initially used for debugging with all displacements constrained, but retained for all runs
      # Let the verification statements determine if this was too loose for the hypo cases.
      acceptable relative residual = 1.0e-1
      maximum iterations = 20
      line search secant 1e-6
      begin full tangent preconditioner
        # take more drastic action and update tangent more
        # It's a small problem anyway, so tangents are not costly from an absolute perspective.
        {if(formulation=="q1p0")}
        iteration update = 5
        {else}
        iteration update = 100
        {endif}
        linear solver = feti
        conditioning = no_check
      end

    {endif}
  end cg
end solver

# Solution verification
# Examine the bounds of the "expected stresses".

begin user output
  block = block_1
  # maximum and minimum of stress components
  compute global sig11max as max of element stress(xx)
  compute global sig11min as min of element stress(xx)
  compute global sig22max as max of element stress(yy)
  compute global sig22min as min of element stress(yy)
  compute global sig33max as max of element stress(zz)
  compute global sig33min as min of element stress(zz)
  compute global sig12max as max of element stress(xy)
  compute global sig12min as min of element stress(xy)
  compute global sig13max as max of element stress(xz)
  compute global sig13min as min of element stress(xz)
  compute global sig23max as max of element stress(yz)
  compute global sig23min as min of element stress(yz)

  # maximum and minimum of left_stretch components
  compute global V11max as max of element left_stretch(xx)
  compute global V11min as min of element left_stretch(xx)
  compute global V22max as max of element left_stretch(yy)
  compute global V22min as min of element left_stretch(yy)
  compute global V33max as max of element left_stretch(zz)
  compute global V33min as min of element left_stretch(zz)
  compute global V12max as max of element left_stretch(xy)
```

```
  compute global V12min as min of element left_stretch(xy)
  compute global V13max as max of element left_stretch(xz)
  compute global V13min as min of element left_stretch(xz)
  compute global V23max as max of element left_stretch(yz)
  compute global V23min as min of element left_stretch(yz)

  compute global usig11max as max of element cauchy_stress(xx)
  compute global usig11min as min of element cauchy_stress(xx)
  compute global usig22max as max of element cauchy_stress(yy)
  compute global usig22min as min of element cauchy_stress(yy)
  compute global usig33max as max of element cauchy_stress(zz)
  compute global usig33min as min of element cauchy_stress(zz)
  compute global usig12max as max of element cauchy_stress(xy)
  compute global usig12min as min of element cauchy_stress(xy)
  compute global usig13max as max of element cauchy_stress(xz)
  compute global usig13min as min of element cauchy_stress(xz)
  compute global usig23max as max of element cauchy_stress(yz)
  compute global usig23min as min of element cauchy_stress(yz)

  # Relative factional error in each stress component
  # Changed from percent to fractional to more easily judge the number of accurate digits
  compute global sig11err from expression "max(abs(sig11max-{sigNormal}),abs(sig11min-{sigNormal}))/{sig
  compute global sig22err from expression "max(abs(sig22max-{sigNormal}),abs(sig22min-{sigNormal}))/{sig
  compute global sig33err from expression "max(abs(sig33max-{sigNormal}),abs(sig33min-{sigNormal}))/{sig
  compute global sigiierr from expression "max(sig11err,sig22err,sig33err)"
  compute global sig12err from expression "max(abs(sig12max-{sigShear} ),abs(sig12min-{sigShear} ))/{sig
  compute global sig13err from expression "max(abs(sig13max-{sigShear} ),abs(sig13min-{sigShear} ))/{sig
  compute global sig23err from expression "max(abs(sig23max-{sigShear} ),abs(sig23min-{sigShear} ))/{sig
  compute global sigijerr from expression "max(sig12err,sig13err,sig23err)"

  compute global usig11err from expression "max(abs(usig11max-{sigNormal}),abs(usig11min-{sigNormal}))/{
  compute global usig22err from expression "max(abs(usig22max-{sigNormal}),abs(usig22min-{sigNormal}))/{
  compute global usig33err from expression "max(abs(usig33max-{sigNormal}),abs(usig33min-{sigNormal}))/{
  compute global usigiierr from expression "max(usig11err,usig22err,usig33err)"
  compute global usig12err from expression "max(abs(usig12max-{sigShear} ),abs(usig12min-{sigShear} ))/{
  compute global usig13err from expression "max(abs(usig13max-{sigShear} ),abs(usig13min-{sigShear} ))/{
  compute global usig23err from expression "max(abs(usig23max-{sigShear} ),abs(usig23min-{sigShear} ))/{
  compute global usigijerr from expression "max(usig12err,usig13err,usig23err)"

  # Relative factional error in each left_stretch component
  compute global V11err from expression "max(abs(V11max-{stretchNormal}),abs(V11min-{stretchNormal}))/{s
  compute global V22err from expression "max(abs(V22max-{stretchNormal}),abs(V22min-{stretchNormal}))/{s
  compute global V33err from expression "max(abs(V33max-{stretchNormal}),abs(V33min-{stretchNormal}))/{s
  compute global Viierr from expression "max(V11err,V22err,V33err)"
  compute global V12err from expression "max(abs(V12max-{stretchShear} ),abs(V12min-{stretchShear} ))/{s
  compute global V13err from expression "max(abs(V13max-{stretchShear} ),abs(V13min-{stretchShear} ))/{s
  compute global V23err from expression "max(abs(V23max-{stretchShear} ),abs(V23min-{stretchShear} ))/{s
  compute global Vijerr from expression "max(V12err,V13err,V23err)"

  # Put tolerances in global variables so they can be output to the heartbeat file
  compute global normal_stretch_tolerance from expression "{normal_stretch_tolerance}"
  compute global shear_stretch_tolerance  from expression "{shear_stretch_tolerance}"
  compute global normal_stress_tolerance  from expression "{normal_stress_tolerance}"
  compute global shear_stress_tolerance   from expression "{shear_stress_tolerance}"

end

# heartbeat files for creating documentation LaTex tables
begin heartbeat output deformation_tabular_output
  stream name = hex_patch_deformation_error_table.csv
  labels = off
  legend = off
  format = spyhis  # a csv file
  start time = {final_time-epsilon_time}
  at step 1 increment = 1
  termination time = {final_time}
  # global time  -- unnecessary with spyhis format
  # commented-out data for debugging
```

```
  #global V11min
  #global V11max
  #global V11err
  #global V22err
  #global V33err
  global Viierr
  global normal_stretch_tolerance
  #global V12err
  #global V13err
  #global V23err
  global Vijerr
  global shear_stretch_tolerance
end heartbeat output deformation_tabular_output

begin heartbeat output stress_tabular_output
  stream name = hex_patch_stress_error_table.csv
  labels = off
  legend = off
  format = spyhis  # a csv file
  start time = {final_time-epsilon_time}
  at step 1 increment = 1
  termination time = {final_time}
  # global time  -- unnecessary with spyhis format
  global sigiierr
  global normal_stress_tolerance
  global sigijerr
  global shear_stress_tolerance
end heartbeat output stress_tabular_output

begin solution verification
  skip times = 0.0 to {final_time-epsilon_time}
  completion file = VerifSigNormal
  verify global sig11err = 0.0
  verify global sig22err = 0.0
  verify global sig33err = 0.0
  tolerance = {normal_stress_tolerance}
end

begin solution verification
  skip times = 0.0 to {final_time-epsilon_time}
  completion file = VerifSigShear
  verify global sig12err = 0.0
  verify global sig13err = 0.0
  verify global sig23err = 0.0
  tolerance = {shear_stress_tolerance}
end

{if(formulation=="q1p0")}

{Else}

begin solution verification
  skip times = 0.0 to {final_time-epsilon_time}
  completion file = VerifUSigNormal
  verify global usig11err = 0.0
  verify global usig22err = 0.0
  verify global usig33err = 0.0
  tolerance = {normal_stress_tolerance}
end

begin solution verification
  skip times = 0.0 to {final_time-epsilon_time}
  completion file = VerifUSigShear
  verify global usig12err = 0.0
  verify global usig13err = 0.0
  verify global usig23err = 0.0
  tolerance = {shear_stress_tolerance}
end
```

```
      {Endif}

      begin solution verification
        skip times = 0.0 to {final_time-epsilon_time}
        completion file = VerifVNormal
        verify global V11err = 0.0
        verify global V22err = 0.0
        verify global V33err = 0.0
        tolerance = {normal_stretch_tolerance}
      end

      begin solution verification
        skip times = 0.0 to {final_time-epsilon_time}
        completion file = VerifVShear
        verify global V12err = 0.0
        verify global V13err = 0.0
        verify global V23err = 0.0
        tolerance = {shear_stretch_tolerance}
      end

    end adagio region adagio_patch_region
  end adagio procedure adagio_patch_procedure

end sierra Hex_patch
```

B.16. HEX PATCH TEST – UNIFORM GRADIENT, STRONGLY OBJECTIVE

See Section 3.3 for problem description.

```
begin sierra

  begin function unit
    type is piecewise linear
    ordinate is time
    abscissa is unit
    begin values
      0.0      0.0
      0.0001   6.15581932461301e-06
      0.0002   2.44717008522228e-05
      0.0003   5.44966475530096e-05
      0.0004   9.54913468384886e-05
      0.0005   0.000146446374860387
      0.0006   0.000206107051833971
      0.0007   0.000273004336366306
      0.0008   0.00034549099806984
      0.0009   0.000421782177773008
      0.001    0.000499999336602552
      0.0011   0.000578216511767115
      0.0012   0.000654507740073118
      0.0013   0.000726994481450545
      0.0014   0.000793891874765996
      0.0015   0.000853552686953322
      0.0016   0.000904507873290184
      0.0017   0.000945502750093614
      0.0018   0.000975527889144267
      0.0019   0.000993843973117372
      0.002    0.00099999999999824
      0.0025   0.001
    end values
  end function unit

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin material linear_elastic
    density         = 2.61e-4

    begin parameters for model elastic
      youngs modulus  = 1.e6
      poissons ratio  = 0.25
    end parameters for model elastic

    begin parameters for model elastic_plastic
      youngs modulus  = 1.e6
      poissons ratio  = 0.25
      yield stress = 1.0e6
      hardening modulus = 10.0
      beta is 0.999999
    end parameters for model elastic_plastic
  end material linear_elastic

  begin solid section solid_1
    strain incrementation = strongly_objective
  end solid section solid_1

  begin finite element model mesh1
    Database Name = ug3dh8_so_patch_test.g
    Database Type = exodusII
```

```
begin parameters for block block_1
  material = linear_elastic
  model = elastic
  linear bulk viscosity = 0.0
  quadratic bulk viscosity = 0.0
  hourglass stiffness = 0.0
  hourglass viscosity = 0.0
  section = solid_1
end parameters for block block_1

begin parameters for block block_2
  material = linear_elastic
  model = elastic
  linear bulk viscosity = 0.0
  quadratic bulk viscosity = 0.0
  hourglass stiffness = 0.0
  hourglass viscosity = 0.0
  section = solid_1
end parameters for block block_2

begin parameters for block block_3
  material = linear_elastic
  model = elastic
  linear bulk viscosity = 0.0
  quadratic bulk viscosity = 0.0
  hourglass stiffness = 0.0
  hourglass viscosity = 0.0
  section = solid_1
end parameters for block block_3

begin parameters for block block_4
  material = linear_elastic
  model = elastic
  linear bulk viscosity = 0.0
  quadratic bulk viscosity = 0.0
  hourglass stiffness = 0.0
  hourglass viscosity = 0.0
  section = solid_1
end parameters for block block_4

begin parameters for block block_5
  material = linear_elastic
  model = elastic
  linear bulk viscosity = 0.0
  quadratic bulk viscosity = 0.0
  hourglass stiffness = 0.0
  hourglass viscosity = 0.0
  section = solid_1
end parameters for block block_5

begin parameters for block block_6
  material = linear_elastic
  model = elastic
  linear bulk viscosity = 0.0
  quadratic bulk viscosity = 0.0
  hourglass stiffness = 0.0
  hourglass viscosity = 0.0
  section = solid_1
end parameters for block block_6

begin parameters for block block_7
  material = linear_elastic
  model = elastic
  linear bulk viscosity = 0.0
  quadratic bulk viscosity = 0.0
  hourglass stiffness = 0.0
  hourglass viscosity = 0.0
  section = solid_1
```

```
  end parameters for block block_7

end finite element model mesh1

begin presto procedure Apst_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region presto
        time step scale factor = 1.0
        time step increase factor = 2.0
        step interval = 100
      end parameters for presto region presto
    end time stepping block p1

    termination time = 0.0025
  end time control

  begin presto region presto
    use finite element model mesh1

    begin node based time step parameters
      step interval = 100
    end

    begin mass scaling
      block = block_5
      allowable mass increase ratio = 1.2
    end mass scaling


    ### output description ###
    begin Results Output output_adagio
      Database Name = ug3dh8_so_patch_test.e
      Database Type = exodusII
      At Time 0.0, Increment = 1.0E-4
      nodal Variables = force_external as f_ext
      nodal Variables = force_internal as f_int
      nodal Variables = velocity as vel
      nodal Variables = acceleration as acc
      nodal Variables = displacement as displ
      element Variables = stress as stress
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
    end results output output_adagio

    ### definition of BCs ###
    begin fixed displacement
      node set = nodelist_1
      components = X Y Z
    end

    # X displacements

    begin prescribed displacement
      node set = nodelist_4 nodelist_5
      component = X
      function = unit
      scale factor = 1.0
    end

    begin prescribed displacement
      node set = nodelist_2 nodelist_8
      component = X
```

```
  function = unit
  scale factor = 2.0
end

begin prescribed displacement
  node set = nodelist_3 nodelist_6
  component = X
  function = unit
  scale factor = 3.0
end

begin prescribed displacement
  node set = nodelist_7
  component = X
  function = unit
  scale factor = 4.0
end

# Y displacements

begin prescribed displacement
  node set = nodelist_2 nodelist_5
  component = Y
  function = unit
  scale factor = 1.0
end

begin prescribed displacement
  node set = nodelist_4 nodelist_6
  component = Y
  function = unit
  scale factor = 2.0
end

begin prescribed displacement
  node set = nodelist_3 nodelist_8
  component = Y
  function = unit
  scale factor = 3.0
end

begin prescribed displacement
  node set = nodelist_7
  component = Y
  function = unit
  scale factor = 4.0
end

# Z displacements

begin prescribed displacement
  node set = nodelist_2 nodelist_4
  component = Z
  function = unit
  scale factor = 1.0
end

begin prescribed displacement
  node set = nodelist_3 nodelist_5
  component = Z
  function = unit
  scale factor = 2.0
end

begin prescribed displacement
  node set = nodelist_6 nodelist_8
  component = Z
  function = unit
```

```
        scale factor = 3.0
      end

      begin prescribed displacement
        node set = nodelist_7
        component = Z
        function = unit
        scale factor = 4.0
      end

    end presto region presto
  end presto procedure Apst_Procedure

end sierra
```

B.17. HEX PATCH TEST – UNIFORM GRADIENT, MIDPOINT INCREMENT

See Section 3.4 for problem description.

```
begin sierra

  begin function unit
    type is piecewise linear
    ordinate is time
    abscissa is unit
    begin values
      0.0      0.0
      0.0001   6.15581932461301e-06
      0.0002   2.44717008522228e-05
      0.0003   5.44966475530096e-05
      0.0004   9.54913468384886e-05
      0.0005   0.000146446374860387
      0.0006   0.000206107051833971
      0.0007   0.000273004336366306
      0.0008   0.00034549099806984
      0.0009   0.000421782177773008
      0.001    0.000499999336602552
      0.0011   0.000578216511767115
      0.0012   0.000654507740073118
      0.0013   0.000726994481450545
      0.0014   0.000793891874765996
      0.0015   0.000853552686953322
      0.0016   0.000904507873290184
      0.0017   0.000945502750093614
      0.0018   0.000975527889144267
      0.0019   0.000993843973117372
      0.002    0.00099999999999824
      0.0025   0.001
    end values
  end function unit

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin material linear_elastic
    density         = 2.61e-4

    begin parameters for model elastic
      youngs modulus  = 1.e6
      poissons ratio  = 0.25
    end parameters for model elastic

    begin parameters for model elastic_plastic
      youngs modulus  = 1.e6
      poissons ratio  = 0.25
      yield stress = 1.0e6
      hardening modulus = 10.0
      beta is 0.999999
    end parameters for model elastic_plastic
  end material linear_elastic

  begin finite element model mesh1
    Database Name = ug3dh8_mi_patch_test.g
    Database Type = exodusII

    begin parameters for block
      include all blocks
      remove block = block_1 block_2 block_3
      material = linear_elastic
```

```
    model = elastic
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
  end

  begin parameters for block
    include all blocks
    remove block = block_4 block_5 block_6 block_7
    material = linear_elastic
    model = elastic
    linear bulk viscosity = 0.0
    quadratic bulk viscosity = 0.0
    hourglass stiffness = 0.0
    hourglass viscosity = 0.0
  end

end finite element model mesh1

begin presto procedure Apst_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region presto
        time step scale factor = 1.0
        time step increase factor = 2.0
        step interval = 100
      end parameters for presto region presto
    end time stepping block p1

    termination time = 0.0025
  end time control

  begin presto region presto
    use finite element model mesh1

    begin node based time step parameters
      step interval = 100
    end

    ### output description ###
    begin Results Output output_adagio
      Database Name = ug3dh8_mi_patch_test.e
      Database Type = exodusII
      At Time 0.0, Increment = 1.0E-4
      nodal Variables = force_external as f_ext
      nodal Variables = force_internal as f_int
      nodal Variables = velocity as vel
      nodal Variables = acceleration as acc
      nodal Variables = displacement as displ
      nodal variables = mass_scaling_added_mass
      element Variables = stress as stress
      element variables = timestep as elem_time_step
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
    end results output output_adagio

    ### definition of BCs ###
    begin fixed displacement
      node set = nodelist_1
      components = X Y Z
    end
```

```
#
#  Test mass scaling capability
#
begin mass scaling
  include all blocks
  target time step = 3.5e-06
end


# X displacements

begin prescribed displacement
  node set = nodelist_4 nodelist_5
  component = X
  function = unit
  scale factor = 1.0
end

begin prescribed displacement
  node set = nodelist_2 nodelist_8
  component = X
  function = unit
  scale factor = 2.0
end

begin prescribed displacement
  node set = nodelist_3 nodelist_6
  component = X
  function = unit
  scale factor = 3.0
end

begin prescribed displacement
  node set = nodelist_7
  component = X
  function = unit
  scale factor = 4.0
end

# Y displacements

begin prescribed displacement
  node set = nodelist_2 nodelist_5
  component = Y
  function = unit
  scale factor = 1.0
end

begin prescribed displacement
  node set = nodelist_4 nodelist_6
  component = Y
  function = unit
  scale factor = 2.0
end

begin prescribed displacement
  node set = nodelist_3 nodelist_8
  component = Y
  function = unit
  scale factor = 3.0
end

begin prescribed displacement
  node set = nodelist_7
  component = Y
  function = unit
  scale factor = 4.0
end
```

```
      # Z displacements

      begin prescribed displacement
        node set = nodelist_2 nodelist_4
        component = Z
        function = unit
        scale factor = 1.0
      end

      begin prescribed displacement
        node set = nodelist_3 nodelist_5
        component = Z
        function = unit
        scale factor = 2.0
      end

      begin prescribed displacement
        node set = nodelist_6 nodelist_8
        component = Z
        function = unit
        scale factor = 3.0
      end

      begin prescribed displacement
        node set = nodelist_7
        component = Z
        function = unit
        scale factor = 4.0
      end

    end presto region presto
  end presto procedure Apst_Procedure

end sierra
```

B.18. HEX PATCH TEST – UNIFORM GRADIENT, MIDPOINT INCREMENT THERMAL

See Section 3.5 for problem description.

```
begin sierra thermal_strains_w_dispload

  begin function unit
    type is piecewise linear
    ordinate is time
    abscissa is unit
    begin values
      0.0      0.0
      0.0001   0.00615581932461301
      0.0002   0.0244717008522228
      0.0003   0.0544966475530096
      0.0004   0.0954913468384886
      0.0005   0.146446374860387
      0.0006   0.206107051833971
      0.0007   0.273004336366306
      0.0008   0.34549099806984
      0.0009   0.421782177773008
      0.001    0.499999336602552
      0.0011   0.578216511767115
      0.0012   0.654507740073119
      0.0013   0.726994481450545
      0.0014   0.793891874765996
      0.0015   0.853552686953322
      0.0016   0.904507873290184
      0.0017   0.945502750093614
      0.0018   0.975527889144267
      0.0019   0.993843973117372
      0.002    0.99999999999824
      0.0025   1
    end values
  end function unit

  begin function THERMAL_STRAIN
    type is piecewise linear
    ordinate is strain
    abscissa is temperature
    begin values
      0.0    0.0
      10.0    0.01
    end values
  end function THERMAL_STRAIN

  begin material linear_elastic
    density         = 2.61e-4
    thermal strain function = THERMAL_STRAIN

    begin parameters for model elastic
      youngs modulus  = 1.e6
      poissons ratio  = 0.25
    end parameters for model elastic

    begin parameters for model elastic_plastic
      youngs modulus  = 1.e6
      poissons ratio  = 0.25
      yield stress = 1.0e6
      hardening modulus = 10.0
      beta is 0.999999
    end parameters for model elastic_plastic
  end material linear_elastic

  begin finite element model mesh1
```

```
  Database Name = thermal_strains_nobc.g
  Database Type = exodusII

  begin block defaults
    material = linear_elastic
    model = elastic
  end block defaults

  begin parameters for block block_1 block_2 block_3 block_4 block_5 block_6 block_7
  end

end finite element model mesh1

begin presto procedure Apst_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region presto
        time step scale factor = 1.0
        time step increase factor = 2.0
        step interval = 100
      end parameters for presto region presto
    end time stepping block p1

    termination time = 0.003
  end time control

  begin presto region presto

    use finite element model mesh1

    begin prescribed temperature
      function = unit
      scale factor = 1.0
      include all blocks
    end

    ### output description ###
    begin Results Output output_adagio
      Database Name = thermal_strains_nobc.e
      Database Type = exodusII
      At Time 0.0, Increment = 1.0E-4
      nodal Variables = force_external as f_ext
      nodal Variables = force_internal as f_int
      nodal Variables = velocity as vel
      nodal Variables = acceleration as acc
      nodal Variables = displacement as displ
      element Variables = stress as stress
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
    end results output output_adagio

    ### definition of BCs ###
    begin fixed displacement
      node set = nodelist_1
      components = X Y Z
    end fixed displacement

    begin fixed displacement
      node set = nodelist_2
      components = Y Z
    end fixed displacement

    begin fixed displacement
```

```
        node set = nodelist_3
        components = Z
      end fixed displacement

      begin fixed displacement
        node set = nodelist_4
        components = X Z
      end fixed displacement

      begin fixed displacement
        node set = nodelist_5
        components = X Y
      end fixed displacement

      begin fixed displacement
        node set = nodelist_6
        components = y
      end fixed displacement

      begin fixed displacement
        node set = nodelist_8
        components = X
      end fixed displacement

    end presto region presto
  end presto procedure Apst_Procedure

end sierra thermal_strains_w_dispload
```

B.19. HEX CONVERGENCE TEST – CANTILEVER BEAM

See Section 3.6 for problem description.

```
# Set default for the mesh to help FCT
#{mesh="2"}
# ------------------------------------------------------------------
#
# Aprepro Variable Settings
# mesh: {mesh}
# formulation: {formulation}
# Formulation flags: 0~off, 1~on
# mean_quad: {mean_quad}
# selective_dev: {selective_dev}
# strain_incrementation: {strain_incrementation}
# material_model: {material_model}
# elem_topo: {elem_topo}
#-------------------------------------------------------------------
#

begin Sierra slender_beam_example

# slender beam example -- constant shear load
#
# Cantilever support at negative-x end with
# transverse shear load at positive-x end
# use length-to-depth ratio of 10

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

# Loading function ----------------------------------------------------

   begin definition for function load
      type        = piecewise linear
      ordinate    = shear
      abscissa    = time
      begin values
         0.0   0.0
         1.0   1.0
      end values
   end definition for function load

# material data -------------------------------------------------------

   begin property specification for material beam_stuff
      density          = 1.0E-2
      begin parameters for model {material_model}
         youngs modulus = 1.0E6
         poissons ratio = 0.3
      end parameters for model {material_model}
   end property specification for material beam_stuff

# section data (required to create non-default hex elements) ----------

   begin solid section solid_section
     strain incrementation = {strain_incrementation}
     formulation = {formulation}
     {Ifdef(selective_dev)}
     deviatoric parameter = 0.5
     {Else}
     # Undefined selective_dev
     {Endif}
    end solid section solid_section

# FE model ------------------------------------------------------------
```

```
   begin finite element model slender_beam
     database name = beam{mesh}_{elem_topo}.g
     database type = exodusII

     begin parameters for block block_1
       material beam_stuff
       solid mechanics use model {material_model}
       section = solid_section
       {Ifdef(mean_quad)}
       linear bulk viscosity = 0.06
       quadratic bulk viscosity = 1.20
       hourglass stiffness = 0.05
       hourglass viscosity = 0.0
       {Else}
       # Undefined mean_quad
       {Endif}
      end parameters for block block_1

   end finite element model slender_beam

# procedure data ------------------------------------------------------

   begin adagio procedure beam_procedure

      begin time control

         begin time stepping block
            start time = 0.0
            begin parameters for adagio region beam_region
               time increment = 0.2
            end parameters for adagio region beam_region
         end time stepping block

         termination time = 1.0

      end time control

      begin adagio region beam_region

         use finite element model slender_beam

# BC data -------------------------------------------------------------

         # traction load on right end
         begin traction
            surface = right_ss
            Node Set Subroutine = parabolic_Yshear
            Scale factor = 1.0
         end traction

         # traction reaction on left end
         begin traction
            surface = left_ss
            Node Set Subroutine = parabolic_Yshear
            scale factor = -1.0
         end traction

         # displacement bcs on left end to prevent rigid body motions
         # and prescribe fixity with respect to x
         begin fixed displacement
            surface = left_ss
            Component = X
         end fixed displacement
         begin fixed displacement
            node set = left_y_line
            Component = Y
         end fixed displacement
```

```
         begin fixed displacement
            node set = left_z_line
            Component = Z
         end fixed displacement

# Output data ----------------------------------------------------------

         begin user output
            include all blocks
            extrapolate element variable stress to nodal variable nodal_stress
         end

         begin results output beam_output
            database name = beam{mesh}_Adagio.e
            database type = exodusII
            at step 0 increment = 1
            nodal variables = displacement as displ
            nodal variables = nodal_stress as nodal_stress
            element variables = stress as elem_stress
                  nodal variables = force_external as fext
         end results output beam_output

# solver data ----------------------------------------------------------

         begin solver
            begin cg
              target relative residual = 1e-8
#               acceptable relative residual = 1e-12
#               {Ifdef(mean_quad)}
#               target relative residual = 0.06
#               {Else}
#               target relative residual = 1e-9
#               {Endif}
              maximum iterations = 1000
              begin full tangent preconditioner
                small number of iterations = 15
                conditioning = no_check
              end
            end
         end solver

      end adagio region beam_region

   end adagio procedure beam_procedure

end sierra slender_beam_example
```

B.20. TET PATCH TESTS – QUASI-STATIC, LINEAR ELASTIC

See Section 3.7 for problem description.

```
begin sierra Tet_patch

  # Units
  # length: inches
  # force: lbs
  # stress: psi
  # time: seconds (pseudo-time for these quasistatic runs)

  # Aprepro macros ..............
  # {u_factor = 1.0E-4}
  # {final_time = 1.0}
  # {number_steps = 2}
  # Expected normal and shear stresses expected from the patch test -- a linear elastic approximation
  # Note that these could be obtained more symbolically to generalize the test.
  # {sigNormal = 400.0}
  # {sigShear =   80.0}
  # epsilon_time is an off set from the final_time used to ignore all results
  # except those associated with the final time step.
  # { epsilon_time = final_time/(number_steps*2)}

  # Required externally provided macros with string values:
  #   normal_stress_tolerance
  #   shear_stress_tolerance
  #   strain_incrementation
  #   formulation
  #   meshFile
  #   material model
  # Required externally provided macros "on" or "off":
  #   nodal_tet
  #   composite_tet

  # Ux displacement component as a function of space and time
  begin definition for function u
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(2.0*x+y+z)"
  end definition for function u

  # Uy displacement component as a function of space and time
  begin definition for function v
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(x+2.0*y+z)"
  end definition for function v

  # Uz displacement component as a function of space and time
  begin definition for function w
    type = analytic
    expression variable: x = nodal model_coordinates(1)
    expression variable: y = nodal model_coordinates(2)
    expression variable: z = nodal model_coordinates(3)
    expression variable: t = global time
    evaluate expression is "t*{u_factor}*(x+y+2.0*z)"
  end definition for function w

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
```

```
define direction z with vector 0.0 0.0 1.0

begin material block_mat
  density         = 2.61e-4
  begin parameters for model {material_model}
    youngs modulus  = 1.0e6
    poissons ratio  = 0.25
  end parameters for model {material_model}
end material block_mat

{Ifdef(composite_tet)}
begin total lagrange section solid_1
   formulation = composite_tet
end
{Else}
begin solid section solid_1
  strain incrementation = {strain_incrementation}
  formulation           = {formulation}
end
{Endif}

begin finite element model mesh1
  Database Name = {meshFile}
  Database Type = exodusII

  begin parameters for block block_1
    material = block_mat
    model = {material_model}
    section = solid_1
  end parameters for block block_1

end finite element model mesh1

begin adagio procedure adagio_patch_procedure

  begin time control
    begin time stepping block ramp_segment
      start time = 0.0
      begin parameters for adagio region adagio_patch_region
        time increment = {final_time/number_steps}
      end parameters for adagio region adagio_patch_region
    end time stepping block ramp_segment
    termination time = {final_time}
  end time control

  begin adagio region adagio_patch_region
    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = tet_patch.e
      Database Type = exodusII
      At Time 0.0, Increment = {final_time}
      nodal Variables = force_external as f_ext
      nodal Variables = force_internal as f_int
      nodal Variables = displacement
      element Variables = stress as stress_el
      element Variables = log_strain as strain_el
      {Ifdef(nodal_tet)}
      nodal Variables = stress_1 as stress_node
      {Endif}
      element variables = timestep as elem_time_step
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
    end results output output_adagio

    ### definition of BCs ###
```

```
# X displacements

begin prescribed displacement
  node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6
  # block = block_1
  component = X
  function = u
  scale factor = 1.0
end

# Y displacements

begin prescribed displacement
  node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6
  # block = block_1
  component = Y
  function = v
  scale factor = 1.0
end

# Z displacements

begin prescribed displacement
  node set = nodelist_1 nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6
  # block = block_1
  component = Z
  function = w
  scale factor = 1.0
end

# Solver parameters

begin solver
  begin loadstep predictor
    type = scale_factor
    scale factor = 0.0
  end
  begin cg
    target relative residual = 1.0e-11
    {Ifndef(nodal_tet)}
    maximum iterations = 20      # Reasonable for tangent preconditioner
    begin full tangent preconditioner
      linear solver = feti  # explicitly putting this line for feature coverage tool.
    end
    {Else}
    # Parameters for the nodal-based tet
    # These parameters from adagio/tet4n_uni_disp_cube_thermal
    maximum iterations = 20
    orthogonality measure for reset = 0.1
    line search secant
    preconditioner = elastic
    {Endif}

  end cg
end solver

# Solution verification
# Examine the bounds of the "expected stresses". Note that these stresses are based on
# linear elastic bodies, and thus are not exact for the finite strain formulation.

begin user output
  block = block_1
  {Ifndef(nodal_tet)}
  compute global sig11max as max of element stress(xx)
  compute global sig11min as min of element stress(xx)
  compute global sig22max as max of element stress(yy)
  compute global sig22min as min of element stress(yy)
```

```
        compute global sig33max as max of element stress(zz)
        compute global sig33min as min of element stress(zz)
        compute global sig12max as max of element stress(xy)
        compute global sig12min as min of element stress(xy)
        compute global sig13max as max of element stress(xz)
        compute global sig13min as min of element stress(xz)
        compute global sig23max as max of element stress(yz)
        compute global sig23min as min of element stress(yz)

        {Else}
        compute global sig11max as max of nodal stress_1(xx)
        compute global sig11min as min of nodal stress_1(xx)
        compute global sig22max as max of nodal stress_1(yy)
        compute global sig22min as min of nodal stress_1(yy)
        compute global sig33max as max of nodal stress_1(zz)
        compute global sig33min as min of nodal stress_1(zz)
        compute global sig12max as max of nodal stress_1(xy)
        compute global sig12min as min of nodal stress_1(xy)
        compute global sig13max as max of nodal stress_1(xz)
        compute global sig13min as min of nodal stress_1(xz)
        compute global sig23max as max of nodal stress_1(yz)
        compute global sig23min as min of nodal stress_1(yz)
        {Endif}
        # Relative percent error in each stress component
        compute global sig11err from expression "max(abs(sig11max-{sigNormal}),abs(sig11min-{sigNormal}))*100/
        compute global sig22err from expression "max(abs(sig22max-{sigNormal}),abs(sig22min-{sigNormal}))*100/
        compute global sig33err from expression "max(abs(sig33max-{sigNormal}),abs(sig33min-{sigNormal}))*100/
        compute global sig12err from expression "max(abs(sig12max-{sigShear} ),abs(sig12min-{sigShear} ))*100/
        compute global sig13err from expression "max(abs(sig13max-{sigShear} ),abs(sig13min-{sigShear} ))*100/
        compute global sig23err from expression "max(abs(sig23max-{sigShear} ),abs(sig23min-{sigShear} ))*100/
      end

      begin solution verification
        skip times = 0.0 to {final_time-epsilon_time}
        completion file = VerifNormal
        verify global sig11err = 0.0
        verify global sig22err = 0.0
        verify global sig33err = 0.0
        tolerance = {normal_stress_tolerance}
      end

      begin solution verification
        skip times = 0.0 to {final_time-epsilon_time}
        completion file = VerifShear
        verify global sig12err = 0.0
        verify global sig13err = 0.0
        verify global sig23err = 0.0
        tolerance = {shear_stress_tolerance}
      end

    end adagio region adagio_patch_region
  end adagio procedure adagio_patch_procedure

  begin feti equation solver feti
  end

end sierra Tet_patch
```

B.21. TET CONVERGENCE TEST – CANTILEVER BEAM

See Section 3.8 for problem description.

```
# ----------------------------------------------------------------------
#

begin Sierra slender_beam_example

  # slender beam example -- constant shear load
  #
  # Cantilever support at negative-x end with
  # transverse shear load at positive-x end
  # use length-to-depth ratio of 10

  user subroutine file = parabolic_Yshear.F

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0


# Loading function ----------------------------------------------------

  begin definition for function load
    type        = piecewise linear
    ordinate    = shear
    abscissa    = time
    begin values
      0.0   0.0
      1.0   1.0
    end values
  end definition for function load

# material data -------------------------------------------------------

  begin property specification for material beam_stuff
    density          = 1.0E-2
    begin parameters for model elastic
      youngs modulus = 1.0E6
      poissons ratio = 0.3
    end parameters for model elastic
  end property specification for material beam_stuff

# section data (required to create non-default hex elements)

  begin solid section solid_section
    strain_incrementation = {strain_incrementation}
    formulation = {formulation}
    {Ifdef(composite_tet)}
    #The ten node tet with a composite tet formulation uses a internal force calculation parameter
    internal force calculation = total_lagrange
    #volume average j = on
    #cubature degree = cubature_degree
    {Endif}
  end solid section solid_section

# FE model ------------------------------------------------------------


  begin finite element model slender_beam
    database name = tet{topology}_beam{mesh}.g
    database type = exodusII

    begin parameters for block block_1
      material beam_stuff
      section = solid_section
```

```
      solid mechanics use model elastic
    end parameters for block block_1

  end finite element model slender_beam


# procedure data ------------------------------------------------------

  begin adagio procedure beam_procedure

    begin time control

      begin time stepping block
        start time = 0
        begin parameters for adagio region beam_region
          time increment = 0.2
        end parameters for adagio region beam_region
      end time stepping block

      termination time = 1.0

    end time control

    begin adagio region beam_region

      use finite element model slender_beam
      {Ifdef(four_node_tet)}
      begin user output
        extrapolate element variable stress to nodal variable nodal_stress
      end
      {Endif}
# BC data -------------------------------------------------------------

# traction load on right end
      begin traction
        surface = right_ss
        Node Set Subroutine = parabolic_Yshear
        Scale factor = 1.0
      end traction

      # traction reaction on left end
      begin traction
        surface = left_ss
        Node Set Subroutine = parabolic_Yshear
        scale factor = -1.0
      end traction

      # displacement bcs on left end to prevent rigid body motions
      # and prescribe fixity with respect to x
      begin fixed displacement
        surface = left_ss
        Component = X
      end fixed displacement
      begin fixed displacement
        node set = left_y_line
        Component = Y
      end fixed displacement
      begin fixed displacement
        node set = left_z_line
        Component = Z
      end fixed displacement


# Output data ---------------------------------------------------------

      begin results output beam_output
        database name = tet{topology}_beam{mesh}_Adagio.e
        database type = exodusII
```

```
        at step 0 increment = 2
        element variables = stress as elem_stress
        nodal variables = displacement as displ
        {Ifdef(four_node_tet)}
        nodal variables = nodal_stress as nodal_stress
        {Endif}
        {Ifdef(nodal_tet)}
        nodal variables = stress_1 as nodal_stress
        {Endif}
      end results output beam_output

# Adaptive time stepping-----------------------------------------------------------

# solver data ----------------------------------------------------

      begin solver

        begin cg
          target_relative_residual = 1e-3
          maximum iterations = 10000
          #The four node tet uses a tangent preconditioner for strongly_objective strain incrementation and ta
          #The four node tet uses a target residual = 1e-6 and no tangent preconditioner for node_based strain
          #The ten node tet uses a tangent preconditioner for fully_integrated formulation
          #The ten node tet uses a target residual = 1e-6,tangent preconditioner, linear solver, and internal
          {Ifdef(four_node_tet)}
          begin full tangent preconditioner
          end
          {Endif}
          {Ifdef(ten_node_tet)}
          begin full tangent preconditioner
            conditioning = no_check
          end
          {Endif}
          {Ifdef(composite_tet)}
          begin full tangent preconditioner
            linear solver = feti
            conditioning = no_check
          end
          {Endif}
          {Ifdef(nodal_tet)}
          preconditioner = elastic
          {Endif}
        end

      end solver

    end adagio region beam_region

  end adagio procedure beam_procedure
  {Ifdef(composite_tet)}
  begin feti equation solver feti
  end feti equation solver feti
  {Endif}
end sierra slender_beam_example
```

B.22. QUAD MEMBRANE PATCH TEST – SELECTIVE DEVIATORIC, MIDPOINT INCREMENT

See Section 3.9 for problem description.

```
begin sierra sd3dm4_patch_test

  begin function zero
    type is constant
    ordinate is unit
    abscissa is time
    begin values
      0.0
    end values
  end function zero

  begin function nodeset_2_u
    type is piecewise linear
    ordinate is displacement
    abscissa is time
    begin values
      0.0       0.0
      0.0001    3.69349159476781e-07
      0.0002    1.46830205113337e-06
      0.0003    3.26979885318058e-06
      0.0004    5.72948081030932e-06
      0.0005    8.78678249162321e-06
      0.0006    1.23664231100382e-05
      0.0007    1.63802601819784e-05
      0.0008    2.07294598841904e-05
      0.0009    2.53069306663805e-05
      0.001     2.99999601961531e-05
      0.0011    3.46929907060269e-05
      0.0012    3.92704644043871e-05
      0.0013    4.36196688870327e-05
      0.0014    4.76335124859597e-05
      0.0015    5.12131612171993e-05
      0.0016    5.4270472397411e-05
      0.0017    5.67301650056169e-05
      0.0018    5.8531673348656e-05
      0.0019    5.96306383870423e-05
      0.002     5.99999999998944e-05
      0.0025    6e-05
    end values
  end function nodeset_2_u

  begin function nodeset_2_v
    type is piecewise linear
    ordinate is displacement
    abscissa is time
    begin values
      0.0     0.0
      0.0001    7.38698318953561e-07
      0.0002    2.93660410226674e-06
      0.0003    6.53959770636115e-06
      0.0004    1.14589616206186e-05
      0.0005    1.75735649832464e-05
      0.0006    2.47328462200765e-05
      0.0007    3.27605203639567e-05
      0.0008    4.14589197683809e-05
      0.0009    5.06138613327609e-05
      0.001     5.99999203923062e-05
      0.0011    6.93859814120538e-05
      0.0012    7.85409288087742e-05
      0.0013    8.72393377740654e-05
      0.0014    9.52670249719195e-05
```

```
    0.0015    0.000102426322434399
    0.0016    0.000108540944794822
    0.0017    0.000113460330011234
    0.0018    0.000117063346697312
    0.0019    0.000119261276774085
    0.002     0.000119999999999789
    0.0025    0.00012
  end values
end function nodeset_2_v

begin function nodeset_4_u
  type is piecewise linear
  ordinate is displacement
  abscissa is time
  begin values
    0.0     0.0
    0.0001    1.47739663790712e-06
    0.0002    5.87320820453348e-06
    0.0003    1.30791954127223e-05
    0.0004    2.29179232412373e-05
    0.0005    3.51471299664929e-05
    0.0006    4.9465692440153e-05
    0.0007    6.55210407279135e-05
    0.0008    8.29178395367617e-05
    0.0009    0.000101227722665522
    0.001     0.000119999840784612
    0.0011    0.000138771962824108
    0.0012    0.000157081857617548
    0.0013    0.000174478675548131
    0.0014    0.000190534049943839
    0.0015    0.000204852644868797
    0.0016    0.000217081889589644
    0.0017    0.000226920660022467
    0.0018    0.000234126693394624
    0.0019    0.000238522553548169
    0.002     0.000239999999999578
    0.0025    0.00024
  end values
end function nodeset_4_u

begin function nodeset_4_v
  type is piecewise linear
  ordinate is displacement
  abscissa is time
  begin values
    0.0       0.0
    0.0001    7.38698318953561e-07
    0.0002    2.93660410226674e-06
    0.0003    6.53959770636115e-06
    0.0004    1.14589616206186e-05
    0.0005    1.75735649832464e-05
    0.0006    2.47328462200765e-05
    0.0007    3.27605203639567e-05
    0.0008    4.14589197683809e-05
    0.0009    5.06138613327609e-05
    0.001     5.99999203923062e-05
    0.0011    6.93859814120538e-05
    0.0012    7.85409288087742e-05
    0.0013    8.72393377740654e-05
    0.0014    9.52670249719195e-05
    0.0015    0.000102426322434399
    0.0016    0.000108540944794822
    0.0017    0.000113460330011234
    0.0018    0.000117063346697312
    0.0019    0.000119261276774085
    0.002     0.000119999999999789
    0.0025    0.00012
  end values
```

```
end function nodeset_4_v

begin function nodeset_5_u
  type is piecewise linear
  ordinate is displacement
  abscissa is time
  begin values
    0.0       0.0
    0.0001    1.8467457973839e-06
    0.0002    7.34151025566685e-06
    0.0003    1.63489942659029e-05
    0.0004    2.86474040515466e-05
    0.0005    4.39339124581161e-05
    0.0006    6.18321155501912e-05
    0.0007    8.19013009098919e-05
    0.0008    0.000103647299420952
    0.0009    0.000126534653331902
    0.001     0.000149999800980765
    0.0011    0.000173464953530134
    0.0012    0.000196352322021936
    0.0013    0.000218098344435163
    0.0014    0.000238167562429799
    0.0015    0.000256065806085997
    0.0016    0.000271352361987055
    0.0017    0.000283650825028084
    0.0018    0.00029265836674328
    0.0019    0.000298153191935212
    0.002     0.000299999999999472
    0.0025    0.0003
  end values
end function nodeset_5_u

begin function nodeset_5_v
  type is piecewise linear
  ordinate is displacement
  abscissa is time
  begin values
    0.0       0.0
    0.0001    1.47739663790712e-06
    0.0002    5.87320820453348e-06
    0.0003    1.30791954127223e-05
    0.0004    2.29179232412373e-05
    0.0005    3.51471299664929e-05
    0.0006    4.9465692440153e-05
    0.0007    6.55210407279135e-05
    0.0008    8.29178395367617e-05
    0.0009    0.000101227722665522
    0.001     0.000119999840784612
    0.0011    0.000138771962824108
    0.0012    0.000157081857617548
    0.0013    0.000174478675548131
    0.0014    0.000190534049943839
    0.0015    0.000204852644868797
    0.0016    0.000217081889589644
    0.0017    0.000226920660022467
    0.0018    0.000234126693394624
    0.0019    0.000238522553548169
    0.002     0.000239999999999578
    0.0025    0.00024
  end values
end function nodeset_5_v

define direction x with vector 1.0 0.0 0.0
define direction y with vector 0.0 1.0 0.0
define direction z with vector 0.0 0.0 1.0

begin material linear_elastic
  density         = 2.61e-4
```

```
  begin parameters for model elastic
    youngs modulus  = 1.e6
    poissons ratio  = 0.25
  end parameters for model elastic

end material linear_elastic

begin membrane section membrane_1
  thickness = 0.001
  formulation = selective_deviatoric
  deviatoric parameter = 0.2
end membrane section membrane_1

begin finite element model mesh1
  Database Name = sd3dm4_patch_test.g
  Database Type = exodusII

  begin parameters for block block_1
    material = linear_elastic
    model = elastic
    section = membrane_1
  end parameters for block block_1

end finite element model mesh1

begin presto procedure Apst_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region presto
        time step scale factor = 1.0
        time step increase factor = 2.0
        step interval = 100
      end parameters for presto region presto
    end time stepping block p1

    termination time = 0.0025
  end time control

  begin presto region presto

    use finite element model mesh1

    ### output description ###
    begin Results Output output_presto
      Database Name = sd3dm4_patch_test.e
      Database Type = exodusII
      At Time 0.0, Increment = 1.0e-4
      nodal Variables = force_external as f_ext
      nodal Variables = force_internal as f_int
      nodal Variables = displacement as displ
      nodal Variables = velocity as vel
      nodal Variables = acceleration as acc
      element Variables = memb_stress as memb_stress
      global Variables = timestep as timestep
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
    end results output output_presto

    ### definition of BCs ###

    begin prescribed displacement
      node set = nodelist_1
      direction = z
```

```
      function = zero
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_2
      direction = x
      function = nodeset_2_u
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_2
      direction = y
      function = nodeset_2_v
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_3
      direction = x
      function = zero
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_3
      direction = y
      function = zero
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_4
      direction = x
      function = nodeset_4_u
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_4
      direction = y
      function = nodeset_4_v
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_5
      direction = x
      function = nodeset_5_u
    end prescribed displacement

    begin prescribed displacement
      node set = nodelist_5
      direction = y
      function = nodeset_5_v
    end prescribed displacement

  end presto region presto
 end presto procedure Apst_Procedure
end sierra sd3dm4_patch_test
```

B.23. ELASTIC BEAM SECTION PROPERTY VERIFICATION IN AXIAL TENSION

See Section 3.10 for problem description.

```
begin sierra All_Beams_Axial_Quasi

  begin function ramp
    type is piecewise linear
    begin values
      0.0 0.0
      1.0 1.0
    end values
  end function ramp

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

#  E = {E = 30.0e3}
#  L = {L = 5.0}
#  l = {l = 5.0001}
#  dl = {dl = l - L}

  begin material aluminum
    density = 2.5880e-4
    begin parameters for model elastic_plastic
      youngs modulus = {E}
      poissons ratio = 0.3
      yield stress = 29.0e1
      hardening modulus = 0.0
    end parameters for model elastic_plastic
  end material aluminum

#   Area for Bar
#   D1_bar = {D1_bar = 0.1}
#   D2_bar = {D2_bar = 0.1}
#   Bar_Area = {Bar_Area = D1_bar * D2_bar}
#   Bar_Force = {Bar_Force = E * (dl/L) * Bar_Area}

  begin function bar_analytic_force
    type is analytic
    evaluate expression = " {Bar_Force} * x "
  end function bar_analytic_force

  begin beam section beam_1
    section = bar
    D1 = {D1_bar}
    D2 = {D2_bar}
    t axis = 0.0 1.0 0.0
  end beam section beam_1

#   Area for Box
#   D1_box = {D1_box = 0.1}
#   D2_box = {D2_box = 0.1}
#   D3_box = {D3_box = 0.0002}
#
Box_Area = {Box_Area = (D1_box * D2_box) - ((D1_box - (2 * D3_box)) * (D2_box - (2 * D3_box)))}
#   Box_Force = {Box_Force = E * (dl/L) * Box_Area}

  begin function box_analytic_force
    type is analytic
    evaluate expression = " {Box_Force} * x "
  end function box_analytic_force

  begin beam section beam_2
```

```
    section = box
    D1 = {D1_box}
    D2 = {D2_box}
    D3 = {D3_box}
    t axis = 0.0 1.0 0.0
  end beam section beam_2

#   Area for Rod (or Ellipse)
#   D1_rod = {D1_rod = 0.1}
#   D2_rod = {D2_rod = 0.1}
#   Rod_Area = {Rod_Area = (PI * D1_rod * D2_rod)/4}
#   Rod_Force = {Rod_Force = E * (dl/L) * Rod_Area}

  begin function rod_analytic_force
    type is analytic
    evaluate expression = " {Rod_Force} * x"
  end function rod_analytic_force

  begin beam section beam_3
    section = rod
    D1 = {D1_rod}
    D2 = {D2_rod}
    t axis = 0.0 1.0 0.0
  end beam section beam_3

#   Area for Tube (or Hallow Ellipse)
#   D1_tube = {D1_tube = 0.1}
#   D2_tube = {D2_tube = 0.1}
#   D3_tube = {D3_tube = 0.0002}
#   Tube_Area = {Tube_Area =
((PI * D1_tube * D2_tube)/4) - (PI * (D1_tube - (2 * D3_tube)) * (D2_tube - (2 * D3_tube))/4)}
#   Tube_Force = {Tube_Force = E * (dl/L) * Tube_Area}

  begin function tube_analytic_force
    type is analytic
    evaluate expression = " {Tube_Force} * x "
  end function tube_analytic_force

  begin beam section beam_4
    section = tube
    D1 = {D1_tube}
    D2 = {D2_tube}
    D3 = {D3_tube}
    t axis = 0.0 1.0 0.0
  end beam section beam_4

#   Area for C1
#   D1_c1 = {D1_c1 = 0.1002}
#   D2_c1 = {D2_c1 = 0.1004}
#   D3_c1 = {D3_c1 = 0.0002}
#   D4_c1 = {D4_c1 = 0.0002}
#   C1_Area = {C1_Area = (2 * D1_c1 * D3_c1) + (D4_c1 * (D2_c1 - (2 * D3_c1)))}
#   C1_Force = {C1_Force = E * (dl/L) * C1_Area}

  begin function c1_analytic_force
    type is analytic
    evaluate expression = " {C1_Force} * x "
  end function c1_Analytic_force

  begin beam section beam_5
    section = c1
    D1 = {D1_c1}
    D2 = {D2_c1}
    D3 = {D3_c1}
    D4 = {D4_c1}
    t axis = 0.0 1.0 0.0
  end beam section beam_5
```

```
#   Area for C2
#   D1_c2 = {D1_c2 = 0.1004}
#   D2_c2 = {D2_c2 = 0.1002}
#   D3_c2 = {D3_c2 = 0.0002}
#   D4_c2 = {D4_c2 = 0.0002}
#   C2_Area = {C2_Area = (2 * D2_c2 * D4_c2) + (D3_c2 * (D1_c2 - (2 * D4_c2)))}
#   C2_Force = {C2_Force = E * (dl/L) * C2_Area}

  begin function c2_analytic_force
    type is analytic
    evaluate expression = " {C2_Force} * x "
  end function c2_analytic_force

  begin beam section beam_6
    section = c2
    D1 = {D1_c2}
    D2 = {D2_c2}
    D3 = {D3_c2}
    D4 = {D4_c2}
    t axis = 0.0 1.0 0.0
  end beam section beam_6

#   Area for I
#   D1_I = {D1_I = 0.1}
#   D2_I = {D2_I = 0.1004}
#   D3_I = {D3_I = 0.0002}
#   D4_I = {D4_I = 0.0002}
#   I_Area = {I_Area = (2 * D1_I * D4_I) + (D3_I * (D2_I - (2 * D4_I)))}
#   I_Force = {I_Force = E * (dl/L) * I_Area}

  begin function I_analytic_force
    type is analytic
    evaluate expression = " {I_Force} * x "
  end function I_analytic_force

  begin beam section beam_7
    section = i
    D1 = {D1_I}
    D2 = {D2_I}
    D3 = {D3_I}
    D4 = {D4_I}
    t axis = 0.0 1.0 0.0
  end beam section beam_7

#   Area for I2
#   D1_I2 = {D1_I2 = 0.1}
#   D2_I2 = {D2_I2 = 0.10045}
#   D3_I2 = {D3_I2 = 0.0002}
#   D4_I2 = {D4_I2 = 0.0002}
#   D5_I2 = {D5_I2 = 0.08}
#   D6_I2 = {D6_I2 = 0.00025}
#
I2_Area = {I2_Area = (D1_I2 * D3_I2) + (D5_I2 * D6_I2) + (D4_I2 * (D2_I2 - D3_I2 - D6_I2))}
#   I2_Force = {I2_Force = E * (dl/L) * I2_Area}

  begin function I2_analytic_force
    type is analytic
    evaluate expression = " {I2_Force} * x "
  end function I2_analytic_force

  begin beam section beam_8
    section = i2
    D1 = {D1_I2}
    D2 = {D2_I2}
    D3 = {D3_I2}
    D4 = {D4_I2}
    D5 = {D5_I2}
    D6 = {D6_I2}
```

```
    t axis = 0.0 1.0 0.0
  end beam section beam_8

#   Area for T
#   D1_T = {D1_T = 0.1}
#   D2_T = {D2_T = 0.1002}
#   D3_T = {D3_T = 0.0002}
#   D4_T = {D4_T = 0.0002}
#   T_Area = {T_Area = (D1_T * D3_T) + (D4_T * (D2_T - D3_T))}
#   T_Force = {T_Force = E * (dl/L) * T_Area}

  begin function T_analytic_force
    type is analytic
    evaluate expression = " {T_Force} * x "
  end function T_analytic_force

  begin beam section beam_9
    section = t
    D1 = {D1_T}
    D2 = {D2_T}
    D3 = {D3_T}
    D4 = {D4_T}
    t axis = 0.0 1.0 0.0
  end beam section beam_9

#   Ar1ea for T1
#   D1_T1 = {D1_T1 = 0.1002}
#   D2_T1 = {D2_T1 = 0.1}
#   D3_T1 = {D3_T1 = 0.0002}
#   D4_T1 = {D4_T1 = 0.0002}
#   T1_Area = {T1_Area = (D2_T * D4_T) + (D3_T * (D1_T - D4_T))}
#   T1_Force = {T1_Force = E * (dl/L) * T1_Area}

  begin function T1_analytic_force
    type is analytic
    evaluate expression = " {T1_Force} * x "
  end function T1_analytic_force

  begin beam section beam_10
    section = t1
    D1 = {D1_T1}
    D2 = {D2_T1}
    D3 = {D3_T1}
    D4 = {D4_T1}
    t axis = 0.0 1.0 0.0
  end beam section beam_10

#   Area for Hat
#   D1_hat = {D1_hat = 0.3004}
#   D2_hat = {D2_hat = 0.1004}
#   D3_hat = {D3_hat = 0.1004}
#   D4_hat = {D4_hat = 0.0002}
#   Hat_Area = {Hat_Area = (D1_hat * D4_hat) + (2 * D4_hat * (D2_hat - D4_hat))}
#   Hat_Force = {Hat_Force = E * (dl/L) * Hat_Area}

  begin function hat_analytic_force
    type is analytic
    evaluate expression = " {Hat_Force} * x "
  end function hat_analytic_force

  begin beam section beam_11
    section = hat
    D1 = {D1_hat}
    D2 = {D2_hat}
    D3 = {D3_hat}
    D4 = {D4_hat}
    t axis = 0.0 1.0 0.0
  end beam section beam_11
```

```
#   Area for Z
#   D1_Z = {D1_Z = 0.2002}
#   D2_Z = {D2_Z = 0.1004}
#   D3_Z = {D3_Z = 0.0002}
#   D4_Z = {D4_Z = 0.0002}
#   Z_Area = {Z_Area = (D1_Z * D3_Z) + (D4_Z * (D2_Z - D3_Z))}
#   Z_Force = {Z_Force = E * (dl/L) * Z_Area}


  begin function z_analytic_force
    type is analytic
    evaluate expression = " {Z_Force} * x "
  end function z_analytic_force

  begin beam section beam_12
    section = z
    D1 = {D1_Z}
    D2 = {D2_Z}
    D3 = {D3_Z}
    D4 = {D4_Z}
    t axis = 0.0 1.0 0.0
  end beam section beam_12

#   Area for L
#   D1_L = {D1_L = 0.1}
#   D2_L = {D2_L = 0.1}
#   D3_L = {D3_L = 0.0002}
#   D4_L = {D4_L = 0.0002}
#   L_Area = {L_Area = (D1_L * D3_L) + (D4_L * (D2_L - D3_L))}
#   L_Force = {L_Force = E * (dl/L) * L_Area}

  begin function L_analytic_force
    type is analytic
    evaluate expression = " {L_Force} * x "
  end function L_analytic_force

  begin beam section beam_13
    section = l
    D1 = {D1_L}
    D2 = {D2_L}
    D3 = {D3_L}
    D4 = {D4_L}
    t axis = 0.0 1.0 0.0
  end beam section beam_13

  begin finite element model mesh1

    Database Name = axial_quasi.g
    Database Type = exodusII

    begin parameters for block block_1
      material = aluminum
      model = elastic_plastic
      section = beam_1
    end parameters for block block_1

    begin parameters for block block_2
      material = aluminum
      model = elastic_plastic
      section = beam_2
    end parameters for block block_2

    begin parameters for block block_3
      material = aluminum
      model = elastic_plastic
      section = beam_3
    end parameters for block block_3
```

```
  begin parameters for block block_4
    material = aluminum
    model = elastic_plastic
    section = beam_4
  end parameters for block block_4

  begin parameters for block block_5
    material = aluminum
    model = elastic_plastic
    section = beam_5
  end parameters for block block_5

  begin parameters for block block_6
    material = aluminum
    model = elastic_plastic
    section = beam_6
  end parameters for block block_6

  begin parameters for block block_7
    material = aluminum
    model = elastic_plastic
    section = beam_7
  end parameters for block block_7

  begin parameters for block block_8
    material = aluminum
    model = elastic_plastic
    section = beam_8
  end parameters for block block_8

  begin parameters for block block_9
    material = aluminum
    model = elastic_plastic
    section = beam_9
  end parameters for block block_9

  begin parameters for block block_10
    material = aluminum
    model = elastic_plastic
    section = beam_10
  end parameters for block block_10

  begin parameters for block block_11
    material = aluminum
    model = elastic_plastic
    section = beam_11
  end parameters for block block_11

  begin parameters for block block_12
    material = aluminum
    model = elastic_plastic
    section = beam_12
  end parameters for block block_12

  begin parameters for block block_13
    material = aluminum
    model = elastic_plastic
    section = beam_13
  end parameters for block block_13

end finite element model mesh1

begin adagio procedure adagio_procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
```

```
    begin parameters for adagio region adagio
      time increment = 0.04
    end parameters for adagio region adagio
  end time stepping block p1

  termination time = 1.0

end time control

begin adagio region adagio

  use finite element model mesh1

  ### output description ###
  begin Results Output output_adagio
    Database Name = axial_quasi.e
    Database Type = exodusII
    At Time 0.0, Increment = 0.04
    nodal Variables = force_internal as f_int
    nodal Variables = displacement as displ
    nodal variables = reaction as rxn
    element variables = cross_sectional_area as cs_area
    element variables = beam_strain_axial
    global variables = bar_error
    global variables = log_strain_xx
  end Results Output output_adagio

  begin solver
    begin cg
      maximum iterations = 5000
      target relative residual = 1.0e-5
    end
  end solver

  begin user output
    node set = nodelist_2
    compute global bar_frc_c as average of nodal reaction(x)
    compute global bar_frc_a as function bar_analytic_force
    compute global bar_error from expression "bar_frc_c > 0.0 ? (((bar_frc_a) - bar_frc_c) / bar_frc_c) *
    compute at every step
  end user output

  begin user output
    block = block_1
    compute global log_strain_xx as average of element beam_strain_axial
    compute at every step
  end user output

  begin user output
    node set = nodelist_3
    compute global box_frc_c as average of nodal reaction(x)
    compute global box_frc_a as function box_analytic_force
    compute global box_error from expression "box_frc_c > 0.0 ? (((box_frc_a) - box_frc_c) / box_frc_c) *
    compute at every step
  end user output

  begin user output
    node set = nodelist_4
    compute global rod_frc_c as average of nodal reaction(x)
    compute global rod_frc_a as function rod_analytic_force
    compute global rod_error from expression "rod_frc_c > 0.0 ? (((rod_frc_a) - rod_frc_c) / rod_frc_c) *
    compute at every step
  end

  begin user output
    node set = nodelist_5
    compute global tube_frc_c as average of nodal reaction(x)
    compute global tube_frc_a as function tube_analytic_force
```

```
  compute global tube_error from expression "tube_frc_c > 0.0 ? (((tube_frc_a) - tube_frc_c) / tube_frc_
  compute at every step
end

begin user output
  node set = nodelist_6
  compute global c1_frc_c as average of nodal reaction(x)
  compute global c1_frc_a as function c1_analytic_force
  compute global c1_error from expression "c1_frc_c > 0.0 ? (((c1_frc_a) - c1_frc_c) / c1_frc_c) * 100 :
  compute at every step
end

begin user output
  node set = nodelist_7
  compute global c2_frc_c as average of nodal reaction(x)
  compute global c2_frc_a as function c2_analytic_force
  compute global c2_error from expression "c2_frc_c > 0.0 ? (((c2_frc_a) - c2_frc_c) / c2_frc_c) * 100 :
  compute at every step
end

begin user output
  node set = nodelist_8
  compute global I_frc_c as average of nodal reaction(x)
  compute global I_frc_a as function I_analytic_force
  compute global I_error from expression "I_frc_c > 0.0 ? (((I_frc_a) - I_frc_c) / I_frc_c) * 100 : 0.0
  compute at every step
end

begin user output
  node set = nodelist_9
  compute global I2_frc_c as average of nodal reaction(x)
  compute global I2_frc_a as function I2_analytic_force
  compute global I2_error from expression "I2_frc_c > 0.0 ? (((I2_frc_a) - I2_frc_c) / I2_frc_c) * 100 :
  compute at every step
end

begin user output
  node set = nodelist_10
  compute global T_frc_c as average of nodal reaction(x)
  compute global T_frc_a as function T_analytic_force
  compute global T_error from expression "T_frc_c > 0.0 ? (((T_frc_a) - T_frc_c) / T_frc_c) * 100 : 0.0
  compute at every step
end

begin user output
  node set = nodelist_11
  compute global T1_frc_c as average of nodal reaction(x)
  compute global T1_frc_a as function T1_analytic_force
  compute global T1_error from expression "T1_frc_c > 0.0 ? (((T1_frc_a) - T1_frc_c) / T1_frc_c) * 100 :
  compute at every step
end

begin user output
  node set = nodelist_12
  compute global hat_frc_c as average of nodal reaction(x)
  compute global hat_frc_a as function hat_analytic_force
  compute global hat_error from expression "hat_frc_c > 0.0 ? (((hat_frc_a) - hat_frc_c) / hat_frc_c) *
  compute at every step
end

begin user output
  node set = nodelist_13
  compute global Z_frc_c as average of nodal reaction(x)
  compute global Z_frc_a as function Z_analytic_force
  compute global Z_error from expression "Z_frc_c > 0.0 ? (((Z_frc_a) - Z_frc_c) / Z_frc_c) * 100 : 0.0
  compute at every step
end
```

```
begin user output
  node set = nodelist_14
  compute global L_frc_c as average of nodal reaction(x)
  compute global L_frc_a as function L_analytic_force
  compute global L_error from expression "L_frc_c > 0.0 ? (((L_frc_a) - L_frc_c) / L_frc_c) * 100 : 0.0
  compute at every step
end

begin history output
  database name = axial_quasi.h
  database type = exodusII
  At Time 0.0, Increment = 0.04
  variable = global bar_frc_a
  variable = global bar_frc_c

  variable = global box_frc_a
  variable = global box_frc_c

  variable = global rod_frc_a
  variable = global rod_frc_c

  variable = global tube_frc_a
  variable = global tube_frc_c

  variable = global c1_frc_a
  variable = global c1_frc_c

  variable = global c2_frc_a
  variable = global c2_frc_c

  variable = global I_frc_a
  variable = global I_frc_c

  variable = global I2_frc_a
  variable = global I2_frc_c

  variable = global T_frc_a
  variable = global T_frc_c

  variable = global T1_frc_a
  variable = global T1_frc_c

  variable = global hat_frc_a
  variable = global hat_frc_c

  variable = global Z_frc_a
  variable = global Z_frc_c

  variable = global L_frc_a
  variable = global L_frc_c

  variable = global log_strain_xx
  variable = global bar_error
  variable = global box_error
  variable = global rod_error
  variable = global tube_error
  variable = global c1_error
  variable = global c2_error
  variable = global I_error
  variable = global I2_error
  variable = global T_error
  variable = global T1_error
  variable = global hat_error
  variable = global Z_error
  variable = global L_error
end

### definition of BCs ###
```

```
      begin fixed rotation
        include all blocks
        components = X Y Z
      end fixed rotation

      begin fixed displacement
        node set = nodelist_1
        components = Y Z
      end fixed displacement

      begin fixed displacement
        node set = nodelist_2 nodelist_3 nodelist_4 nodelist_5 nodelist_6 nodelist_7 nodelist_8 nodelist_9 noc
        components = X Y Z
      end fixed displacement

      begin prescribed displacement
        node set = nodelist_1
        component = X
        function = ramp
        scale factor = -{dl}
      end prescribed displacement

      begin solution verification
        verify global box_error = 0.0 plus or minus 0.03
        verify global rod_error = 0.0 plus or minus 0.03
        verify global tube_error = 0.0 plus or minus 0.03
        verify global c1_error = 0.0 plus or minus 0.03
        verify global c2_error = 0.0 plus or minus 0.03
        verify global I_error = 0.0 plus or minus 0.03
        verify global I2_error = 0.0 plus or minus 0.03
        verify global T_error = 0.0 plus or minus 0.03
        verify global T1_error = 0.0 plus or minus 0.03
        verify global hat_error = 0.0 plus or minus 0.03
        verify global Z_error = 0.0 plus or minus 0.03
        verify global L_error = 0.0 plus or minus 0.03
        completion file = verif1
      end

    end adagio region adagio
  end adagio procedure adagio_procedure
end sierra All_Beams_Axial_Quasi
```

B.24. ELASTIC BEAM BENDING VERIFICATION

See Section 3.11 for problem description.

```
begin sierra Beam_Problems

    begin function constant
      type is constant
      begin values
        1.0
      end values
    end function constant

    begin function constant_1
      type is constant
      begin values
        3.125e-2
      end values
    end function constant_1

    begin function cos_function
      type is analytic
      evaluate expression = " (1/64)*(1-cos(x*pi/3.2e-2)) "
    end function cos_function


  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0


  begin material aluminum
    density         = 2.5880e-4

# E = {E = 10.0e6}

    begin parameters for model elastic
      youngs modulus  = {E}
      poissons ratio  = 0.3
    end parameters for model elastic

  end material aluminum

begin finite element model beams

    Database Name = all_beams.g
    Database Type = exodusII

    begin parameters for block block_28 block_14 block_42
      material = aluminum
      model = elastic
      section = bar
    end parameters for block block_28 block_14 block_42

    begin parameters for block block_27 block_13 block_41
      material = aluminum
      model = elastic
      section = box
    end parameters for block block_27 block_13 block_41

    begin parameters for block block_26 block_12 block_40
      material = aluminum
      model = elastic
      section = rod
    end parameters for block block_26 block_12 block_40
```

```
  begin parameters for block block_25 block_11 block_39
    material = aluminum
    model = elastic
    section = tube
  end parameters for block block_25 block_11 block_39

  begin parameters for block block_24 block_10 block_38
    material = aluminum
    model = elastic
    section = c1
  end parameters for block block_24 block_10 block_38

  begin parameters for block block_23 block_9 block_37
    material = aluminum
    model = elastic
    section = c2
  end parameters for block block_23 block_9 block_37

  begin parameters for block block_22 block_8 block_36
    material = aluminum
    model = elastic
    section = I
  end parameters for block block_22 block_8 block_36

  begin parameters for block block_21 block_7 block_35
    material = aluminum
    model = elastic
    section = I2
  end parameters for block block_21 block_7 block_35

  begin parameters for block block_20 block_6 block_34
    material = aluminum
    model = elastic
    section = T
  end parameters for block block_20 block_6 block_34

  begin parameters for block block_19 block_5 block_33
    material = aluminum
    model = elastic
    section = T1
  end parameters for block block_19 block_5 block_33

  begin parameters for block block_18 block_4 block_32
    material = aluminum
    model = elastic
    section = HAT
  end parameters for block block_18 block_4 block_32

  begin parameters for block block_17 block_3 block_31
    material = aluminum
    model = elastic
    section = Z
  end parameters for block block_17 block_3 block_31

  begin parameters for block block_16 block_2 block_30
    material = aluminum
    model = elastic
    section = L
  end parameters for block block_16 block_2 block_30

 begin parameters for block block_15 block_1 block_29
    material = aluminum
    model = elastic
    section = ellipse
  end parameters for block block_15 block_1 block_29

end
```

```
# Ix and Iy for BAR
# D1_bar = {D1_bar = 0.1}
# D2_bar = {D2_bar = 0.1}
# Ix_bar = {Ix_bar = (1/12)*D1_bar*(D2_bar**(3))}
# Iy_bar = {Iy_bar = (1/12)*D2_bar*(D1_bar**(3))}
# Ip_bar = {Ip_bar = Ix_bar + Iy_bar}

  begin beam section bar
    section = BAR
    D1 = {D1_bar}
    D2 = {D2_bar}
    t axis = 0.0 1.0 0.0
  end beam section bar

# Ix and Iy for BOX
# D1_box = {D1_box = 0.1}
# D2_box = {D2_box = 0.1}
# D3_box = {D3_box = 0.0002}
# Ix_box = {Ix_box = (1/12)*D1_box*(D2_box**(3))-(1/12)*(D1_box-2*D3_box)*((D2_box-2*D3_box)**(3))}
# Iy_box = {Iy_box = (1/12)*D2_box*(D1_box**(3))-(1/12)*(D2_box-2*D3_box)*((D1_box-2*D3_box)**(3))}
# Ip_box = {Ip_box = Ix_box + Iy_box}

 begin beam section box
    section = BOX
    D1 = {D1_box}
    D2 = {D2_box}
    D3 = {D3_box}
    t axis = 0.0 1.0 0.0
  end beam section box

# Ix and Iy for ROD
# D1_ROD = {D1_ROD = 0.1}
# D2_ROD = {D2_ROD = 0.1}
# Ix_ROD = {Ix_ROD = (PI*(D1_ROD**(4)))/64}
# Iy_ROD = {Iy_ROD = (PI*(D1_ROD**(4)))/64}
# Ip_ROD = {Ip_ROD = Ix_ROD + Iy_ROD}


  begin beam section rod
    section = ROD
    D1 = {D1_ROD}
    D2 = {D2_ROD}
    t axis = 0.0 1.0 0.0
  end beam section rod

# Ix and Iy for TUBE
# D1_Tube = {D1_Tube = 0.1}
# D2_Tube = {D2_Tube = 0.1}
# D3_Tube = {D3_Tube = 0.0002}

#Ix_TUBE = {Ix_TUBE = PI*(((D1_Tube/2)-(D3_Tube/2))**(3))*D3_Tube}
#Iy_TUBE = {Iy_TUBE = PI*(((D1_Tube/2)-(D3_Tube/2))**(3))*D3_Tube}
#Ip_TUBE = {Ip_TUBE = Ix_TUBE + Iy_TUBE}


  begin beam section tube
    section = TUBE
    D1 = {D1_Tube}
    D2 = {D2_Tube}
    D3 = {D3_Tube}
    t axis = 0.0 1.0 0.0
  end beam section tube

# Ix and Iy for C1 beam
#D1_C1 = {D1_C1 = 0.1002}
#D2_C1 = {D2_C1 = 0.1004}
#D3_C1 = {D3_C1 = 0.0002}
#D4_C1 = {D4_C1 = 0.0002}
```

```
#b1_C1 = {b1_C1 = D1_C1}
#h1_C1 = {h1_C1 = D3_C1}
#b2_C1 = {b2_C1 = D4_C1}
#h2_C1 = {h2_C1 = D2_C1 -2*D3_C1}
#b3_C1 = {b3_C1 = D1_C1}
#h3_C1 = {h3_C1 = D3_C1}

#a1_C1 = {a1_C1 = b1_C1*h1_C1}
#a2_C1 = {a2_C1 = b2_C1*h2_C1}
#a3_C1 = {a3_C1 = b3_C1*h3_C1}
#at_C1 = {at_C1 = a1_C1 + a2_C1 + a3_C1}

#x1_C1 = {x1_C1 = D1_C1/2}
#x2_C1 = {x2_C1 = D4_C1/2}
#x3_C1 = {x3_C1 = D1_C1/2}

#y1_C1 = {y1_C1 = D3_C1/2}
#y2_C1 = {y2_C1 = D2_C1/2}
#y3_C1 = {y3_C1 = D2_C1 - D3_C1/2}

#xbar_C1 = {xbar_C1 = (a1_C1*x1_C1+a2_C1*x2_C1+a3_C1*x3_C1)/at_C1}
#ybar_C1 = {ybar_C1 = (a1_C1*y1_C1+a2_C1*y2_C1+a3_C1*y3_C1)/at_C1}

#Ix1_C1 = {Ix1_C1 = (1/12)*b1_C1*(h1_C1**(3))+b1_C1*h1_C1*((y1_C1-ybar_C1)**(2))}
#Ix2_C1 = {Ix2_C1 = (1/12)*b2_C1*(h2_C1**(3))+b2_C1*h2_C1*((y2_C1-ybar_C1)**(2))}
#Ix3_C1 = {Ix3_C1 = (1/12)*b3_C1*(h3_C1**(3))+b3_C1*h3_C1*((y3_C1-ybar_C1)**(2))}
#Ix_C1  = {Ix_C1 = Ix1_C1 + Ix2_C1 + Ix3_C1}

#Iy1_C1 = {Iy1_C1 = (1/12)*(b1_C1**(3))*h1_C1+b1_C1*h1_C1*((x1_C1-xbar_C1)**(2))}
#Iy2_C1 = {Iy2_C1 = (1/12)*(b2_C1**(3))*h2_C1+b2_C1*h2_C1*((x2_C1-xbar_C1)**(2))}
#Iy3_C1 = {Iy3_C1 = (1/12)*(b3_C1**(3))*h3_C1+b3_C1*h3_C1*((x3_C1-xbar_C1)**(2))}
#Iy_C1  = {Iy_C1 = Iy1_C1 + Iy2_C1 + Iy3_C1}
#Ip_C1 = {Ip_C1 = Ix_C1 + Iy_C1}


  begin beam section c1
    section = C1
    D1 = {D1_C1}
    D2 = {D2_C1}
    D3 = {D3_C1}
    D4 = {D4_C1}
    t axis = 0.0 1.0 0.0
  end beam section c1

# Ix and Iy for C2 beam
#D1_C2 = {D1_C2 = 0.1004}
#D2_C2 = {D2_C2 = 0.1002}
#D3_C2 = {D3_C2 = 0.0002}
#D4_C2 = {D4_C2 = 0.0002}

#b1_C2 = {b1_C2 = D4_C2}
#h1_C2 = {h1_C2 = D2_C2}
#b2_C2 = {b2_C2 = D1_C2-2*D4_C2}
#h2_C2 = {h2_C2 = D3_C2}
#b3_C2 = {b3_C2 = D4_C2}
#h3_C2 = {h3_C2 = D2_C2}

#a1_C2 = {a1_C2 = b1_C2 * h1_C2}
#a2_C2 = {a2_C2 = b2_C2 * h2_C2}
#a3_C2 = {a3_C2 = b3_C2 * h3_C2}
#at_C2 = {at_C2 = a1_C2 + a2_C2 + a3_C2}

#x1_C2 = {x1_C2 = D4_C2/2}
#x2_C2 = {x2_C2 = D1_C2/2}
#x3_C2 = {x3_C2 = D1_C2 - D4_C2/2}

#y1_C2 = {y1_C2 = D2_C2/2}
```

```
#y2_C2 = {y2_C2 = D3_C2/2}
#y3_C2 = {y3_C2 = D2_C2/2}

#xbar_C2 = {xbar_C2 = (a1_C2*x1_C2+a2_C2*x2_C2+a3_C2*x3_C2)/at_C2}
#ybar_C2 = {ybar_C2 = (a1_C2*y1_C2+a2_C2*y2_C2+a3_C2*y3_C2)/at_C2}

#Ix1_C2 = {Ix1_C2 = (1/12)*b1_C2*(h1_C2**(3))+b1_C2*h1_C2*((y1_C2-ybar_C2)**(2))}
#Ix2_C2 = {Ix2_C2 = (1/12)*b2_C2*(h2_C2**(3))+b2_C2*h2_C2*((y2_C2-ybar_C2)**(2))}
#Ix3_C2 = {Ix3_C2 = (1/12)*b3_C2*(h3_C2**(3))+b3_C2*h3_C2*((y3_C2-ybar_C2)**(2))}
#Ix_C2  = {Ix_C2 = Ix1_C2 + Ix2_C2 + Ix3_C2}

#Iy1_C2 = {Iy1_C2 = (1/12)*(b1_C2**(3))*h1_C2+b1_C2*h1_C2*((x1_C2-xbar_C2)**(2))}
#Iy2_C2 = {Iy2_C2 = (1/12)*(b2_C2**(3))*h2_C2+b2_C2*h2_C2*((x2_C2-xbar_C2)**(2))}
#Iy3_C2 = {Iy3_C2 = (1/12)*(b3_C2**(3))*h3_C2+b3_C2*h3_C2*((x3_C2-xbar_C2)**(2))}
#Iy_C2  = {Iy_C2 = Iy1_C2 + Iy2_C2 + Iy3_C2}
#Ip_C2 = {Ip_C2 = Ix_C2 + Iy_C2}


  begin beam section c2
    section = C2
    D1 = {D1_C2}
    D2 = {D2_C2}
    D3 = {D3_C2}
    D4 = {D4_C2}
    t axis = 0.0 1.0 0.0
  end beam section c2

# Ix and Iy for I beam
#D1_I = {D1_I = 0.1}
#D2_I = {D2_I = 0.1004}
#D3_I = {D3_I = 0.0002}
#D4_I = {D4_I = 0.0002}

#b1_I = {b1_I = D1_I}
#h1_I = {h1_I = D4_I}
#b2_I = {b2_I = D3_I}
#h2_I = {h2_I = D2_I - 2*D4_I}
#b3_I = {b3_I = D1_I}
#h3_I = {h3_I = D4_I}

#a1_I = {a1_I = b1_I * h1_I}
#a2_I = {a2_I = b2_I * h2_I}
#a3_I = {a3_I = b3_I * h3_I}
#at_I = {at_I = a1_I + a2_I + a3_I}

#x1_I = {x1_I = D1_I/2}
#x2_I = {x2_I = D1_I/2}
#x3_I = {x3_I = D1_I/2}

#y1_I = {y1_I = D4_I/2}
#y2_I = {y2_I = D2_I/2}
#y3_I = {y3_I = D2_I - D4_I/2}

#xbar_I = {xbar_I = (a1_I*x1_I+a2_I*x2_I+a3_I*x3_I)/at_I}
#ybar_I = {ybar_I = (a1_I*y1_I+a2_I*y2_I+a3_I*y3_I)/at_I}

#Ix1_I = {Ix1_I = (1/12)*b1_I*(h1_I**(3))+b1_I*h1_I*((y1_I-ybar_I)**(2))}
#Ix2_I = {Ix2_I = (1/12)*b2_I*(h2_I**(3))+b2_I*h2_I*((y2_I-ybar_I)**(2))}
#Ix3_I = {Ix3_I = (1/12)*b3_I*(h3_I**(3))+b3_I*h3_I*((y3_I-ybar_I)**(2))}
#Ix_I  = {Ix_I = Ix1_I + Ix2_I + Ix3_I}

#Iy1_I = {Iy1_I = (1/12)*(b1_I**(3))*h1_I+b1_I*h1_I*((x1_I-xbar_I)**(2))}
#Iy2_I = {Iy2_I = (1/12)*(b2_I**(3))*h2_I+b2_I*h2_I*((x2_I-xbar_I)**(2))}
#Iy3_I = {Iy3_I = (1/12)*(b3_I**(3))*h3_I+b3_I*h3_I*((x3_I-xbar_I)**(2))}
#Iy_I  = {Iy_I = Iy1_I + Iy2_I + Iy3_I}
#Ip_I =  {Ip_I = Ix_I + Iy_I}
```

```
  begin beam section I
    section = I
    D1 = {D1_I}
    D2 = {D2_I}
    D3 = {D3_I}
    D4 = {D4_I}
    t axis = 0.0 1.0 0.0
  end beam section I

# Ix and Iy for I2 beam
#D1_I2 = {D1_I2 = 0.1}
#D2_I2 = {D2_I2 = 0.10045}
#D3_I2 = {D3_I2 = 0.0002}
#D4_I2 = {D4_I2 = 0.0002}
#D5_I2 = {D5_I2 = 0.08}
#D6_I2 = {D6_I2 = 0.00025 }

#b1_I2 = {b1_I2 = D5_I2}
#h1_I2 = {h1_I2 = D6_I2}
#b2_I2 = {b2_I2 = D4_I2}
#h2_I2 = {h2_I2 = D2_I2 - D3_I2 - D6_I2}
#b3_I2 = {b3_I2 = D1_I2}
#h3_I2 = {h3_I2 = D3_I2}

#a1_I2 = {a1_I2 = b1_I2 * h1_I2}
#a2_I2 = {a2_I2 = b2_I2 * h2_I2}
#a3_I2 = {a3_I2 = b3_I2 * h3_I2}
#at_I2 = {at_I2 = a1_I2 + a2_I2 + a3_I2}

#x1_I2 = {x1_I2 = D1_I2/2}
#x2_I2 = {x2_I2 = D1_I2/2}
#x3_I2 = {x3_I2 = D1_I2/2}

#y1_I2 = {y1_I2 = D6_I2/2}
#y2_I2 = {y2_I2 = D6_I2 + (D2_I2-D6_I2-D3_I2)/2}
#y3_I2 = {y3_I2 = D2_I2 - D3_I2/2}

#xbar_I2 = {xbar_I2 = (a1_I2*x1_I2+a2_I2*x2_I2+a3_I2*x3_I2)/at_I2}
#ybar_I2 = {ybar_I2 = (a1_I2*y1_I2+a2_I2*y2_I2+a3_I2*y3_I2)/at_I2}

#Ix1_I2 = {Ix1_I2 = (1/12)*b1_I2*(h1_I2**(3))+b1_I2*h1_I2*((y1_I2-ybar_I2)**(2))}
#Ix2_I2 = {Ix2_I2 = (1/12)*b2_I2*(h2_I2**(3))+b2_I2*h2_I2*((y2_I2-ybar_I2)**(2))}
#Ix3_I2 = {Ix3_I2 = (1/12)*b3_I2*(h3_I2**(3))+b3_I2*h3_I2*((y3_I2-ybar_I2)**(2))}
#Ix_I2  = {Ix_I2 = Ix1_I2 + Ix2_I2 + Ix3_I2}

#Iy1_I2 = {Iy1_I2 = (1/12)*(b1_I2**(3))*h1_I2+b1_I2*h1_I2*((x1_I2-xbar_I2)**(2))}
#Iy2_I2 = {Iy2_I2 = (1/12)*(b2_I2**(3))*h2_I2+b2_I2*h2_I2*((x2_I2-xbar_I2)**(2))}
#Iy3_I2 = {Iy3_I2 = (1/12)*(b3_I2**(3))*h3_I2+b3_I2*h3_I2*((x3_I2-xbar_I2)**(2))}
#Iy_I2  = {Iy_I2 = Iy1_I2 + Iy2_I2 + Iy3_I2}
#Ip_I2 =  {Ip_I2 = Ix_I2 + Iy_I2}


  begin beam section I2
    section = I2
    D1 = {D1_I2}
    D2 = {D2_I2}
    D3 = {D3_I2}
    D4 = {D4_I2}
    D5 = {D5_I2}
    D6 = {D6_I2}
    t axis = 0.0 1.0 0.0
  end beam section I2

# Ix and Iy for T beam
#D1_T = {D1_T = 0.1}
#D2_T = {D2_T = 0.1002}
#D3_T = {D3_T = 0.0002}
#D4_T = {D4_T = 0.0002}
```

```
#b1_T = {b1_T = D4_T}
#h1_T = {h1_T = D2_T -D3_T}
#b2_T = {b2_T = D1_T}
#h2_T = {h2_T = D3_T}

#a1_T = {a1_T = b1_T * h1_T}
#a2_T = {a2_T = b2_T * h2_T}
#at_T = {at_T = a1_T + a2_T}

#x1_T = {x1_T = b2_T/2}
#x2_T = {x2_T = b2_T/2}

#y1_T = {y1_T = h1_T/2}
#y2_T = {y2_T = h1_T + h2_T/2}

#xbar_T = {xbar_T = (a1_T*x1_T+a2_T*x2_T)/at_T}
#ybar_T = {ybar_T = (a1_T*y1_T+a2_T*y2_T)/at_T}

#Ix1_T = {Ix1_T = (1/12)*b1_T*(h1_T**(3))+b1_T*h1_T*((y1_T-ybar_T)**(2))}
#Ix2_T = {Ix2_T = (1/12)*b2_T*(h2_T**(3))+b2_T*h2_T*((y2_T-ybar_T)**(2))}
#Ix_T  = {Ix_T = Ix1_T + Ix2_T}

#Iy1_T = {Iy1_T = (1/12)*(b1_T**(3))*h1_T+b1_T*h1_T*((x1_T-xbar_T)**(2))}
#Iy2_T = {Iy2_T = (1/12)*(b2_T**(3))*h2_T+b2_T*h2_T*((x2_T-xbar_T)**(2))}
#Iy_T  = {Iy_T = Iy1_T + Iy2_T}
#Ip_T  = {Ip_T = Ix_T + Iy_T}

  begin beam section T
    section = T
    D1 = {D1_T}
    D2 = {D2_T}
    D3 = {D3_T}
    D4 = {D4_T}
    t axis = 0.0 1.0 0.0
  end beam section T

# Ix and Iy for T1 beam

#D1_T1 = {D1_T1 = 0.1002}
#D2_T1 = {D2_T1 = 0.1}
#D3_T1 = {D3_T1 = 0.0002}
#D4_T1 = {D4_T1 = 0.0002}

#b1_T1 = {b1_T1 = D1_T1 - D4_T1}
#h1_T1 = {h1_T1 = D3_T1}
#b2_T1 = {b2_T1 = D4_T1}
#h2_T1 = {h2_T1 = D2_T1}

#a1_T1 = {a1_T1 = b1_T1* h1_T1}
#a2_T1 = {a2_T1 = b2_T1*h2_T1}
#at_T1 = {at_T1 = a1_T1+a2_T1}

#x1_T1 = {x1_T1 =  (D1_T1-D4_T1)/2}
#x2_T1 = {x2_T1 =  D1_T1 - (D4_T1/2)}

#y1_T1 = {y1_T1 = D2_T1/2}
#y2_T1 = {y2_T1 = D2_T1/2}

#xbar_T1 = {xbar_T1 = (a1_T1*x1_T1+a2_T1*x2_T1)/at_T1}
#ybar_T1 = {ybar_T1 = (a1_T1*y1_T1+a2_T1*y2_T1)/at_T1}

#Ix1_T1 = {Ix1_T1 = (1/12)*b1_T1*(h1_T1**(3))+b1_T1*h1_T1*((y1_T1-ybar_T1)**(2))}
#Ix2_T1 = {Ix2_T1 = (1/12)*b2_T1*(h2_T1**(3))+b2_T1*h2_T1*((y2_T1-ybar_T1)**(2))}
#Ix_T1  = {Ix_T1 = Ix1_T1 + Ix2_T1}

#Iy1_T1 = {Iy1_T1 = (1/12)*(b1_T1**(3))*h1_T1+b1_T1*h1_T1*((x1_T1-xbar_T1)**(2))}
#Iy2_T1 = {Iy2_T1 = (1/12)*(b2_T1**(3))*h2_T1+b2_T1*h2_T1*((x2_T1-xbar_T1)**(2))}
```

```
#Iy_T1  = {Iy_T1 = Iy1_T1 + Iy2_T1}
#Ip_T1  = {Ip_T1 = Ix_T1 + Iy_T1}

  begin beam section T1
    section = T1
    D1 = {D1_T1}
    D2 = {D2_T1}
    D3 = {D3_T1}
    D4 = {D4_T1}
    t axis = 0.0 1.0 0.0
  end beam section T1

# Ix and Iy for HAT beam
#D1_HAT = {D1_HAT = 0.3004}
#D2_HAT = {D2_HAT = 0.1004}
#D3_HAT = {D3_HAT = 0.1004}
#D4_HAT = {D4_HAT = 0.0002}

#b1_HAT = {b1_HAT = D3_HAT-D4_HAT}
#h1_HAT = {h1_HAT = D4_HAT}
#b2_HAT = {b2_HAT = D4_HAT}
#h2_HAT = {h2_HAT = D3_HAT-2*D4_HAT}
#b3_HAT = {b3_HAT = D3_HAT}
#h3_HAT = {h3_HAT = D4_HAT}
#b4_HAT = {b4_HAT = D4_HAT}
#h4_HAT = {h4_HAT = D3_HAT-2*D4_HAT}
#b5_HAT = {b5_HAT = D3_HAT-D4_HAT}
#h5_HAT = {h5_HAT = D4_HAT}

#a1_HAT = {a1_HAT = b1_HAT * h1_HAT}
#a2_HAT = {a2_HAT = b2_HAT * h2_HAT}
#a3_HAT = {a3_HAT = b3_HAT * h3_HAT}
#a4_HAT = {a4_HAT = b4_HAT * h4_HAT}
#a5_HAT = {a5_HAT = b5_HAT * h5_HAT}
#at_HAT = {at_HAT = a1_HAT + a2_HAT + a3_HAT + a4_HAT + a5_HAT}

#x1_HAT = {x1_HAT = (D3_HAT - D4_HAT)/2}
#x2_HAT = {x2_HAT = D3_HAT - D4_HAT - (D4_HAT/2)}
#x3_HAT = {x3_HAT = D1_HAT/2}
#x4_HAT = {x4_HAT = D1_HAT - b1_HAT + D4_HAT/2}
#x5_HAT = {x5_HAT = D1_HAT - b1_HAT/2}

#y1_HAT = {y1_HAT = D4_HAT/2}
#y2_HAT = {y2_HAT = D3_HAT/2}
#y3_HAT = {y3_HAT = D3_HAT - D4_HAT/2}
#y4_HAT = {y4_HAT = D3_HAT/2}
#y5_HAT = {y5_HAT = D4_HAT/2}

#xbar_HAT = {xbar_HAT =(a1_HAT*x1_HAT+a2_HAT*x2_HAT+a3_HAT*x3_HAT+a4_HAT*x4_HAT+a5_HAT*x5_HAT)/at_HAT}
#ybar_HAT = {ybar_HAT =(a1_HAT*y1_HAT+a2_HAT*y2_HAT+a3_HAT*y3_HAT+a4_HAT*y4_HAT+a5_HAT*y5_HAT)/at_HAT}

#Ix1_HAT = {Ix1_HAT = (1/12)*b1_HAT*(h1_HAT**(3))+b1_HAT*h1_HAT*((y1_HAT-ybar_HAT)**(2))}
#Ix2_HAT = {Ix2_HAT = (1/12)*b2_HAT*(h2_HAT**(3))+b2_HAT*h2_HAT*((y2_HAT-ybar_HAT)**(2))}
#Ix3_HAT = {Ix3_HAT = (1/12)*b3_HAT*(h3_HAT**(3))+b3_HAT*h3_HAT*((y3_HAT-ybar_HAT)**(2))}
#Ix4_HAT = {Ix4_HAT = (1/12)*b4_HAT*(h4_HAT**(3))+b4_HAT*h4_HAT*((y4_HAT-ybar_HAT)**(2))}
#Ix5_HAT = {Ix5_HAT = (1/12)*b5_HAT*(h5_HAT**(3))+b5_HAT*h5_HAT*((y5_HAT-ybar_HAT)**(2))}
#Ix_HAT  = {Ix_HAT = Ix1_HAT + Ix2_HAT + Ix3_HAT + Ix4_HAT + Ix5_HAT}

#Iy1_HAT = {Iy1_HAT = (1/12)*(b1_HAT**(3))*h1_HAT+b1_HAT*h1_HAT*((x1_HAT-xbar_HAT)**(2))}
#Iy2_HAT = {Iy2_HAT = (1/12)*(b2_HAT**(3))*h2_HAT+b2_HAT*h2_HAT*((x2_HAT-xbar_HAT)**(2))}
#Iy3_HAT = {Iy3_HAT = (1/12)*(b3_HAT**(3))*h3_HAT+b3_HAT*h3_HAT*((x3_HAT-xbar_HAT)**(2))}
#Iy4_HAT = {Iy4_HAT = (1/12)*(b4_HAT**(3))*h4_HAT+b4_HAT*h4_HAT*((x4_HAT-xbar_HAT)**(2))}
#Iy5_HAT = {Iy5_HAT = (1/12)*(b5_HAT**(3))*h5_HAT+b5_HAT*h5_HAT*((x5_HAT-xbar_HAT)**(2))}
#Iy_HAT  = {Iy_HAT = Iy1_HAT + Iy2_HAT + Iy3_HAT + Iy4_HAT + Iy5_HAT}
#Ip_HAT  = {Ip_HAT = Ix_HAT + Iy_HAT}

  begin beam section HAT
    section = HAT
```

```
    D1 = {D1_HAT}
    D2 = {D2_HAT}
    D3 = {D3_HAT}
    D4 = {D4_HAT}
    t axis = 0.0 1.0 0.0
  end beam section HAT

# Ix and Iy for Z beam
#D1_Z = {D1_Z = 0.2002}
#D2_Z = {D2_Z = 0.1004}
#D3_Z = {D3_Z = 0.0002}
#D4_Z = {D4_Z = 0.0002}

#b1_Z = {b1_Z = D2_Z - D3_Z}
#h1_Z = {h1_Z = D3_Z}
#b2_Z = {b2_Z = D4_Z}
#h2_Z = {h2_Z = D2_Z - 2*D3_Z}
#b3_Z = {b3_Z = D2_Z - D3_Z}
#h3_Z = {h3_Z = D3_Z}

#a1_Z = {a1_Z = b1_Z * h1_Z}
#a2_Z = {a2_Z = b2_Z * h2_Z}
#a3_Z = {a3_Z = b3_Z * h3_Z}
#at_Z = {at_Z = a1_Z + a2_Z + a3_Z}

#x1_Z = {x1_Z = b1_Z/2}
#x2_Z = {x2_Z = b1_Z - D4_Z/2}
#x3_Z = {x3_Z = D1_Z - b3_Z/2}

#y1_Z = {y1_Z = D2_Z - D3_Z/2}
#y2_Z = {y2_Z = D2_Z/2}
#y3_Z = {y3_Z = D3_Z/2}

#xbar_Z = {xbar_Z = (a1_Z*x1_Z+a2_Z*x2_Z+a3_Z*x3_Z)/at_Z}
#ybar_Z = {ybar_Z = (a1_Z*y1_Z+a2_Z*y2_Z+a3_Z*y3_Z)/at_Z}

#Ix1_Z = {Ix1_Z = (1/12)*b1_Z*(h1_Z**(3))+b1_Z*h1_Z*((y1_Z-ybar_Z)**(2))}
#Ix2_Z = {Ix2_Z = (1/12)*b2_Z*(h2_Z**(3))+b2_Z*h2_Z*((y2_Z-ybar_Z)**(2))}
#Ix3_Z = {Ix3_Z = (1/12)*b3_Z*(h3_Z**(3))+b3_Z*h3_Z*((y3_Z-ybar_Z)**(2))}
#Ix_Z  = {Ix_Z = Ix1_Z + Ix2_Z + Ix3_Z}

#Iy1_Z = {Iy1_Z = (1/12)*(b1_Z**(3))*h1_Z+b1_Z*h1_Z*((x1_Z-xbar_Z)**(2))}
#Iy2_Z = {Iy2_Z = (1/12)*(b2_Z**(3))*h2_Z+b2_Z*h2_Z*((x2_Z-xbar_Z)**(2))}
#Iy3_Z = {Iy3_Z = (1/12)*(b3_Z**(3))*h3_Z+b3_Z*h3_Z*((x3_Z-xbar_Z)**(2))}
#Iy_Z  = {Iy_Z = Iy1_Z + Iy2_Z + Iy3_Z}
#Ip_Z  = {Ip_Z = Ix_Z + Iy_Z}

  begin beam section Z
    section = Z
    D1 = {D1_Z}
    D2 = {D2_Z}
    D3 = {D3_Z}
    D4 = {D4_Z}
    t axis = 0.0 1.0 0.0
  end beam section Z

# Ix and Iy for L beam
#D1_L = {D1_L = 0.1}
#D2_L = {D2_L = 0.1}
#D3_L = {D3_L = 0.0002}
#D4_L = {D4_L = 0.0002}

#b1_L = {b1_L = D1_L-D4_L}
#h1_L = {h1_L= D3_L}
#b2_L = {b2_L = D4_L}
#h2_L = {h2_L = D2_L}

#a1_L = {a1_L = b1_L * h1_L}
```

```
#a2_L = {a2_L = b2_L * h2_L}
#at_L = {at_L = a1_L + a2_L}

#x1_L = {x1_L = b1_L/2 + b2_L}
#x2_L = {x2_L = b2_L/2}

#y1_L = {y1_L = h1_L/2}
#y2_L = {y2_L = h2_L/2}

#xbar_L = {xbar_L = (a1_L*x1_L+a2_L*x2_L)/at_L}
#ybar_L = {ybar_L = (a1_L*y1_L+a2_L*y2_L)/at_L}

#Ix1_L = {Ix1_L = (1/12)*b1_L*(h1_L**(3))+b1_L*h1_L*((y1_L-ybar_L)**(2))}
#Ix2_L = {Ix2_L = (1/12)*b2_L*(h2_L**(3))+b2_L*h2_L*((y2_L-ybar_L)**(2))}
#Ix_L  = {Ix_L = Ix1_L + Ix2_L}

#Iy1_L = {Iy1_L = (1/12)*(b1_L**(3))*h1_L+b1_L*h1_L*((x1_L-xbar_L)**(2))}
#Iy2_L = {Iy2_L = (1/12)*(b2_L**(3))*h2_L+b2_L*h2_L*((x2_L-xbar_L)**(2))}
#Iy_L  = {Iy_L = Iy1_L + Iy2_L}
#Ip_L  = {Ip_L = Ix_L + Iy_L}

  begin beam section L
    section = L
    D1 = {D1_L}
    D2 = {D2_L}
    D3 = {D3_L}
    D4 = {D4_L}
    t axis = 0.0 1.0 0.0
  end beam section L

# Ix and Iy for ROD (Ellipse) beam
#D1_E = {D1_E = 0.1}
#D2_E = {D2_E = 0.08}

#Ix_E = {Ix_E = (PI*(D1_E/2)*(D2_E/2)**3)/4}
#Iy_E = {Iy_E = (PI*(D2_E/2)*(D1_E/2)**3)/4}
#Ip_E = {Ip_E = Ix_E + Iy_E}

 begin beam section ellipse
    section = ROD
    D1 = {D1_E}
    D2 = {D2_E}
    t axis = 0.0 1.0 0.0
  end beam section ellipse

  begin presto procedure Apst_Procedure

    begin time control
        begin time stepping block p1
          start time = 0.0
          begin parameters for presto region presto
            time step scale factor = 1.0
            time step increase factor = 1.1
            step interval = 100
          end parameters for presto region presto
        end time stepping block p1

       begin time stepping block p2
          start time = 3.2e-2
          begin parameters for presto region presto
            time step scale factor = 1.0
            time step increase factor = 1.1
            step interval = 100
          end parameters for presto region presto
        end time stepping block p2

        termination time = 4.2e-2
```

```
    end time control

begin presto region presto

    use finite element model beams

    ### output description ###
    begin Results Output results
      Database Name = beamElasticVerif.e
      Database Type = exodusII
      At Time 0.0, Increment = 2.0e-5
      nodal Variables = moment_external as mext
      nodal Variables = force_external as fext
      nodal Variables = displacement as displacement
      nodal Variables = rotational_displacement as rdispl
    end results output results

# L = {L= 5}
# L1 = {L1 = L**3}
# G = {G = E/(2*(1.3))}
# ======= BAR ======
   Begin user output
      node set = nodelist_1001
      compute global F_bar_bending as average of nodal force_external(Y)
      compute global Disp_bar_bending as max absolute value of nodal displacement(Y)
      compute global bar_Ix from expression "((F_bar_bending*{L1})/(3*{E}*Disp_bar_bending))"
      compute global bar_bending_Ix_error from expression " abs(((bar_Ix - {Ix_bar})/{Ix_bar})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1002
      compute global F_bar_bending_Z as average of nodal force_external(Z)
      compute global Disp_bar_bending_Z as max absolute value of nodal displacement(Z)
      compute global bar_Iy from expression "((F_bar_bending_Z*({L1}))/(3*{E}*Disp_bar_bending_Z))"
      compute global bar_bending_Iy_error from expression " abs(((bar_Iy - {Iy_bar})/{Iy_bar})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1003
      compute global Moment_bar_torsion as average of nodal moment_external(X)
      compute global Disp_bar_theta as max absolute value of nodal rotational_displacement(X)
      compute global bar_Ip from expression "( (Moment_bar_torsion * {L})/({G}*Disp_bar_theta) )"
      compute global bar_torsion_Ip_error from expression " abs(((bar_Ip - {Ip_bar})/{Ip_bar})*100) "
      compute at every step
    End user output

# ===== BOX ======
  Begin user output
      node set = nodelist_2001
      compute global F_box_bending as average of nodal force_external(Y)
      compute global Disp_box_bending as max absolute value of nodal displacement(Y)
      compute global box_Ix from expression "((F_box_bending*({L1}))/(3*{E}*Disp_box_bending))"
      compute global box_bending_Ix_error from expression " abs(((box_Ix - {Ix_box})/{Ix_box})*100) "
      compute at every step
    End user output

  Begin user output
      node set = nodelist_2002
      compute global F_box_bending_Z as average of nodal force_external(Z)
      compute global Disp_box_bending_Z as max absolute value of nodal displacement(Z)
      compute global box_Iy from expression "((F_box_bending_Z*({L1}))/(3*{E}*Disp_box_bending_Z))"
      compute global box_bending_Iy_error from expression " abs(((box_Iy - {Iy_box})/{Iy_box})*100) "
      compute at every step
    End user output

  Begin user output
```

```
      node set = nodelist_2003
      compute global Moment_box_torsion as average of nodal moment_external(X)
      compute global Disp_box_theta as max absolute value of nodal rotational_displacement(X)
      compute global box_Ip from expression "( (Moment_box_torsion * {L})/({G}*Disp_box_theta) )"
      compute global box_torsion_Ip_error from expression " abs(((box_Ip - {Ip_box})/{Ip_box})*100) "
      compute at every step
    End user output

# ===== ROD ======
  Begin user output
      node set = nodelist_3001
      compute global F_rod_bending as average of nodal force_external(Y)
      compute global Disp_rod_bending as max absolute value of nodal displacement(Y)
      compute global rod_Ix from expression "((F_rod_bending*({L1}))/(3*{E}*Disp_rod_bending))"
      compute global rod_bending_Ix_error from expression " abs(((rod_Ix - {Ix_ROD})/{Ix_ROD})*100) "
      compute at every step
    End user output

  Begin user output
      node set = nodelist_3002
      compute global F_rod_bending_Z as average of nodal force_external(Z)
      compute global Disp_rod_bending_Z as max absolute value of nodal displacement(Z)
      compute global rod_Iy from expression "((F_rod_bending_Z*({L1}))/(3*{E}*Disp_rod_bending_Z))"
      compute global rod_bending_Iy_error from expression " abs(((rod_Iy - {Iy_ROD})/{Iy_ROD})*100) "
      compute at every step
    End user output

  Begin user output
      node set = nodelist_3003
      compute global Moment_rod_torsion as average of nodal moment_external(X)
      compute global Disp_rod_theta as max absolute value of nodal rotational_displacement(X)
      compute global rod_Ip from expression "( (Moment_rod_torsion * {L})/({G}*Disp_rod_theta) )"
      compute global rod_torsion_Ip_error from expression " abs(((rod_Ip - {Ip_ROD})/{Ip_ROD})*100) "
      compute at every step
    End user output

# ===== TUBE ======
  Begin user output
      node set = nodelist_4001
      compute global F_tube_bending as average of nodal force_external(Y)
      compute global Disp_tube_bending as max absolute value of nodal displacement(Y)
      compute global tube_Ix from expression "((F_tube_bending*({L1}))/(3*{E}*Disp_tube_bending))"
      compute global tube_bending_Ix_error from expression " abs(((tube_Ix - {Ix_TUBE})/{Ix_TUBE})*100) "
      compute at every step
    End user output

  Begin user output
      node set = nodelist_4002
      compute global F_tube_bending_Z as average of nodal force_external(Z)
      compute global Disp_tube_bending_Z as max absolute value of nodal displacement(Z)
      compute global tube_Iy from expression "((F_tube_bending_Z*({L1}))/(3*{E}*Disp_tube_bending_Z))"
      compute global tube_bending_Iy_error from expression " abs(((tube_Iy - {Iy_TUBE})/{Iy_TUBE})*100) "
      compute at every step
    End user output

  Begin user output
      node set = nodelist_4003
      compute global Moment_TUBE_torsion as average of nodal moment_external(X)
      compute global Disp_TUBE_theta as max absolute value of nodal rotational_displacement(X)
      compute global TUBE_Ip from expression "( (Moment_TUBE_torsion * {L})/({G}*Disp_TUBE_theta) )"
      compute global TUBE_torsion_Ip_error from expression " abs(((TUBE_Ip - {Ip_TUBE})/{Ip_TUBE})*100) "
      compute at every step
    End user output

# ===== C1 =======
    Begin user output
      node set = nodelist_5001
      compute global F_c1_bending as average of nodal force_external(Y)
```

```
      compute global Disp_c1_bending as max absolute value of nodal displacement(Y)
      compute global c1_Ix from expression "((F_c1_bending*({L1}))/(3*{E}*Disp_c1_bending))"
      compute global c1_bending_Ix_error from expression " abs(((c1_Ix - {Ix_C1})/{Ix_C1})*100) "
      compute at every step
    End user output

  Begin user output
      node set = nodelist_5002
      compute global F_c1_bending_Z as average of nodal force_external(Z)
      compute global Disp_c1_bending_Z as max absolute value of nodal displacement(Z)
      compute global c1_Iy from expression "((F_c1_bending_Z*({L1}))/(3*{E}*Disp_c1_bending_Z))"
      compute global c1_bending_Iy_error from expression " abs(((c1_Iy - {Iy_C1})/{Iy_C1})*100) "
      compute at every step
    End user output

  Begin user output
      node set = nodelist_5003
      compute global Moment_c1_torsion as average of nodal moment_external(X)
      compute global Disp_c1_theta as max absolute value of nodal rotational_displacement(X)
      compute global c1_Ip from expression "( (Moment_c1_torsion * {L})/({G}*Disp_c1_theta) )"
      compute global c1_torsion_Ip_error from expression " abs(((c1_Ip - {Ip_C1})/{Ip_C1})*100) "
      compute at every step
    End user output

# ===== C2 =======
    Begin user output
      node set = nodelist_6001
      compute global F_C2_bending as average of nodal force_external(Y)
      compute global Disp_C2_bending as max absolute value of nodal displacement(Y)
      compute global C2_Ix from expression "((F_C2_bending*({L1}))/(3*{E}*Disp_C2_bending))"
      compute global C2_bending_Ix_error from expression " abs(((C2_Ix - {Ix_C2})/{Ix_C2})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_6002
      compute global F_c2_bending_Z as average of nodal force_external(Z)
      compute global Disp_c2_bending_Z as max absolute value of nodal displacement(Z)
      compute global c2_Iy from expression "((F_c2_bending_Z*({L1}))/(3*{E}*Disp_c2_bending_Z))"
      compute global c2_bending_Iy_error from expression " abs(((c2_Iy - {Iy_C2})/{Iy_C2})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_6003
      compute global Moment_c2_torsion as average of nodal moment_external(X)
      compute global Disp_c2_theta as max absolute value of nodal rotational_displacement(X)
      compute global c2_Ip from expression "( (Moment_c2_torsion * {L})/({G}*Disp_c2_theta) )"
      compute global c2_torsion_Ip_error from expression " abs(((c2_Ip - {Ip_C2})/{Ip_C2})*100) "
      compute at every step
    End user output

# ===== I =======
    Begin user output
      node set = nodelist_7001
      compute global F_I_bending as average of nodal force_external(Y)
      compute global Disp_I_bending as max absolute value of nodal displacement(Y)
      compute global I_Ix from expression "((F_I_bending*({L1}))/(3*{E}*Disp_I_bending))"
      compute global I_bending_Ix_error from expression " abs(((I_Ix - {Ix_I})/{Ix_I})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_7002
      compute global F_I_bending_Z as average of nodal force_external(Z)
      compute global Disp_I_bending_Z as max absolute value of nodal displacement(Z)
      compute global I_Iy from expression "((F_I_bending_Z*({L1}))/(3*{E}*Disp_I_bending_Z))"
      compute global I_bending_Iy_error from expression " abs(((I_Iy - {Iy_I})/{Iy_I})*100) "
```

```
      compute at every step
    End user output

    Begin user output
      node set = nodelist_7003
      compute global Moment_I_torsion as average of nodal moment_external(X)
      compute global Disp_I_theta as max absolute value of nodal rotational_displacement(X)
      compute global I_Ip from expression "( (Moment_I_torsion * {L})/({G}*Disp_I_theta) )"
      compute global I_torsion_Ip_error from expression " abs(((I_Ip - {Ip_I})/{Ip_I})*100) "
      compute at every step
    End user output

# ===== I2 =======
    Begin user output
      node set = nodelist_8001
      compute global F_I2_bending as average of nodal force_external(Y)
      compute global Disp_I2_bending as max absolute value of nodal displacement(Y)
      compute global I2_Ix from expression "((F_I2_bending*({L1}))/(3*{E}*Disp_I2_bending))"
      compute global I2_bending_Ix_error from expression " abs(((I2_Ix - {Ix_I2})/{Ix_I2})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_8002
      compute global F_I2_bending_Z as average of nodal force_external(Z)
      compute global Disp_I2_bending_Z as max absolute value of nodal displacement(Z)
      compute global I2_Iy from expression "((F_I2_bending_Z*({L1}))/(3*{E}*Disp_I2_bending_Z))"
      compute global I2_bending_Iy_error from expression " abs(((I2_Iy - {Iy_I2})/{Iy_I2})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_8003
      compute global Moment_I2_torsion as average of nodal moment_external(X)
      compute global Disp_I2_theta as max absolute value of nodal rotational_displacement(X)
      compute global I2_Ip from expression "( (Moment_I2_torsion * {L})/({G}*Disp_I2_theta) )"
      compute global I2_torsion_Ip_error from expression " abs(((I2_Ip - {Ip_I2})/{Ip_I2})*100) "
      compute at every step
    End user output

# ===== T =======
    Begin user output
      node set = nodelist_9001
      compute global F_T_bending as average of nodal force_external(Y)
      compute global Disp_T_bending as max absolute value of nodal displacement(Y)
      compute global T_Ix from expression "((F_T_bending*({L1}))/(3*{E}*Disp_T_bending))"
      compute global T_bending_Ix_error from expression " abs(((T_Ix - {Ix_T})/{Ix_T})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_9002
      compute global F_T_bending_Z as average of nodal force_external(Z)
      compute global Disp_T_bending_Z as max absolute value of nodal displacement(Z)
      compute global T_Iy from expression "((F_T_bending_Z*({L1}))/(3*{E}*Disp_T_bending_Z))"
      compute global T_bending_Iy_error from expression " abs(((T_Iy - {Iy_T})/{Iy_T})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_9003
      compute global Moment_T_torsion as average of nodal moment_external(X)
      compute global Disp_T_theta as max absolute value of nodal rotational_displacement(X)
      compute global T_Ip from expression "( (Moment_T_torsion * {L})/({G}*Disp_T_theta) )"
      compute global T_torsion_Ip_error from expression " abs(((T_Ip - {Ip_T})/{Ip_T})*100) "
      compute at every step
    End user output
```

```
# ===== T1 ======
    Begin user output
      node set = nodelist_100001
      compute global F_T1_bending as average of nodal force_external(Y)
      compute global Disp_T1_bending as max absolute value of nodal displacement(Y)
      compute global T1_Ix from expression "((F_T1_bending*({L1}))/(3*{E}*Disp_T1_bending))"
      compute global T1_bending_Ix_error from expression " abs(((T1_Ix - {Ix_T1})/{Ix_T1})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_100002
      compute global F_T1_bending_Z as average of nodal force_external(Z)
      compute global Disp_T1_bending_Z as max absolute value of nodal displacement(Z)
      compute global T1_Iy from expression "((F_T1_bending_Z*({L1}))/(3*{E}*Disp_T1_bending_Z))"
      compute global T1_bending_Iy_error from expression " abs(((T1_Iy - {Iy_T1})/{Iy_T1})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_100003
      compute global Moment_T1_torsion as average of nodal moment_external(X)
      compute global Disp_T1_theta as max absolute value of nodal rotational_displacement(X)
      compute global T1_Ip from expression "( (Moment_T1_torsion * {L})/({G}*Disp_T1_theta) )"
      compute global T1_torsion_Ip_error from expression " abs(((T1_Ip - {Ip_T1})/{Ip_T1})*100) "
      compute at every step
    End user output

# ===== HAT =====
    Begin user output
      node set = nodelist_1101
      compute global F_HAT_bending as average of nodal force_external(Y)
      compute global Disp_HAT_bending as max absolute value of nodal displacement(Y)
      compute global HAT_Ix from expression "((F_HAT_bending*({L1}))/(3*{E}*Disp_HAT_bending))"
      compute global HAT_bending_Ix_error from expression " abs(((HAT_Ix - {Ix_HAT})/{Ix_HAT})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1102
      compute global F_HAT_bending_Z as average of nodal force_external(Z)
      compute global Disp_HAT_bending_Z as max absolute value of nodal displacement(Z)
      compute global HAT_Iy from expression "((F_HAT_bending_Z*({L1}))/(3*{E}*Disp_HAT_bending_Z))"
      compute global HAT_bending_Iy_error from expression " abs(((HAT_Iy - {Iy_HAT})/{Iy_HAT})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1103
      compute global Moment_HAT_torsion as average of nodal moment_external(X)
      compute global Disp_HAT_theta as max absolute value of nodal rotational_displacement(X)
      compute global HAT_Ip from expression "( (Moment_HAT_torsion * {L})/({G}*Disp_HAT_theta) )"
      compute global HAT_torsion_Ip_error from expression " abs(((HAT_Ip - {Ip_HAT})/{Ip_HAT})*100) "
      compute at every step
    End user output

# ===== Z =======
    Begin user output
      node set = nodelist_1201
      compute global F_Z_bending as average of nodal force_external(Y)
      compute global Disp_Z_bending as max absolute value of nodal displacement(Y)
      compute global Z_Ix from expression "((F_Z_bending*({L1}))/(3*{E}*Disp_Z_bending))"
      compute global Z_bending_Ix_error from expression " abs(((Z_Ix - {Ix_Z})/{Ix_Z})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1202
```

```
      compute global F_Z_bending_Z as average of nodal force_external(Z)
      compute global Disp_Z_bending_Z as max absolute value of nodal displacement(Z)
      compute global Z_Iy from expression "((F_Z_bending_Z*({L1}))/(3*{E}*Disp_Z_bending_Z))"
      compute global Z_bending_Iy_error from expression " abs(((Z_Iy - {Iy_Z})/{Iy_Z})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1203
      compute global Moment_Z_torsion as average of nodal moment_external(X)
      compute global Disp_Z_theta as max absolute value of nodal rotational_displacement(X)
      compute global Z_Ip from expression "( (Moment_Z_torsion * {L})/({G}*Disp_Z_theta) )"
      compute global Z_torsion_Ip_error from expression " abs(((Z_Ip - {Ip_Z})/{Ip_Z})*100) "
      compute at every step
    End user output

# ====== L ======
    Begin user output
      node set = nodelist_1301
      compute global F_L_bending as average of nodal force_external(Y)
      compute global Disp_L_bending as max absolute value of nodal displacement(Y)
      compute global L_Ix from expression "((F_L_bending*({L1}))/(3*{E}*Disp_L_bending))"
      compute global L_bending_Ix_error from expression " abs(((L_Ix - {Ix_L})/{Ix_L})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1302
      compute global F_L_bending_Z as average of nodal force_external(Z)
      compute global Disp_L_bending_Z as max absolute value of nodal displacement(Z)
      compute global L_Iy from expression "((F_L_bending_Z*({L1}))/(3*{E}*Disp_L_bending_Z))"
      compute global L_bending_Iy_error from expression " abs(((L_Iy - {Iy_L})/{Iy_L})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1303
      compute global Moment_L_torsion as average of nodal moment_external(X)
      compute global Disp_L_theta as max absolute value of nodal rotational_displacement(X)
      compute global L_Ip from expression "( (Moment_L_torsion * {L})/({G}*Disp_L_theta) )"
      compute global L_torsion_Ip_error from expression " abs(((L_Ip - {Ip_L})/{Ip_L})*100) "
      compute at every step
    End user output

# ===== Ellipse =====
    Begin user output
      node set = nodelist_1401
      compute global F_E_bending as average of nodal force_external(Y)
      compute global Disp_E_bending as max absolute value of nodal displacement(Y)
      compute global E_Ix from expression "((F_E_bending*({L1}))/(3*{E}*Disp_E_bending))"
      compute global E_bending_Ix_error from expression " abs(((E_Ix - {Ix_E})/{Ix_E})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1402
      compute global F_E_bending_Z as average of nodal force_external(Z)
      compute global Disp_E_bending_Z as max absolute value of nodal displacement(Z)
      compute global E_Iy from expression "((F_E_bending_Z*({L1}))/(3*{E}*Disp_E_bending_Z))"
      compute global E_bending_Iy_error from expression " abs(((E_Iy - {Iy_E})/{Iy_E})*100) "
      compute at every step
    End user output

    Begin user output
      node set = nodelist_1403
      compute global Moment_E_torsion as average of nodal moment_external(X)
      compute global Disp_E_theta as max absolute value of nodal rotational_displacement(X)
      compute global E_Ip from expression "( (Moment_E_torsion * {L})/({G}*Disp_E_theta) )"
```

```
  compute global E_torsion_Ip_error from expression " abs(((E_Ip - {Ip_E})/{Ip_E})*100) "
  compute at every step
End user output

begin history output
  database name = beamElasticVerif.h
  database type = exodusII
  At Time 0.0, Increment = 2.0e-5
  variable =  global F_bar_bending
  variable =  global Disp_bar_bending
  variable =  global bar_Ix
  variable =  global bar_bending_Ix_error
  variable =  global F_bar_bending_Z
  variable =  global Disp_bar_bending_Z
  variable =  global bar_Iy
  variable =  global bar_bending_Iy_error
  variable =  global Moment_bar_torsion
  variable =  global Disp_bar_theta
  variable =  global bar_Ip
  variable =  global bar_torsion_Ip_error

  variable =  global F_box_bending
  variable =  global Disp_box_bending
  variable =  global box_Ix
  variable =  global box_bending_Ix_error
  variable =  global F_box_bending_Z
  variable =  global Disp_box_bending_Z
  variable =  global box_Iy
  variable =  global box_bending_Iy_error
  variable =  global Moment_box_torsion
  variable =  global Disp_box_theta
  variable =  global box_Ip
  variable =  global box_torsion_Ip_error

  variable =  global F_rod_bending
  variable =  global Disp_rod_bending
  variable =  global rod_Ix
  variable =  global rod_bending_Ix_error
  variable =  global F_rod_bending_Z
  variable =  global Disp_rod_bending_Z
  variable =  global rod_Iy
  variable =  global rod_bending_Iy_error
  variable =  global Moment_rod_torsion
  variable =  global Disp_rod_theta
  variable =  global rod_Ip
  variable =  global rod_torsion_Ip_error

  variable =  global F_tube_bending
  variable =  global Disp_tube_bending
  variable =  global tube_Ix
  variable =  global tube_bending_Ix_error
  variable =  global F_tube_bending_Z
  variable =  global Disp_tube_bending_Z
  variable =  global tube_Iy
  variable =  global tube_bending_Iy_error
  variable =  global Moment_tube_torsion
  variable =  global Disp_tube_theta
  variable =  global tube_Ip
  variable =  global tube_torsion_Ip_error

  variable =  global F_c1_bending
  variable =  global Disp_c1_bending
  variable =  global c1_Ix
  variable =  global c1_bending_Ix_error
  variable =  global F_c1_bending_Z
  variable =  global Disp_c1_bending_Z
  variable =  global c1_Iy
  variable =  global c1_bending_Iy_error
```

```
variable =  global Moment_c1_torsion
variable =  global Disp_c1_theta
variable =  global c1_Ip
variable =  global c1_torsion_Ip_error

variable =  global F_C2_bending
variable =  global Disp_C2_bending
variable =  global C2_Ix
variable =  global C2_bending_Ix_error
variable =  global F_C2_bending_Z
variable =  global Disp_C2_bending_Z
variable =  global C2_Iy
variable =  global C2_bending_Iy_error
variable =  global Moment_C2_torsion
variable =  global Disp_C2_theta
variable =  global C2_Ip
variable =  global C2_torsion_Ip_error

variable =  global F_I_bending
variable =  global Disp_I_bending
variable =  global I_Ix
variable =  global I_bending_Ix_error
variable =  global F_I_bending_Z
variable =  global Disp_I_bending_Z
variable =  global I_Iy
variable =  global I_bending_Iy_error
variable =  global Moment_I_torsion
variable =  global Disp_I_theta
variable =  global I_Ip
variable =  global I_torsion_Ip_error

variable =  global F_I2_bending
variable =  global Disp_I2_bending
variable =  global I2_Ix
variable =  global I2_bending_Ix_error
variable =  global F_I2_bending_Z
variable =  global Disp_I2_bending_Z
variable =  global I2_Iy
variable =  global I2_bending_Iy_error
variable =  global Moment_I2_torsion
variable =  global Disp_I2_theta
variable =  global I2_Ip
variable =  global I2_torsion_Ip_error

variable =  global F_T_bending
variable =  global Disp_T_bending
variable =  global T_Ix
variable =  global bar_T_Ix_error
variable =  global F_T_bending_Z
variable =  global Disp_T_bending_Z
variable =  global T_Iy
variable =  global T_bending_Iy_error
variable =  global Moment_T_torsion
variable =  global Disp_T_theta
variable =  global T_Ip
variable =  global T_torsion_Ip_error

variable =  global F_T1_bending
variable =  global Disp_T1_bending
variable =  global T1_Ix
variable =  global T1_bending_Ix_error
variable =  global F_T1_bending_Z
variable =  global Disp_T1_bending_Z
variable =  global T1_Iy
variable =  global T1_bending_Iy_error
variable =  global Moment_T1_torsion
variable =  global Disp_T1_theta
variable =  global T1_Ip
```

```
    variable =  global T1_torsion_Ip_error

    variable =  global F_HAT_bending
    variable =  global Disp_HAT_bending
    variable =  global HAT_Ix
    variable =  global HAT_bending_Ix_error
    variable =  global F_HAT_bending_Z
    variable =  global Disp_HAT_bending_Z
    variable =  global HAT_Iy
    variable =  global HAT_bending_Iy_error
    variable =  global Moment_HAT_torsion
    variable =  global Disp_HAT_theta
    variable =  global HAT_Ip
    variable =  global HAT_torsion_Ip_error

    variable =  global F_Z_bending
    variable =  global Disp_Z_bending
    variable =  global Z_Ix
    variable =  global Z_bending_Ix_error
    variable =  global F_Z_bending_Z
    variable =  global Disp_Z_bending_Z
    variable =  global Z_Iy
    variable =  global Z_bending_Iy_error
    variable =  global Moment_Z_torsion
    variable =  global Disp_Z_theta
    variable =  global Z_Ip
    variable =  global Z_torsion_Ip_error

    variable =  global F_L_bending
    variable =  global Disp_L_bending
    variable =  global L_Ix
    variable =  global L_bending_Ix_error
    variable =  global F_L_bending_Z
    variable =  global Disp_L_bending_Z
    variable =  global L_Iy
    variable =  global L_bending_Iy_error
    variable =  global Moment_L_torsion
    variable =  global Disp_L_theta
    variable =  global L_Ip
    variable =  global L_torsion_Ip_error

    variable =  global F_E_bending
    variable =  global Disp_E_bending
    variable =  global E_Ix
    variable =  global E_bending_Ix_error
    variable =  global F_E_bending_Z
    variable =  global Disp_E_bending_Z
    variable =  global E_Iy
    variable =  global E_bending_Iy_error
    variable =  global Moment_E_torsion
    variable =  global Disp_E_theta
    variable =  global E_Ip
    variable =  global E_torsion_Ip_error
  end

### Solution Verifcation###

  Begin Solution Verification
    Skip Times = 0.0 to 3.199e-2
    completion file = elastic_verif_bending
    verify global bar_bending_Ix_error  = 0.6 plus or minus 0.2
    verify global bar_bending_Iy_error  = 0.6 plus or minus 0.2

    verify global box_bending_Ix_error  = 8.0 plus or minus 2.0
    verify global box_bending_Iy_error  = 8.0 plus or minus 2.0

    verify global rod_bending_Ix_error  = 0.7 plus or minus 0.2
    verify global rod_bending_Iy_error  = 0.7 plus or minus 0.2
```

```
      verify global tube_bending_Ix_error = 2.0 plus or minus 1.0
      verify global tube_bending_Iy_error = 2.0 plus or minus 1.0

      verify global c1_bending_Ix_error   = 1.0 plus or minus 0.5
      verify global c1_bending_Iy_error   = 2.0 plus or minus 1.0

      verify global C2_bending_Ix_error   = 2.0 plus or minus 1.0
      verify global C2_bending_Iy_error   = 1.0 plus or minus 0.5

      verify global I_bending_Ix_error    = 0.02 plus or minus 0.02
      verify global I_bending_Iy_error    = 0.7  plus or minus 0.2

      verify global I2_bending_Ix_error   = 1.0 plus or minus 0.5
      verify global I2_bending_Iy_error   = 5.0 plus or minus 2.0

      verify global T_bending_Ix_error    = 1.5 plus or minus 0.5
      verify global T_bending_Iy_error    = 4.0 plus or minus 2.0

      verify global T1_bending_Ix_error   = 3.0 plus or minus 2.0
      verify global T1_bending_Iy_error   = 1.0 plus or minus 0.5

      verify global HAT_bending_Ix_error  = 2.0 plus or minus 1.0
      verify global HAT_bending_Iy_error  = 0.2 plus or minus 0.1

      verify global Z_bending_Ix_error    = 1.0 plus or minus 0.5
      verify global Z_bending_Iy_error    = 1.0 plus or minus 0.5

      verify global L_bending_Ix_error    = 1.5 plus or minus 1.5
      verify global L_bending_Iy_error    = 1.5 plus or minus 1.5

      verify global E_bending_Ix_error    = 0.7 plus or minus 0.2
      verify global E_bending_Iy_error    = 0.7 plus or minus 0.2

    End Solution Verification

Begin Solution Verification
      Skip Times = 0.0 to 4.199e-2
      completion file = elastic_verif_torsion
      verify global bar_torsion_Ip_error  = 15  plus or minus  5
      verify global box_torsion_Ip_error  = 10  plus or minus  5
      verify global rod_torsion_Ip_error  = 1.0 plus or minus  1.0
      verify global tube_torsion_Ip_error = 20  plus or minus  5
      verify global c1_torsion_Ip_error   = 35  plus or minus  5
      verify global C2_torsion_Ip_error   = 35  plus or minus  5
      verify global I_torsion_Ip_error    = 35  plus or minus  5
      verify global I2_torsion_Ip_error   = 30  plus or minus  5
      verify global T_torsion_Ip_error    = 80  plus or minus  5
      verify global T1_torsion_Ip_error   = 80  plus or minus  5
      verify global HAT_torsion_Ip_error  = 80  plus or minus  15
      verify global Z_torsion_Ip_error    = 60  plus or minus  5
      verify global L_torsion_Ip_error    = 70  plus or minus  5
      verify global E_torsion_Ip_error    = 5   plus or minus  3
End Solution Verification

    Begin Viscous Damping
      Include all blocks
      Mass Damping coefficient = 1000.0
      Stiffness Damping Coefficient = 0.0
    End Viscous Damping

    begin fixed displacement
      node set = nodelist_1
      components = X Y Z
    end fixed displacement

    begin fixed displacement
      node set = nodelist_12011
```

```
      components = Z
    end fixed displacement

    begin fixed displacement
      node set = nodelist_12022
      components = Y
    end fixed displacement

   begin fixed displacement
      node set = nodelist_13011
      components = Z
    end fixed displacement

    begin fixed displacement
      node set = nodelist_13022
      components = Y
    end fixed displacement

    begin fixed rotation
      node set = nodelist_1
      components = X Y Z
    end fixed rotation

    begin prescribed force
      node set = nodelist_2
      direction = y
      function = constant
      scale factor = 0.0005
    end prescribed force

   begin prescribed force
      node set = nodelist_3
      direction = z
      function = constant
      scale factor = 0.0005
    end prescribed force

    begin prescribed moment
      node set = nodelist_4
      direction = x
      function = cos_function
      scale factor = 0.1
      active periods = p1
    end prescribed moment

    begin prescribed moment
      node set = nodelist_4
      direction = x
      function = constant_1
      scale factor = 0.1
      active periods = p2
    end prescribed moment

     end presto region presto
  end presto procedure Apst_Procedure
end sierra Beam_Problems
```

B.25. ELASTIC AND PLASTIC BEAM SECTION VERIFICATION

See Section 3.12 for problem description.

```
#
#  Test elastic and plastic properties of beam sections
#
begin sierra beamPropertyTest



  begin function ramp
    type is piecewise linear
    begin values
      0.0 0.0
      1.0 1.0
    end
  end


  # Material properties
  # {ym = 1.0e+6}
  # {y0 = 1.0}


  begin material mat1
    density = .0002588
    begin parameters for model elastic_plastic
      poissons ratio = 0.0
      youngs modulus = {ym}
      yield stress = {y0}
      hardening modulus = 0.0
    end
  end


  define direction x with vector 1 0 0
  define direction y with vector 0 1 0
  define direction z with vector 0 0 1

#
#
#  Hat Section:
#     Note, thickness is d4 for all sections
#
#
#                          d3
#                    +-----------------+ -------------------+-
#                    |       r2        |                    |
#                    +--+-----------+--+                    |
#          d4        |  |           |  |                    |
#           |        |r1|         ->|r3|<-d4               d2
#           v        |  |           |  |                    |
#       +------------+--+           +--+------------+       |
#       |      r0       |           |       r4      |       |
#       +---------------+           +---------------+ -----+-
#           ^
#       |   |                                       |
#  t    +-------------------d1----------------------+
#  ^
#  |
#  |
#  +--->s
#
```

```
  begin beam section hat
    section = HAT
    d1 = 0.40
    d2 = 0.10
    d3 = 0.15
    d4 = 0.01
    t axis = 0 1 0
    #Visualize Integration Points = on
  end
#
#  BAR, Rectangluar Section:
#
#             d1
#      +--------------+
#      |              |
#      |              |
#      |              |d2
#      |              |
#  t   |              |
#  ^   +--------------+
#  |
#  |
#  +--->s
#
  begin beam section bar
    section = BAR
    d1 = 0.1
    d2 = 0.2
    t axis = 0 1 0
    #Visualize Integration Points = on
  end
#
#  Box, Hollow Rectangluar Section:
#
#             d1
#      +--------------+
#      | +----------+ |
#      | |        ->| |<-d3
#      | |          | |
#      | |          | |d2
#      | |          | |
#      | |          | |
#  t   | +----------+ |
#  ^   +--------------+
#  |
#  |
#  +--->s
#
  begin beam section box
    section = BOX
    d1 = 0.10
    d2 = 0.20
    d3 = 0.01
    t axis = 0 1 0
    #Visualize Integration Points = on
  end
#
#  ROD, elliptical Section:
#
#          _______ ___________
#        /       \           |
#       /         \          |
#      /           \      d2
#     |\           /|       |
#     | \         / |       |
#     |  \_______/__|_____|_
#  t  |             |
#  ^  +-----d1------+
```

```
#  |  |            |
#  |
#  +--->s
#
  begin beam section rod
    section = ROD
    d1 = 0.20
    d2 = 0.20
    t axis = 0 1 0
    #Visualize Integration Points = on
  end
#
#  I Section:
#
#      +-------------+ ------+-
#      |             |       |
#      +-------------+       |
#         ->|  |<-d3         |
#           |  |             |
#           |  |             d2
#           |  |             |
#           |  |             |
#  t   +-------------+       |
#  ^   |             |d4     |
#  |   +-------------+ ------+-
#  |         d1
#  +--->s
#
  begin beam section I
    section = I
    d1 = 0.15
    d2 = 0.20
    d3 = 0.01
    d4 = 0.015
    t axis = 0 1 0
  end



#
#  Analytic Results:
#    Problem consists of four beams for each section type.
The beams are organized in a matrix with
#    numbering scheme
#      block_A00 :  Beams of section type A.
#      nodelist_ABC : Nodelist on beam section type A, loading condition B, and beam end C
#
#
The four beams of a section type are subjected to pure axial extension, pure T direction bending, pure
#   S direction bending, and pure R direction torsion.
#
#
The analytic results are computed with the following mathematica code (betwen the #ifdef and #endif lines.
#   In order to run this mathematica code:
#     1) run the 'MATERIAL MODEL SETUP BLOCK'
#     2) Run ONE of the section defintion blocks
#     3) Run the 'RESPONSE EQUATIONS BLOCK'
#     4) Run the 'DATA GATHERING BLOCK'
#     5) run the  'DATA PRINT BLOCK'.
he data print block will output the analytic force response functions which are found
#        found below
#
#{ifdef(COMMENT_OUT)}

(*=============================================================================*)
(* MATERIAL MODEL SETUP BLOCK *)
Clear[d1, d2, d3, d5, height, width, y0, ym, stress, x, y, strain, e0];
(* Stress Strain Law.  Elastic Perfectly Plastic.
```

```
  ym      = youngs modulus
  y0      = yield stress
  gm      = shear modulus
  g0      = shear yield strength
  beamLen = the physical length of the beams in the mesh *)
(*
Note, here assuming poissons ratio 0.0 which makes the shear modulus one half of youngs modulus *)
beamLen = 2.0;
y0 = 1.0;
ym = 1000000;
g0 = y0/2.0;
gm = ym/2.0;
stressRR[strain_] =  If[strain*ym < y0, If[strain*ym > -y0, strain*ym, -y0], y0];
stressST[strain_] =  If[strain*gm < g0, If[strain*gm > -g0, strain*gm, -g0], g0];
(*=============================================================================*)

(*=============================================================================*)
(* Specific Parameters for I section *)
Clear[x, y, IStencil, DataI];
SectName = "I";
d1 = 0.15;
d2 = 0.2;
d3 = 0.01;
d4 = 0.015;
height = d2;
width = d1;
(*  Use open section analytic J, which is given by sum of section sub lengths B
times t^3 / 3.  *)
SumB1 = d1 + d1;
SumB2 = d2 - 2*d4;
J = SumB1 * (d4^3) / 3.0 + SumB2 * (d3^3)/3.0;
(*  Use open section analytic Tmax, T = g0 * B * t^2 /2  *)
Tmax = SumB1 * g0 * (d4^2)/2.0 + SumB2*(d3^2)/2.0;
(* I stencil.  Returns 1 if X/Y is in the section, Returns 0 if X/Y is not in the section *)

IStencil[x_, y_] = If[y <= d4  || y >= (d2 - d4), 1,
   If[x > d1/2.0 - d3/2.0 &&  x < d1/2.0 + d3/2.0, 1, 0]];

(* Plot the section to make sure it the stencil is right *)
DataI = {};
For[x = 0.0, x <= width, x += width/200,
  For[y = 0.0, y < height, y += height/200,
    If[IStencil[x, y] > 0, DataI = Append[DataI, {x, y}]]
    ];
  ];
ListPlot[DataI]
SectionStencil = IStencil;
(*=============================================================================*)

(*=============================================================================*)
(* Specific Parameters for Hat section *)
Clear[x, y, HatStencil, DataHat];
SectName = "Hat";
d1 = 0.4;
d2 = 0.1;
d3 = 0.15;
d4 = 0.01;
height = d2;
width = d1;
(*
Engineering Approximation: Use open section analytic J, which is given by sum of section sub lengths B times t
*)
SumB = (2*(d1/2.0 - d3/2.0 + d4/2.0) + 2*(d2 - d4) + (d3 - d4));
J = SumB * (d4^3) / 3.0;
(*  Use open section analytic Tmax, T = g0 * B * t^2 /2  *)
Tmax = SumB * g0 * (d4^2)/2.0;
(* Hat stencil.  Returns 1 if X/Y is in the section, Returns 0 if X/Y is not in the section *)
HatStencil[x_, y_] =
```

```
  If[y <= d4 && (x <= (d1/2.0 - d3/2.0 + d4) ||
      x >= (d1/2.0 + d3/2.0 - d4)), 1,
   If[y >= d4 &&
     y <= d2 -
       d4 && (x >= d1/2.0 - d3/2.0 && x <= (d1/2.0 - d3/2.0) + d4 ||
       x >= d1/2.0 + d3/2.0 - d4 && x <= d1/2.0 + d3/2.0), 1,
    If[y >=
       d2 - d4 && (x >= d1/2.0 - d3/2.0 && x <= d1/2.0 + d3/2.0), 1,
     0]]];
(* Plot the section to make sure it the stencil is right *)
DataHat = {};
For[x = 0.0, x <= width, x += width/200,
  For[y = 0.0, y < height, y += height/200,
    If[HatStencil[x, y] > 0, DataHat = Append[DataHat, {x, y}]]
    ];
  ];
ListPlot[DataHat]
SectionStencil = HatStencil;
(*============================================================================*)

(*============================================================================*)
(* Specific Parameters for Bar section *)
Clear[x, y, BarStencil, DataBar];
SectName = "Bar";
d1 = 0.10;
d2 = 0.20;
height = d2;
width = d1;
(* Engineering Approximation: compute J and Tmax based of of the warping correction factors for a 2 by 1 recta
J = 0.229*d2*d1^3;
Tmax = ((g0*d2*d1^2)/4.07)*1.694;
(* Bar stencil.  Returns 1 if X/Y is in the section, Returns 0 if X/Y is not in the section *)
BarStencil[x_, y_] = 1.0;
DataBar = {};
For[x = 0.0, x <= width, x += width/200,
  For[y = 0.0, y < height, y += height/200,
    If[BarStencil[x, y] > 0, DataBar = Append[DataBar, {x, y}]]
    ];
  ];
ListPlot[DataBar]
SectionStencil = BarStencil;
(*============================================================================*)

(*============================================================================*)
(* Specific Parameters for Rod section *)

Clear[x, y, RodStencil, DataRod];
SectName = "Rod";
d1 = 0.20;
d2 = 0.20;
height = d2;
width = d1;
(* Engineering Approximation: For a circular section under torsion plane sections actually
will remain plane (no warping) analytic solution exists for J and max torision *)
J = Pi * (d1^4)/32;
Tmax = ((2*Pi*(d1/2)^3)/3)*g0;
(* Rod stencil.  Returns 1 if X/Y is in the section, Returns 0 if X/Y is not in the section *)
RodStencil[x_, y_] =
  If[(x - d1/2)^2 + (y - d1/2)^2 < (d1/2)^2, 1, 0];
DataRod = {};
For[x = 0.0, x <= width, x += width/100,
  For[y = 0.0, y < height, y += height/100,
    If[RodStencil[x, y] > 0, DataRod = Append[DataRod, {x, y}]]
    ];
  ];
ListPlot[DataRod]
SectionStencil = RodStencil;
(*============================================================================*)
```

```
(*=============================================================================*)
(* Box Section Parameters *)
Clear[x, y, BoxStencil, DataBox];
SectName = "Box";
d1 = 0.10;
d2 = 0.20;
d3 = 0.01;
height = d2;
width = d1;
(*  Use closed section analytic J, which is given by J = 4A^2/(B/t).  *)
(*  Engineering Approximation:  Compute J useing closed seciton approximation.
A is the area enclosed by the section midplane, and B is the total section length at the midplane *)
A = (d1 - d3)*(d2 - d3);
B = 2*(d1 - d3) + 2*(d2 - d3);
J = (4*A^2)/(B/d3);
(*  Use closed section analytic T = 2*g0*t*A *)
Tmax = 2*g0*d3*A;
(* Box stencil.  Returns 1 if X/Y is in the section, Returns 0 if X/Y is not in the section *)
BoxStencil[x_, y_] =  If[x > d3 && x < d1 - d3 && y > d3 && y < d2 - d3, 0, 1];
DataBox = {};
For[x = 0.0, x <= width, x += width/200,
  For[y = 0.0, y < height, y += height/200,
    If[BoxStencil[x, y] > 0, DataBox = Append[DataBox, {x, y}]]
    ];
  ];
ListPlot[DataBox]
SectionStencil = BoxStencil;
(*=============================================================================*)

(*=============================================================================*)
(*  RESPONSE EQUATIONS BLOCK *)
(* Define equation for force and moment produced by section for a given incremental rotation or extension.
*)
Clear[Mtt, Mss, Frr, Mrr];
(*  X moment in a beam for incremental rotation dthet about x at end of beam *)
Mtt[dthet_] := Module[
   {eps, Force, MomentTot, ForceTot, x, y, yc},
   Clear[eps, Force, MomentTot, ForceTot, x, y, yc];
   If[dthet == 0.0, Return[0]];
   (* eps is the strain at a given X/Y point.  dthet known,
location of neutral axis, yc is unknown*)
   eps[x_, y_, yc_] = (dthet)*(y - yc);
   (* Stress at a given point in a section.  If point is not in the stencil it is zero.
If point is in section the stress is
      given by the stress stain law evaluated at the given strain *)
   Force[x_, y_, yc_] = stressRR[eps[x, y, yc]]*SectionStencil[x, y];
   (*  Binary search to find neutral axis.
Find the yc at which the integral of force over the section is zero. *)
   yp0 = 0.0;
   yp1 = height;
   vy0 = Re[ NIntegrate[Force[x, y, yp0], {x, 0, width}, {y, 0, height},
AccuracyGoal -> 8]];
   vy1 = Re[ NIntegrate[Force[x, y, yp1], {x, 0, width}, {y, 0, height},
AccuracyGoal -> 8]];
   For[i = 0, i < 20, ++i,
      ypm = (yp0 + yp1)/2.0;
      vym =  Re[NIntegrate[Force[x, y, ypm], {x, 0, width}, {y, 0, height},
AccuracyGoal -> 8]];
      If[(vy0 <= 0 && vym >= 0)  || (vy0 >= 0 && vym <= 0),
       yp1 = ypm;
       vy1 = vym;
       Continue[];
      ];
      If[(vym <= 0 && vy1 >= 0)  || (vym >= 0 && vy1 <= 0),
       yp0 = ypm;
       vy0 = vym;
       Continue[];
```

```
      ];
    ];
   yc = (yp0 + yp1)/2.0;
   (* Return the actual moment assocaited with the solved for neutral axis*)
   MomentTot =  NIntegrate[Force[x, y, yc]*y, {x, 0, width}, {y, 0, height}];
   Re[MomentTot]
   ];
(*  Y moment in a beam for incremental rotation dthet about y at end of beam *)
Mss[dthet_] := Module[
   {eps, Force, MomentTot, ForceTotx, x, y, xc},
   Clear[eps, Force, MomentTot, ForceTot, x, y, xc];
   If[dthet == 0.0, Return[0]];
   (* eps is the strain at a given X/Y point.  dthet known,
location of neutral axis, xc is unknown*)
   eps[x_, y_, xc_] = (dthet)*(x - xc);
   (* Stress at a given point in a section.  If point is not in the stencil it is zero.
If point is in section the stress is
      given by the stress stain law evaluated at the given strain *)
   Force[x_, y_, xc_] =  stressRR[eps[x, y, xc]]*SectionStencil[x, y];
   (*  Binary search to find neutral axis.
Find the xc at which the integral of force over the section is zero. *)
   xp0 = 0.0;
   xp1 = width;
   vx0 = Re[NIntegrate[Force[x, y, xp0], {x, 0, width}, {y, 0, height}, AccuracyGoal -> 8]];
   vx1 = Re[NIntegrate[Force[x, y, xp1], {x, 0, width}, {y, 0, height}, AccuracyGoal -> 8]];
   For[i = 0, i < 20, ++i,
      xpm = (xp0 + xp1)/2.0;
      vxm = Re[NIntegrate[Force[x, y, xpm], {x, 0, width}, {y, 0, height},
AccuracyGoal -> 8]];
      If[(vx0 <= 0 && vxm >= 0)  || (vx0 >= 0 && vxm <= 0),
       xp1 = xpm;
       vx1 = vxm;
       Continue[];
      ];
      If[(vxm <= 0 && vx1 >= 0)  || (vxm >= 0 && vx1 <= 0),
       xp0 = xpm;
       vx0 = vxm;
       Continue[];
      ];
    ];
   xc = (xp0 + xp1)/2.0;
   (* Return the actual moment assocaited with the solved for neutral axis*)
   MomentTot =  NIntegrate[Force[x, y, xc]*x, {x, 0, width}, {y, 0, height}];
   Re[MomentTot]
   ];

(* Define equation of force produced by section for a given incremental extension dL/length.
This is for the extension in the RR (length) direction *)
Frr[dL_] := Module[
   {eps, Force, ForceTot, x, y},
   Clear[eps, Force, ForceTot, x, y]
   If[dL == 0.0, Return[0]];
   (* eps is the strain at a given X/Y point.
This is just the uniform value based on the apply dL*)
   eps[x_, y_] = dL;
   (* Stress at a given point in a section.  If point is not in the stencil it is zero.
If point is in section the stress is
      given by the stress stain law evaluated at the given strain.
Integrate stress over section to compute total resistance force *)
   Force[x_, y_] = stressRR[eps[x, y]]*SectionStencil[x, y];
   ForceTot = NIntegrate[Force[x, y], {x, 0, width}, {y, 0, height}];
   ForceTot
   ];
(*  R moment in a beam for incremental rotation dthet about r at end of beam.
This is torsional resistance.  This equations set assumes the beam does not warp in plane *)
MrrNoWarping[dthet_] := Module[
   {eps, Force, ForceTot, x, y, xc, yc, MomentRR, MomentTot,
dx, dy, dtot, Fxy, Forcex, Forcey},
```

```
   Clear[eps, Force, ForceTot, x, y, xc, yc, MomentRR,
MomentTot, dx, dy, dtot, Fxy, Forcex, Forcey, yp0, yp1, vy0,  vy1];
   If[dthet == 0.0, Return[0]];
(*  Compute strains at an XY point.  Location of neutral axis (xc, yc) is unknown. *)
   dx[x_, y_, xc_, yc_] = dthet*(yc - y);
   dy[x_, y_, xc_, yc_] = dthet*(x - xc);
   dtot[x_, y_, xc_, yc_] = Sqrt[dx[x, y, xc, yc]^2 + dy[x, y, xc, yc]^2];
(*
compute shear stress at a given point (Fxy) and force per unit area at a given point (Forcex and Forcey) *)
   Fxy[x_, y_, xc_, yc_] =  stressST[dtot[x, y, xc, yc]]*SectionStencil[x, y];
   Forcex[x_, y_, xc_, yc_] =  Fxy [x, y, xc, yc]* (dx[x, y, xc, yc]/dtot[x, y, xc, yc]);
   Forcey[x_, y_, xc_, yc_] =  Fxy [x, y, xc, yc]* (dy[x, y, xc, yc]/dtot[x, y, xc, yc]);
(*  Binary search for location of neutral axis.
The neutral axis is the (xc, yc) point in the beam at which the sum of both the Forcex and Forcey over the sec
(*
Note all current sections tested are symmetric about x, so for all these sections it is known that the neutral
Explicitly plug this in to speed *)
(*  up the netural axis calculations.
The only section this may not hold for are L and T1, and C1, will need to change this approximation if evaluat
   xc = width/2.0;
   yp0 = 0.0;
   yp1 = height;
   vy0 = Re[NIntegrate[Forcex[x, y, xc, yp0], {x, 0, width}, {y, 0, height},
AccuracyGoal -> 8]];
   vy1 = Re[ NIntegrate[Forcex[x, y, xc, yp1], {x, 0, width}, {y, 0, height},
AccuracyGoal -> 8]];
   For[i = 0, i < 20, ++i,
      ypm = (yp0 + yp1)/2.0;
      vym =  Re[NIntegrate[ Forcex[x, y, xc, ypm], {x, 0, width}, {y, 0, height},
AccuracyGoal -> 8]];
      If[(vy0 <= 0 && vym >= 0)  || (vy0 >= 0 && vym <= 0),
       yp1 = ypm;
       vy1 = vym;
       Continue[];
      ];
      If[(vym <= 0 && vy1 >= 0)  || (vym >= 0 && vy1 <= 0),
       yp0 = ypm;
       vy0 = vym;
       Continue[];
      ];
    ];
   yc = (yp0 + yp1)/2.0;
(*  Integrate to compute the actual moment assocaited with the solved for refernce axis. *)
   MomentRR[x_, y_, xc_, yc_] = (x - xc)*Forcey[x, y, xc, yc] - (y - yc)*
Forcex[x, y, xc, yc];
   MomentTot =  NIntegrate[ MomentRR[x, y, xc, yc], {x, 0, width}, {y, 0, height}];
   Re[MomentTot]
   ];
(*
Calculate torsion based on engineering assumptions, I.e., known elastic J and max torsion resitance Tmax. *)
MrrEnginneeringApprox[dthet_] := Module[
   {Melastic, MomentTot},
   Clear[Melastic, MomentTot];
   (*
Compute the elastic moment, if this is less than the maximum plastic moment use it, othersise use max plastic
   Melastic = J*dthet*gm;
   If[Melastic < Tmax, MomentTot = Melastic, MomentTot = Tmax];
   MomentTot
   ];
(*============================================================================*)

(*============================================================================*)
(* DATA GATHERING BLOCK *)
(*
Given applied total displacements, compute incremental dtheta and dL, pass those values to the moment and forc
DataMtt = {};
DataMss = {};
DataFrr = {};
```

```
DataMrrNoWarping = {};
DataMrrEngineeringApprox = {};
Nstep = 25;
Tterm = 1.0;
RotFinalTT = 1.0*10^-4;
RotFinalSS = 1.0*10^-4;
RotFinalRR = 1.0*10^-4;
ExtFinalRR = 1.0*10^-5;
For[dt = 0.0, dt <= 1.0, dt += Tterm/Nstep,
  dthetTT = (dt*1.0*RotFinalTT)/beamLen;
  dthetSS = (dt*1.0*RotFinalSS)/beamLen;
  dthetRR = (dt*1.0*RotFinalRR)/beamLen;
  dLRR = (dt*1.0*ExtFinalRR)/beamLen;
  valMtt = Mtt[dthetTT];
  valMss = Mss[dthetSS];
  valMrrNoWarping = MrrNoWarping[dthetRR];
  valMrrEngineeringApprox = MrrEngineeringApprox[dthetRR];
  valFrr = Frr[dLRR];
  DataMtt = Append[DataMtt, {dt, valMtt}];
  DataMss = Append[DataMss, {dt, valMss}];
  DataMrrNoWarping = Append[DataMrrNoWarping, {dt, valMrrNoWarping}];
  DataMrrEngineeringApprox = Append[DataMrrEngineeringApprox, {dt, valMrrEngineeringApprox}];
  DataFrr = Append[DataFrr, {dt, valFrr}];
  Print[dt, ": ", valMtt, " ", valMss, " ", valMrrNoWarping, " ",
   valMrrEngineeringApprox, " ", valFrr]
  ];
ListPlot[DataMtt]
ListPlot[DataMss]
ListPlot[DataMrrNoWarping]
ListPlot[DataMrrEngineeringApprox]
ListPlot[DataFrr]
(*=============================================================================*)

(*=============================================================================*)
(* DATA PRINT BLOCK *)
(*  Print the section response data points in a form usable by sierra *)
Print["Begin Function AnalyticMtt", SectName];
Print["  type is piecewise linear"]
Print["  begin values"];
For[i = 1, i <= Length[DataMtt], i++,
  Print["    ", DataMtt[[i]][[1]], " ", DataMtt[[i]][[2]]];
  ];
Print["  end values"];
Print["End function AnalyticMtt", SectName];
Print[""];
Print["Begin Function AnalyticMss", SectName];
Print["  type is piecewise linear"]
Print["  begin values"];
For[i = 1, i <= Length[DataMss], i++,
  Print["    ", DataMss[[i]][[1]], " ", DataMss[[i]][[2]]];
  ];
Print["  end values"];
Print["End function AnalyticMss", SectName];
Print[""];
Print["Begin Function AnalyticMrrNoWarping", SectName];
Print["  type is piecewise linear"]
Print["  begin values"];
For[i = 1, i <= Length[DataMrrNoWarping], i++,
  Print["    ", DataMrrNoWarping[[i]][[1]], " ", DataMrrNoWarping[[i]][[2]]];
  ];
Print["  end values"];
Print["End function AnalyticMrrNoWarping", SectName];
Print[""];
Print["Begin Function AnalyticMrrEngineeringApprox", SectName];
Print["  type is piecewise linear"]
Print["  begin values"];
For[i = 1, i <= Length[DataMrrEngineeringApprox], i++,
  Print["    ", DataMrrEngineeringApprox[[i]][[1]], " ",
```

```
    DataMrrEngineeringApprox[[i]][[2]]];
  ];
Print["  end values"];
Print["End function AnalyticMrrEngineeringApprox", SectName];
Print[""];
Print["Begin Function AnalyticFrr", SectName];
Print["  type is piecewise linear"]
Print["  begin values"];
For[i = 1, i <= Length[DataFrr], i++,
  Print["    ", DataFrr[[i]][[1]], " ", DataFrr[[i]][[2]]];
  ];
Print["  end values"];
Print["End function AnalyticFrr", SectName];
(*=============================================================================*)

#{endif}
#
#  Analytic beam responses from the above Mathematica code
#
Begin Function AnalyticMttHat
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000177812
    0.08 0.0000355623
    0.12 0.0000533435
    0.16 0.0000711247
    0.2 0.0000889059
    0.24 0.000106687
    0.28 0.000124468
    0.32 0.000142249
    0.36 0.000158804
    0.4 0.00016808
    0.44 0.000173186
    0.48 0.000177359
    0.52 0.000180819
    0.56 0.000183725
    0.6 0.000186193
    0.64 0.00018831
    0.68 0.000190132
    0.72 0.000191667
    0.76 0.000192964
    0.8 0.000194072
    0.84 0.000195023
    0.88 0.000195847
    0.92 0.000196565
    0.96 0.000197194
    1. 0.000197749
  end values
End function AnalyticMttHat

Begin Function AnalyticMssHat
  type is piecewise linear
  begin values
    0. 0
    0.04 0.000124336
    0.08 0.000248672
    0.12 0.000360416
    0.16 0.000418595
    0.2 0.000455015
    0.24 0.00048287
    0.28 0.000505496
    0.32 0.000512978
    0.36 0.000515711
    0.4 0.000517667
    0.44 0.000519112
    0.48 0.000520212
    0.52 0.000521068
```

```
    0.56 0.000521748
    0.6 0.000522296
    0.64 0.000522744
    0.68 0.000523116
    0.72 0.000523429
    0.76 0.000523691
    0.8 0.000523917
    0.84 0.000524111
    0.88 0.000524277
    0.92 0.000524425
    0.96 0.000524552
    1. 0.000524666
  end values
End function AnalyticMssHat

Begin Function AnalyticMrrNoWarpingHat
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000710589
    0.08 0.000142118
    0.12 0.000205361
    0.16 0.000239442
    0.2 0.000261395
    0.24 0.000277704
    0.28 0.000287394
    0.32 0.000290289
    0.36 0.000291849
    0.4 0.000292932
    0.44 0.00029366
    0.48 0.000294058
    0.52 0.000294227
    0.56 0.000294269
    0.6 0.000294269
    0.64 0.000294269
    0.68 0.000294269
    0.72 0.000294269
    0.76 0.000294269
    0.8 0.000294269
    0.84 0.000294269
    0.88 0.000294269
    0.92 0.000294269
    0.96 0.000294269
    1. 0.000294269
  end values
End function AnalyticMrrNoWarpingHat

Begin Function AnalyticMrrEngineeringApproxHat
  type is piecewise linear
  begin values
    0. 0.
    0.04 5.93333e-7
    0.08 1.18667e-6
    0.12 1.78e-6
    0.16 2.37333e-6
    0.2 2.96667e-6
    0.24 3.56e-6
    0.28 4.15333e-6
    0.32 4.74667e-6
    0.36 5.34e-6
    0.4 5.93333e-6
    0.44 6.52667e-6
    0.48 7.12e-6
    0.52 7.71333e-6
    0.56 8.30667e-6
    0.6 8.9e-6
    0.64 9.49333e-6
    0.68 0.0000100867
```

```
    0.72 0.00001068
    0.76 0.0000112733
    0.8 0.0000118667
    0.84 0.00001246
    0.88 0.0000130533
    0.92 0.0000136467
    0.96 0.00001424
    1. 0.0000148333
  end values
End function AnalyticMrrEngineeringApproxHat


Begin Function AnalyticFrrHat
  type is piecewise linear
  begin values
    0. 0
    0.04 0.00116
    0.08 0.00232
    0.12 0.00348
    0.16 0.00464
    0.2 0.0058
    0.24 0.0058
    0.28 0.0058
    0.32 0.0058
    0.36 0.0058
    0.4 0.0058
    0.44 0.0058
    0.48 0.0058
    0.52 0.0058
    0.56 0.0058
    0.6 0.0058
    0.64 0.0058
    0.68 0.0058
    0.72 0.0058
    0.76 0.0058
    0.8 0.0058
    0.84 0.0058
    0.88 0.0058
    0.92 0.0058
    0.96 0.0058
    1. 0.0058
  end values
End function AnalyticFrrHat
Begin Function AnalyticMttBar
  type is piecewise linear
  begin values
    0. 0
    0.04 0.000133333
    0.08 0.000266666
    0.12 0.000399999
    0.16 0.000533332
    0.2 0.000666665
    0.24 0.00076852
    0.28 0.00082993
    0.32 0.00086979
    0.36 0.000897117
    0.4 0.000916669
    0.44 0.000931128
    0.48 0.000942128
    0.52 0.000950688
    0.56 0.000957481
    0.6 0.000962965
    0.64 0.000967446
    0.68 0.000971167
    0.72 0.000974278
    0.76 0.000976914
    0.8 0.000979169
    0.84 0.000981105
```

```
    0.88 0.00098278
    0.92 0.000984245
    0.96 0.000985531
    1. 0.000986665
  end values
End function AnalyticMttBar

Begin Function AnalyticMssBar
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000333332
    0.08 0.0000666665
    0.12 0.0000999997
    0.16 0.000133333
    0.2 0.000166666
    0.24 0.000199999
    0.28 0.000233333
    0.32 0.000266666
    0.36 0.000299999
    0.4 0.000333332
    0.44 0.00036226
    0.48 0.000384258
    0.52 0.00040138
    0.56 0.000414967
    0.6 0.000425925
    0.64 0.000434895
    0.68 0.000442331
    0.72 0.000448559
    0.76 0.000453833
    0.8 0.000458332
    0.84 0.000462206
    0.88 0.000465566
    0.92 0.000468493
    0.96 0.000471064
    1. 0.000473332
  end values
End function AnalyticMssBar

Begin Function AnalyticMrrNoWarpingBar
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000833333
    0.08 0.000166667
    0.12 0.00025
    0.16 0.000333333
    0.2 0.000415009
    0.24 0.000472429
    0.28 0.000508213
    0.32 0.000532089
    0.36 0.000548719
    0.4 0.000560508
    0.44 0.000568647
    0.48 0.000574295
    0.52 0.000578338
    0.56 0.000581307
    0.6 0.000583537
    0.64 0.000585244
    0.68 0.000586573
    0.72 0.000587622
    0.76 0.000588462
    0.8 0.000589143
    0.84 0.0005897
    0.88 0.00059016
    0.92 0.000590544
    0.96 0.000590866
    1. 0.000591139
```

```
  end values
End function AnalyticMrrNoWarpingBar

Begin Function AnalyticMrrEngineeringApproxBar
  type is piecewise linear
  begin values
    0. 0.
    0.04 0.0000458
    0.08 0.0000916
    0.12 0.0001374
    0.16 0.0001832
    0.2 0.000229
    0.24 0.0002748
    0.28 0.0003206
    0.32 0.0003664
    0.36 0.0004122
    0.4 0.000416216
    0.44 0.000416216
    0.48 0.000416216
    0.52 0.000416216
    0.56 0.000416216
    0.6 0.000416216
    0.64 0.000416216
    0.68 0.000416216
    0.72 0.000416216
    0.76 0.000416216
    0.8 0.000416216
    0.84 0.000416216
    0.88 0.000416216
    0.92 0.000416216
    0.96 0.000416216
    1. 0.000416216
  end values
End function AnalyticMrrEngineeringApproxBar

Begin Function AnalyticFrrBar
  type is piecewise linear
  begin values
    0. 0
    0.04 0.004
    0.08 0.008
    0.12 0.012
    0.16 0.016
    0.2 0.02
    0.24 0.02
    0.28 0.02
    0.32 0.02
    0.36 0.02
    0.4 0.02
    0.44 0.02
    0.48 0.02
    0.52 0.02
    0.56 0.02
    0.6 0.02
    0.64 0.02
    0.68 0.02
    0.72 0.02
    0.76 0.02
    0.8 0.02
    0.84 0.02
    0.88 0.02
    0.92 0.02
    0.96 0.02
    1. 0.02
  end values
End function AnalyticFrrBar

Begin Function AnalyticMttRod
```

```
  type is piecewise linear
  begin values
    0. 0
    0.04 0.000157079
    0.08 0.000314158
    0.12 0.000471237
    0.16 0.000628316
    0.2 0.000785395
    0.24 0.000923749
    0.28 0.00102111
    0.32 0.00108898
    0.36 0.00113749
    0.4 0.00117311
    0.44 0.00119997
    0.48 0.00122067
    0.52 0.00123694
    0.56 0.00124995
    0.6 0.00126051
    0.64 0.0012692
    0.68 0.00127642
    0.72 0.00128249
    0.76 0.00128765
    0.8 0.00129206
    0.84 0.00129586
    0.88 0.00129917
    0.92 0.00130205
    0.96 0.00130458
    1. 0.00130682
  end values
End function AnalyticMttRod

Begin Function AnalyticMssRod
  type is piecewise linear
  begin values
    0. 0
    0.04 0.00015708
    0.08 0.000314158
    0.12 0.000471237
    0.16 0.000628316
    0.2 0.000785401
    0.24 0.000923756
    0.28 0.00102111
    0.32 0.00108899
    0.36 0.00113748
    0.4 0.00117312
    0.44 0.00119997
    0.48 0.00122067
    0.52 0.00123695
    0.56 0.00124995
    0.6 0.00126051
    0.64 0.00126919
    0.68 0.00127642
    0.72 0.0012825
    0.76 0.00128765
    0.8 0.00129206
    0.84 0.00129586
    0.88 0.00129916
    0.92 0.00130206
    0.96 0.00130458
    1. 0.00130683
  end values
End function AnalyticMssRod

Begin Function AnalyticMrrNoWarpingRod
  type is piecewise linear
  begin values
    0. 0
    0.04 0.00015708
```

```
    0.08 0.000314159
    0.12 0.000471239
    0.16 0.000628319
    0.2 0.000785398
    0.24 0.000895693
    0.28 0.00095179
    0.32 0.000983282
    0.36 0.00100231
    0.4 0.00101447
    0.44 0.00102261
    0.48 0.00102826
    0.52 0.0010323
    0.56 0.00103527
    0.6 0.0010375
    0.64 0.00103921
    0.68 0.00104054
    0.72 0.00104159
    0.76 0.00104243
    0.8 0.00104311
    0.84 0.00104366
    0.88 0.00104412
    0.92 0.00104451
    0.96 0.00104483
    1. 0.0010451
  end values
End function AnalyticMrrNoWarpingRod

Begin Function AnalyticMrrEngineeringApproxRod
  type is piecewise linear
  begin values
    0. 0.
    0.04 0.00015708
    0.08 0.000314159
    0.12 0.000471239
    0.16 0.000628319
    0.2 0.000785398
    0.24 0.000942478
    0.28 0.0010472
    0.32 0.0010472
    0.36 0.0010472
    0.4 0.0010472
    0.44 0.0010472
    0.48 0.0010472
    0.52 0.0010472
    0.56 0.0010472
    0.6 0.0010472
    0.64 0.0010472
    0.68 0.0010472
    0.72 0.0010472
    0.76 0.0010472
    0.8 0.0010472
    0.84 0.0010472
    0.88 0.0010472
    0.92 0.0010472
    0.96 0.0010472
    1. 0.0010472
  end values
End function AnalyticMrrEngineeringApproxRod

Begin Function AnalyticFrrRod
  type is piecewise linear
  begin values
    0. 0
    0.04 0.00628319
    0.08 0.0125664
    0.12 0.0188496
    0.16 0.0251327
    0.2 0.0314159
```

```
    0.24 0.0314159
    0.28 0.0314159
    0.32 0.0314159
    0.36 0.0314159
    0.4 0.0314159
    0.44 0.0314159
    0.48 0.0314159
    0.52 0.0314159
    0.56 0.0314159
    0.6 0.0314159
    0.64 0.0314159
    0.68 0.0314159
    0.72 0.0314159
    0.76 0.0314159
    0.8 0.0314159
    0.84 0.0314159
    0.88 0.0314159
    0.92 0.0314159
    0.96 0.0314159
    1. 0.0314159
  end values
End function AnalyticFrrRod

Begin Function AnalyticMttBox
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000555732
    0.08 0.000111146
    0.12 0.00016672
    0.16 0.000222293
    0.2 0.000277866
    0.24 0.000305703
    0.28 0.000317986
    0.32 0.000325958
    0.36 0.000331423
    0.4 0.000335333
    0.44 0.000338226
    0.48 0.000340426
    0.52 0.000342138
    0.56 0.000343496
    0.6 0.000344592
    0.64 0.000345489
    0.68 0.000346233
    0.72 0.000346856
    0.76 0.000347383
    0.8 0.000347833
    0.84 0.00034822
    0.88 0.000348556
    0.92 0.000348849
    0.96 0.000349106
    1. 0.000349333
  end values
End function AnalyticMttBox

Begin Function AnalyticMssBox
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000179734
    0.08 0.0000359467
    0.12 0.0000539201
    0.16 0.0000718934
    0.2 0.0000898668
    0.24 0.00010784
    0.28 0.000125814
    0.32 0.000143787
    0.36 0.00016176
```

```
    0.4 0.000179734
    0.44 0.000193299
    0.48 0.000199939
    0.52 0.000202138
    0.56 0.000203497
    0.6 0.000204593
    0.64 0.00020549
    0.68 0.000206233
    0.72 0.000206856
    0.76 0.000207383
    0.8 0.000207833
    0.84 0.000208221
    0.88 0.000208557
    0.92 0.000208849
    0.96 0.000209107
    1. 0.000209333
  end values
End function AnalyticMssBox

Begin Function AnalyticMrrNoWarpingBox
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000367733
    0.08 0.0000735467
    0.12 0.00011032
    0.16 0.000147093
    0.2 0.000182209
    0.24 0.00019754
    0.28 0.000205895
    0.32 0.000211478
    0.36 0.000215247
    0.4 0.000217644
    0.44 0.000218744
    0.48 0.000219046
    0.52 0.000219063
    0.56 0.000219063
    0.6 0.000219063
    0.64 0.000219063
    0.68 0.000219063
    0.72 0.000219063
    0.76 0.000219063
    0.8 0.000219063
    0.84 0.000219063
    0.88 0.000219063
    0.92 0.000219063
    0.96 0.000219063
    1. 0.000219063
  end values
End function AnalyticMrrNoWarpingBox

Begin Function AnalyticMrrEngineeringApproxBox
  type is piecewise linear
  begin values
    0. 0.
    0.04 0.0000208864
    0.08 0.0000417729
    0.12 0.0000626593
    0.16 0.0000835457
    0.2 0.000104432
    0.24 0.000125319
    0.28 0.000146205
    0.32 0.000167091
    0.36 0.000171
    0.4 0.000171
    0.44 0.000171
    0.48 0.000171
    0.52 0.000171
```

```
    0.56 0.000171
    0.6 0.000171
    0.64 0.000171
    0.68 0.000171
    0.72 0.000171
    0.76 0.000171
    0.8 0.000171
    0.84 0.000171
    0.88 0.000171
    0.92 0.000171
    0.96 0.000171
    1. 0.000171
  end values
End function AnalyticMrrEngineeringApproxBox

Begin Function AnalyticFrrBox
  type is piecewise linear
  begin values
    0. 0
    0.04 0.00112
    0.08 0.00224
    0.12 0.00336
    0.16 0.00448
    0.2 0.0056
    0.24 0.0056
    0.28 0.0056
    0.32 0.0056
    0.36 0.0056
    0.4 0.0056
    0.44 0.0056
    0.48 0.0056
    0.52 0.0056
    0.56 0.0056
    0.6 0.0056
    0.64 0.0056
    0.68 0.0056
    0.72 0.0056
    0.76 0.0056
    0.8 0.0056
    0.84 0.0056
    0.88 0.0056
    0.92 0.0056
    0.96 0.0056
    1. 0.0056
  end values
End function AnalyticFrrBox


Begin Function AnalyticMttI
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000853635
    0.08 0.000170727
    0.12 0.00025609
    0.16 0.000341454
    0.2 0.000426817
    0.24 0.000465352
    0.28 0.000471493
    0.32 0.000475479
    0.36 0.000478212
    0.4 0.000480167
    0.44 0.000481613
    0.48 0.000482713
    0.52 0.000483569
    0.56 0.000484248
    0.6 0.000484796
    0.64 0.000485245
```

```
    0.68 0.000485617
    0.72 0.000485928
    0.76 0.000486192
    0.8 0.000486417
    0.84 0.000486611
    0.88 0.000486778
    0.92 0.000486925
    0.96 0.000487053
    1. 0.000487167
  end values
End function AnalyticMttI

Begin Function AnalyticMssI
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000169034
    0.08 0.0000338068
    0.12 0.0000507102
    0.16 0.0000676136
    0.2 0.000084517
    0.24 0.00010142
    0.28 0.000117927
    0.32 0.000129915
    0.36 0.00013814
    0.4 0.000144034
    0.44 0.000148401
    0.48 0.000151729
    0.52 0.000154325
    0.56 0.000156392
    0.6 0.000158064
    0.64 0.000159438
    0.68 0.000160582
    0.72 0.000161543
    0.76 0.000162364
    0.8 0.000163067
    0.84 0.000163677
    0.88 0.000164209
    0.92 0.000164677
    0.96 0.000165089
    1. 0.000165459
  end values
End function AnalyticMssI

Begin Function AnalyticMrrNoWarpingI
  type is piecewise linear
  begin values
    0. 0
    0.04 0.0000511333
    0.08 0.000102267
    0.12 0.0001534
    0.16 0.000204533
    0.2 0.000243874
    0.24 0.000253751
    0.28 0.000256828
    0.32 0.000258827
    0.36 0.000260198
    0.4 0.00026118
    0.44 0.000261907
    0.48 0.00026246
    0.52 0.000262892
    0.56 0.000263234
    0.6 0.000263511
    0.64 0.000263738
    0.68 0.000263927
    0.72 0.000264085
    0.76 0.000264219
    0.8 0.000264333
```

```
    0.84 0.000264432
    0.88 0.000264518
    0.92 0.000264593
    0.96 0.000264659
    1. 0.000264717
  end values
End function AnalyticMrrNoWarpingI

Begin Function AnalyticMrrEngineeringApproxI
  type is piecewise linear
  begin values
    0. 0.
    0.04 3.94167e-7
    0.08 7.88333e-7
    0.12 1.1825e-6
    0.16 1.57667e-6
    0.2 1.97083e-6
    0.24 2.365e-6
    0.28 2.75917e-6
    0.32 3.15333e-6
    0.36 3.5475e-6
    0.4 3.94167e-6
    0.44 4.33583e-6
    0.48 4.73e-6
    0.52 5.12417e-6
    0.56 5.51833e-6
    0.6 5.9125e-6
    0.64 6.30667e-6
    0.68 6.70083e-6
    0.72 7.095e-6
    0.76 7.48917e-6
    0.8 7.88333e-6
    0.84 8.2775e-6
    0.88 8.67167e-6
    0.92 9.06583e-6
    0.96 9.46e-6
    1. 9.85417e-6
  end values
End function AnalyticMrrEngineeringApproxI

Begin Function AnalyticFrrI
  type is piecewise linear
  begin values
    0. 0.
    0.04 0.00124
    0.08 0.00248
    0.12 0.00372
    0.16 0.00496
    0.2 0.0062
    0.24 0.0062
    0.28 0.0062
    0.32 0.0062
    0.36 0.0062
    0.4 0.0062
    0.44 0.0062
    0.48 0.0062
    0.52 0.0062
    0.56 0.0062
    0.6 0.0062
    0.64 0.0062
    0.68 0.0062
    0.72 0.0062
    0.76 0.0062
    0.8 0.0062
    0.84 0.0062
    0.88 0.0062
    0.92 0.0062
    0.96 0.0062
```

```
    1. 0.0062
  end values
End function AnalyticFrrI



  begin finite element model beams
    database name = beamPropertyTest.g
    database type = exodusII

    begin parameters for block block_100
      material = mat1
      model = elastic_plastic
      section = hat
    end

    begin parameters for block block_200
      material = mat1
      model = elastic_plastic
      section = bar
    end

    begin parameters for block block_300
      material = mat1
      model = elastic_plastic
      section = box
    end

    begin parameters for block block_400
      material = mat1
      model = elastic_plastic
      section = rod
    end

    begin parameters for block block_500
      material = mat1
      model = elastic_plastic
      section = I
    end
  end

  begin adagio procedure beam_setup_test

    #
    # *** Time step control information
    begin time control

      begin time stepping block p1
        start time = 0.0
        begin parameters for adagio region adagio
          number of time steps = 25
        end parameters for adagio region
      end time stepping block p1

      termination time = 1.0

    end time control

    begin adagio region adagio

      use finite element model beams

     ###  output description ###
      begin results output results
        database name = beamElasticPlasticVerif.e
        database type = exodusII
        At Time 0.0, Increment = 1.0e-3
        nodal variables = displacement
```

```
        nodal variables = force_internal
        nodal variables = moment_internal
        element variables = beam_stress_axial
        element variables = beam_strain_axial
      end results output results


#
#  Axial extension boundary conditions.  Hold right node fixed, pull on left node
#
   Begin fixed displacement
     node set = nodeset_111 nodeset_211 nodeset_311 nodeset_411 nodeset_511
     component = xyz
   end
   Begin fixed rotation
     node set = nodeset_111 nodeset_211 nodeset_311 nodeset_411 nodeset_511
     component = xyz
   end
   Begin fixed displacement
     node set = nodeset_112 nodeset_212 nodeset_312 nodeset_412 nodeset_512
     component = xy
   end
   begin prescribed displacement
     node set = nodeset_112 nodeset_212 nodeset_312 nodeset_412 nodeset_512
     function = ramp
     scale factor = 1.0e-5
     component = z
   end
   Begin fixed rotation
     node set = nodeset_112 nodeset_212 nodeset_312 nodeset_412 nodeset_512
     component = xyz
   end
#
#  Uniform T bending boundary condition.  Rotate ends of beam in opposite directions by theta
#
   Begin fixed displacement
     node set = nodeset_121 nodeset_221 nodeset_321 nodeset_421 nodeset_521
     component = xyz
   end
   Begin fixed rotation
     node set = nodeset_121 nodeset_221 nodeset_321 nodeset_421 nodeset_521
     component = xyz
   end

   Begin fixed displacement
     node set = nodeset_122 nodeset_222 nodeset_322 nodeset_422 nodeset_522
     component = x
   end
   Begin fixed rotation
     node set = nodeset_122 nodeset_222 nodeset_322 nodeset_422 nodeset_522
     component = yz
   end
   begin prescribed rotation
     node set = nodeset_122 nodeset_222 nodeset_322 nodeset_422 nodeset_522
     function = ramp
     scale factor = 1.0e-4
     direction = x
   end
#
#  Uniform S bending boundary condition.  Rotate ends of beam in opposite directions by theta
#
   Begin fixed displacement
     node set = nodeset_131 nodeset_231 nodeset_331 nodeset_431 nodeset_531
     component = xyz
   end
   Begin fixed rotation
     node set = nodeset_131 nodeset_231 nodeset_331 nodeset_431 nodeset_531
     component = xyz
```

```
   end
   Begin fixed displacement
     node set = nodeset_132 nodeset_232 nodeset_332 nodeset_432 nodeset_532
     component = y
   end
   Begin fixed rotation
     node set = nodeset_132 nodeset_232 nodeset_332 nodeset_432 nodeset_532
     component = xz
   end
   begin prescribed rotation
     node set = nodeset_132 nodeset_232 nodeset_332 nodeset_432 nodeset_532
     function = ramp
     scale factor = 1.0e-4
     direction = y
   end
#
#  Uniform R torsion boundary condition. Hold left end fixed, rotate right end by theta
#
  begin fixed displacement
    node set = nodeset_141 nodeset_241 nodeset_341 nodeset_441 nodeset_541
    component = xyz
  end
  begin fixed rotation
    node set = nodeset_141 nodeset_241 nodeset_341 nodeset_441 nodeset_541
    component = xyz
  end
  begin fixed displacement
    node set = nodeset_142 nodeset_242 nodeset_342 nodeset_442 nodeset_542
    component = xyz
  end
  Begin fixed rotation
    node set = nodeset_142 nodeset_242 nodeset_342 nodeset_442 nodeset_542
    component = xy
  end
  begin prescribed rotation
    node set = nodeset_142 nodeset_242 nodeset_342 nodeset_442 nodeset_542
    function = ramp
    scale factor = 1.0e-4
    direction = z
  end
#
#  Extract the react resultants to verify for each loading condition
#
#========== HAT ==========
   begin user output
     node set = nodelist_112
     compute global F_hat_axial_c as average of nodal reaction(z)
     compute global F_hat_axial_a as function AnalyticFrrHat
     compute global hat_axial_err from expression "abs(F_hat_axial_c-F_hat_axial_a)/min(abs(F_hat_axial_a), ab
     compute at every step
   end
   begin user output
     node set = nodelist_122
     compute global M_hat_bend_t_c as average of nodal rotational_reaction(x)
     compute global M_hat_bend_t_a as function AnalyticMttHat
     compute global hat_mt_err from expression "abs(M_hat_bend_t_c-M_hat_bend_t_a)/min(abs(M_hat_bend_t_a), ab
     compute at every step
   end
   begin user output
     node set = nodelist_132
     compute global M_hat_bend_s_c as average of nodal rotational_reaction(y)
     compute global M_hat_bend_s_a as function AnalyticMssHat
     compute global hat_ms_err from expression "abs(M_hat_bend_s_c-M_hat_bend_s_a)/min(abs(M_hat_bend_s_a), ab
     compute at every step
   end
   begin user output
     node set = nodelist_142
     compute global M_hat_torsion_r_c as average of nodal rotational_reaction(z)
```

```
    compute global M_hat_torsion_r_a1 as function AnalyticMrrNoWarpingHat
    compute global M_hat_torsion_r_a2 as function AnalyticMrrEngineeringApproxHat
    compute global hat_mr_err from expression "abs(M_hat_torsion_r_c-M_hat_torsion_r_a2)/min(abs(M_hat_torsio
    compute at every step
  end
#========== BAR ==========
  begin user output
    node set = nodelist_212
    compute global F_bar_axial_c as average of nodal reaction(z)
    compute global F_bar_axial_a as function AnalyticFrrBar
    compute global bar_axial_err from expression "abs(F_bar_axial_c-F_bar_axial_a)/min(abs(F_bar_axial_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_222
    compute global M_bar_bend_t_c as average of nodal rotational_reaction(x)
    compute global M_bar_bend_t_a as function AnalyticMttBar
    compute global bar_mt_err from expression "abs(M_bar_bend_t_c-M_bar_bend_t_a)/min(abs(M_bar_bend_t_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_232
    compute global M_bar_bend_s_c as average of nodal rotational_reaction(y)
    compute global M_bar_bend_s_a as function AnalyticMssBar
    compute global bar_ms_err from expression "abs(M_bar_bend_s_c-M_bar_bend_s_a)/min(abs(M_bar_bend_s_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_242
    compute global M_bar_torsion_r_c as average of nodal rotational_reaction(z)
    compute global M_bar_torsion_r_a1 as function AnalyticMrrNoWarpingBar
    compute global M_bar_torsion_r_a2 as function AnalyticMrrEngineeringApproxBar
    compute global bar_mr_err from expression "abs(M_bar_torsion_r_c-M_bar_torsion_r_a2)/min(abs(M_bar_torsio
    compute at every step
  end
#========== BOX ==========
  begin user output
    node set = nodelist_312
    compute global F_box_axial_c as average of nodal reaction(z)
    compute global F_box_axial_a as function AnalyticFrrBox
    compute global box_axial_err from expression "abs(F_box_axial_c-F_box_axial_a)/min(abs(F_box_axial_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_322
    compute global M_box_bend_t_c as average of nodal rotational_reaction(x)
    compute global M_box_bend_t_a as function AnalyticMttBox
    compute global box_mt_err from expression "abs(M_box_bend_t_c-M_box_bend_t_a)/min(abs(M_box_bend_t_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_332
    compute global M_box_bend_s_c as average of nodal rotational_reaction(y)
    compute global M_box_bend_s_a as function AnalyticMssBox
    compute global box_ms_err from expression "abs(M_box_bend_s_c-M_box_bend_s_a)/min(abs(M_box_bend_s_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_342
    compute global M_box_torsion_r_c as average of nodal rotational_reaction(z)
    compute global M_box_torsion_r_a1 as function AnalyticMrrNoWarpingBox
    compute global M_box_torsion_r_a2 as function AnalyticMrrEngineeringApproxBox
    compute global box_mr_err from expression "abs(M_box_torsion_r_c-M_box_torsion_r_a2)/min(abs(M_box_torsio
    compute at every step
  end
#========== ROD ==========
  begin user output
    node set = nodelist_412
```

```
    compute global F_rod_axial_c as average of nodal reaction(z)
    compute global F_rod_axial_a as function AnalyticFrrRod
    compute global rod_axial_err from expression "abs(F_rod_axial_c-F_rod_axial_a)/min(abs(F_rod_axial_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_422
    compute global M_rod_bend_t_c as average of nodal rotational_reaction(x)
    compute global M_rod_bend_t_a as function AnalyticMttRod
    compute global rod_mt_err from expression "abs(M_rod_bend_t_c-M_rod_bend_t_a)/min(abs(M_rod_bend_t_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_432
    compute global M_rod_bend_s_c as average of nodal rotational_reaction(y)
    compute global M_rod_bend_s_a as function AnalyticMssRod
    compute global rod_ms_err from expression "abs(M_rod_bend_s_c-M_rod_bend_s_a)/min(abs(M_rod_bend_s_a), ab
    compute at every step
  end
  begin user output
    node set = nodelist_442
    compute global M_rod_torsion_r_c as average of nodal rotational_reaction(z)
    compute global M_rod_torsion_r_a1 as function AnalyticMrrNoWarpingRod
    compute global M_rod_torsion_r_a2 as function AnalyticMrrEngineeringApproxRod
    compute global rod_mr_err from expression "abs(M_rod_torsion_r_c-M_rod_torsion_r_a2)/min(abs(M_rod_torsio
    compute at every step
  end
#========== I ==========
  begin user output
    node set = nodelist_512
    compute global F_i_axial_c as average of nodal reaction(z)
    compute global F_i_axial_a as function AnalyticFrrI
    compute global i_axial_err from expression "abs(F_i_axial_c-F_i_axial_a)/min(abs(F_i_axial_a), abs(F_i_ax
    compute at every step
  end
  begin user output
    node set = nodelist_522
    compute global M_i_bend_t_c as average of nodal rotational_reaction(x)
    compute global M_i_bend_t_a as function AnalyticMttI
    compute global i_mt_err from expression "abs(M_i_bend_t_c-M_i_bend_t_a)/min(abs(M_i_bend_t_a), abs(M_i_be
    compute at every step
  end
  begin user output
    node set = nodelist_532
    compute global M_i_bend_s_c as average of nodal rotational_reaction(y)
    compute global M_i_bend_s_a as function AnalyticMssI
    compute global i_ms_err from expression "abs(M_i_bend_s_c-M_i_bend_s_a)/min(abs(M_i_bend_s_a), abs(M_i_be
    compute at every step
  end
  begin user output
    node set = nodelist_542
    compute global M_i_torsion_r_c as average of nodal rotational_reaction(z)
    compute global M_i_torsion_r_a1 as function AnalyticMrrNoWarpingI
    compute global M_i_torsion_r_a2 as function AnalyticMrrEngineeringApproxI
    compute global i_mr_err from expression "abs(M_i_torsion_r_c-M_i_torsion_r_a2)/min(abs(M_i_torsion_r_c),
    compute at every step
  end



  begin history output
    database name = beamElasticPlasticVerif.h
    database type = exodusII
    At Time 0.0, Increment = 1.0e-4
    variable = global F_hat_axial_a
    variable = global M_hat_bend_t_a
    variable = global M_hat_bend_s_a
    variable = global M_hat_torsion_r_a1
```

```
variable = global M_hat_torsion_r_a2

variable = global F_bar_axial_a
variable = global M_bar_bend_t_a
variable = global M_bar_bend_s_a
variable = global M_bar_torsion_r_a1
variable = global M_bar_torsion_r_a2

variable = global F_box_axial_a
variable = global M_box_bend_t_a
variable = global M_box_bend_s_a
variable = global M_box_torsion_r_a1
variable = global M_box_torsion_r_a2

variable = global F_rod_axial_a
variable = global M_rod_bend_t_a
variable = global M_rod_bend_s_a
variable = global M_rod_torsion_r_a1
variable = global M_rod_torsion_r_a2

variable = global F_i_axial_a
variable = global M_i_bend_t_a
variable = global M_i_bend_s_a
variable = global M_i_torsion_r_a1
variable = global M_i_torsion_r_a2

variable = global F_hat_axial_c
variable = global M_hat_bend_t_c
variable = global M_hat_bend_s_c
variable = global M_hat_torsion_r_c

variable = global F_bar_axial_c
variable = global M_bar_bend_t_c
variable = global M_bar_bend_s_c
variable = global M_bar_torsion_r_c

variable = global F_box_axial_c
variable = global M_box_bend_t_c
variable = global M_box_bend_s_c
variable = global M_box_torsion_r_c

variable = global F_rod_axial_c
variable = global M_rod_bend_t_c
variable = global M_rod_bend_s_c
variable = global M_rod_torsion_r_c

variable = global F_i_axial_c
variable = global M_i_bend_t_c
variable = global M_i_bend_s_c
variable = global M_i_torsion_r_c


variable = global hat_axial_err
variable = global hat_mt_err
variable = global hat_ms_err
variable = global hat_mr_err

variable = global bar_axial_err
variable = global bar_mt_err
variable = global bar_ms_err
variable = global bar_mr_err

variable = global box_axial_err
variable = global box_mt_err
variable = global box_ms_err
variable = global box_mr_err

variable = global rod_axial_err
```

```
     variable = global rod_mt_err
     variable = global rod_ms_err
     variable = global rod_mr_err

     variable = global i_axial_err
     variable = global i_mt_err
     variable = global i_ms_err
     variable = global i_mr_err
   end


#
#  elastic response solution verification, verify known error bounds are reproduced.
Verify one time in the early loading presumably elastic regime
#
   begin solution verification
    skip times = 0.0 to 0.0399
    skip times = 0.0401 to 2.0

    verify global hat_axial_err = 0.00 plus or minus 0.001
    verify global hat_mt_err = 1.5 plus or minus 0.2
    verify global hat_ms_err = 1.00 plus or minus 0.1
    verify global hat_mr_err = 2500 plus or minus 100

    verify global bar_axial_err = 0.00 plus or minus 0.001
    verify global bar_mt_err = 0.005 plus or minus 0.005
    verify global bar_ms_err = 0.02 plus or minus 0.02
    verify global bar_mr_err = 25 plus or minus 5.0

    verify global box_axial_err = 0.00 plus or minus 0.001
    verify global box_mt_err = 24 plus or minus 5
    verify global box_ms_err = 5 plus or minus 5
    verify global box_mr_err = 91 plus or minus 10

    verify global rod_axial_err = 0.00 plus or minus 0.001
    verify global rod_mt_err = 0.004 plus or minus 0.004
    verify global rod_ms_err = 0.004 plus or minus 0.004
    verify global rod_mr_err = 0.0 plus or minus 0.001

    verify global i_axial_err = 0.00 plus or minus 0.001
    verify global i_mt_err = 0.2 plus or minus 0.2
    verify global i_ms_err = 0.2 plus or minus 0.2
    verify global i_mr_err = 9668 plus or minus 500


    completion file = elastic_verif
   end

#
#  plastic response solution verification, verify known error bounds are reproduced.
Verify one time in the late loading presumably plastic regime
#
   begin solution verification
    skip times = 0.0 to 0.99

    verify global hat_axial_err = 0.0005 plus or minus 0.001
    verify global hat_mt_err = 0.36 plus or minus 0.1
    verify global hat_ms_err = 0.1 plus or minus 0.1
    verify global hat_mr_err = 1028 plus or minus 100

    verify global bar_axial_err = 0.000 plus or minus 0.001
    verify global bar_mt_err = 9 plus or minus 1
    verify global bar_ms_err = 14 plus or minus 1
    verify global bar_mr_err = 34 plus or minus 5.0

    verify global box_axial_err = 0.000 plus or minus 0.001
    verify global box_mt_err = 9 plus or minus 5
    verify global box_ms_err = 16 plus or minus 5
```

```
    verify global box_mr_err = 50 plus or minus 10

    verify global rod_axial_err = 0.000 plus or minus 0.001
    verify global rod_mt_err = 0.16 plus or minus 0.05
    verify global rod_ms_err = 0.16 plus or minus 0.05
    verify global rod_mr_err = 10 plus or minus 01

    verify global i_axial_err = 0.000 plus or minus 0.001
    verify global i_mt_err = 1.0 plus or minus 0.5
    verify global i_ms_err = 7.5 plus or minus 2.0
    verify global i_mr_err = 2400 plus or minus 500



    completion file = plastic_verif
  end




      begin solver
        begin cg
          maximum iterations = 5000
          target relative residual = 1.0e-5

        end
      end




    end adagio region adagio

  end

end
```

B.26. PRESSURE LOADED LAYERED CANTILEVER

See Section 3.13 for problem description.

B.26.1. Input File - Multiple Lofted Shells Implicit Dynamics

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
# This is the input file for the multiple lofted shells,
# implicit dynamics version of this test.

# shell formulation = bt_shell

begin sierra vp20


  ###########################################################
  # Aprepro Variables:
  # 0.2  transverse load
  # 30000000  Youngs Modulus
  # 10  length
  # 0.0625  layer thickness
  # 1  width
  # 30000000  Top Shell Modulus
  # 30000000  Bottom Shell Modulus
  # 0.5  Top Shell Loft
  # -0.5  Bottom Shell Loft
  # 0.0001627604167  I
  # 4882.8125  E
  # 0.0512  Tip displacement
  # 10  Base moment
  # 0.002  within 0.2 % of Analytic
  ###########################################################
  begin function ramp
    type is piecewise analytic
    begin expressions
      0.000  "0.5*(1-cos(x * pi))"
      1.000  "1.000"
      10.00  "1.000"
    end expressions
  end function

  begin material mat_1
    density =  1.000E-1
    begin parameters for model elastic
      youngs modulus = 30000000
      poissons ratio = 0.300000
    end parameters for model elastic
  end material mat_1

  begin material mat_2
    density =  1.000E-1
    begin parameters for model elastic
      youngs modulus = 30000000
      poissons ratio = 0.300000
    end parameters for model elastic
  end material mat_2


  begin shell section SHELL_1
    formulation = bt_shell
    thickness = 0.0625
    lofting factor = 0.5
    integration rule = lobatto
  end shell section SHELL_1x
```

```
begin shell section SHELL_2
  formulation = bt_shell
  thickness = 0.0625
  lofting factor = -0.5
  integration rule = lobatto
end shell section SHELL_2

begin layered shell section LAYERED_SHELL
  layer 1 = SHELL_2 mat_2 elastic
  layer 2 = SHELL_1 mat_1 elastic
end


begin finite element model vp20
  database name = vp20_layered-cantilever.shell.g
  database type = exodusII

  begin parameters for block block_1
    material mat_1
    model = elastic
    section = SHELL_1
    hourglass stiffness = 0.0
  end parameters for block block_1

  begin parameters for block block_2
    material mat_2
    model = elastic
    section = SHELL_2
    hourglass stiffness = 0.0
  end parameters for block block_2

end finite element model vp20

begin adagio procedure adagio_PROCEDURE
  begin time control
    begin time stepping block
      start time = 0.0
      begin parameters for adagio region adagio_REGION
        number of time steps = 101
      end
    end time stepping block
    termination time = 1.01
  end time control

  begin adagio region adagio_REGION


    use finite element model vp20

     begin results output adagio_output
      database name = vp20_layered-cantilever.e
      database type = exodusII
      at time 0.0 increment = 1.0e-2
      nodal Variables = displacement as displ
      nodal Variables = rotation as rot
      element variables = stress as stress
      element variables = von_mises as vonm
      nodal varialbes = force_external
      global variables = tip_disp
      global variables = base_moment
      global variables = axial_load
      global variables = transverse_load
      element variables = transform_shell_strain as tss
    end results output adagio_output

   begin user output
      node set = nodelist_3
```

```
      compute global tip_disp as max of nodal displacement(2)
    end

    begin user output
      node set = nodelist_1
      compute global base_moment as sum of nodal rotational_reaction(3)
    end

#
#  Verify the solution.  The problem is a cantalever beam with different materials
#
    begin solution verification
      verify global tip_disp = -0.0512
      skip times = 0.0 to 1.0
      relative tolerance = 0.002
      completion file = verif_tip_disp
    end

    begin solution verification
      verify global base_moment = 10
      skip times = 0.0 to 1.0
      relative tolerance = 0.002
      completion file = verif_M0
    end

    begin fixed displacement
      node set = nodelist_1
      components = X Y Z
    end

    begin fixed rotation
      node set = nodelist_1
      components = x y z
    end

    begin pressure
      surface = surface_1
      function = ramp
      scale factor = 0.2
    end

    # For Viz of Loft ## begin contact definition frictionless
    # For Viz of Loft ##   search      = dash
    # For Viz of Loft ##   enforcement = al
    # For Viz of Loft ##   skin all blocks = on
    # For Viz of Loft ##   begin debug
    # For Viz of Loft ##     visualize contact facets = on
    # For Viz of Loft ##   end
    # For Viz of Loft ##   begin interaction defaults
    # For Viz of Loft ##     self contact = on
    # For Viz of Loft ##     general contact = off
    # For Viz of Loft ##   end
    # For Viz of Loft ##   begin surface options
    # For Viz of Loft ##     coincident shell treatment = best
    # For Viz of Loft ##   end
    # For Viz of Loft ## end contact definition frictionless

    begin solver
      begin cg
        target relative residual = 1.0e-12
        reference = energy
        begin full tangent preconditioner
            tangent diagonal shift = 1.0e-6
        end
      end
    end

  end adagio region adagio_REGION
```

```
  end adagio procedure adagio_PROCEDURE
end sierra vp20
```

B.26.2. Input File - Multiple Lofted Shells Explicit Dynamics

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
# This is the input file for the multiple lofted shells,
# explicit dynamics version of this test.

# shell formulation = bt_shell

begin sierra vp20


  ##########################################################
  # Aprepro Variables:
  # 0.2  transverse load
  # 30000000  Youngs Modulus
  # 10  length
  # 0.0625  layer thickness
  # 1  width
  # 30000000  Top Shell Modulus
  # 30000000  Bottom Shell Modulus
  # 0.5  Top Shell Loft
  # -0.5  Bottom Shell Loft
  # 0.0001627604167  I
  # 4882.8125  E
  # 0.0512  Tip displacement
  # 10  Base moment
  # 0.002  within 0.2 % of Analytic
  ##########################################################
  begin function ramp
    type is piecewise analytic
    begin expressions
      0.000  "0.5*(1-cos(x * pi))"
      1.000  "1.000"
      10.00  "1.000"
    end expressions
  end function

  begin material mat_1
    density =  1.000E-1
    begin parameters for model elastic
      youngs modulus = 30000000
      poissons ratio = 0.300000
    end parameters for model elastic
  end material mat_1

  begin material mat_2
    density =  1.000E-1
    begin parameters for model elastic
      youngs modulus = 30000000
      poissons ratio = 0.300000
    end parameters for model elastic
  end material mat_2


  begin shell section SHELL_1
    formulation = bt_shell
    thickness = 0.0625
    lofting factor = 0.5
    integration rule = lobatto
  end shell section SHELL_1x

  begin shell section SHELL_2
```

```
  formulation = bt_shell
  thickness = 0.0625
  lofting factor = -0.5
  integration rule = lobatto
end shell section SHELL_2

begin layered shell section LAYERED_SHELL
  layer 1 = SHELL_2 mat_2 elastic
  layer 2 = SHELL_1 mat_1 elastic
end


begin finite element model vp20
  database name = vp20_layered-cantilever.shell.g
  database type = exodusII

  begin parameters for block block_1
    material mat_1
    model = elastic
    section = SHELL_1
    hourglass stiffness = 0.0
  end parameters for block block_1

  begin parameters for block block_2
    material mat_2
    model = elastic
    section = SHELL_2
    hourglass stiffness = 0.0
  end parameters for block block_2

end finite element model vp20

begin presto procedure presto_PROCEDURE
  begin time control
    begin time stepping block
      start time = 0.0
      begin parameters for presto region presto_REGION
      end
    end time stepping block
    termination time = 1.01
  end time control

  begin presto region presto_REGION


    use finite element model vp20

     begin results output adagio_output
      database name = vp20_layered-cantilever.e
      database type = exodusII
      at time 0.0 increment = 1.0e-2
      nodal Variables = displacement as displ
      nodal Variables = rotation as rot
      element variables = stress as stress
      element variables = von_mises as vonm
      nodal varialbes = force_external
      global variables = tip_disp
      global variables = base_moment
      global variables = axial_load
      global variables = transverse_load
      element variables = transform_shell_strain as tss
    end results output adagio_output

   begin user output
      node set = nodelist_3
      compute global tip_disp as max of nodal displacement(2)
    end
```

```
      begin user output
        node set = nodelist_1
        compute global base_moment as sum of nodal rotational_reaction(3)
      end

#
#  Verify the solution.  The problem is a cantalever beam with different materials
#
      begin solution verification
        verify global tip_disp = -0.0512
        skip times = 0.0 to 1.0
        relative tolerance = 0.002
        completion file = verif_tip_disp
      end

      begin solution verification
        verify global base_moment = 10
        skip times = 0.0 to 1.0
        relative tolerance = 0.002
        completion file = verif_M0
      end

      begin fixed displacement
        node set = nodelist_1
        components = X Y Z
      end

      begin fixed rotation
        node set = nodelist_1
        components = x y z
      end

      begin pressure
        surface = surface_1
        function = ramp
        scale factor = 0.2
      end

      # For Viz of Loft ## begin contact definition frictionless
      # For Viz of Loft ##   search      = dash
      # For Viz of Loft ##   enforcement = al
      # For Viz of Loft ##   skin all blocks = on
      # For Viz of Loft ##   begin debug
      # For Viz of Loft ##     visualize contact facets = on
      # For Viz of Loft ##   end
      # For Viz of Loft ##   begin interaction defaults
      # For Viz of Loft ##     self contact = on
      # For Viz of Loft ##     general contact = off
      # For Viz of Loft ##   end
      # For Viz of Loft ##   begin surface options
      # For Viz of Loft ##     coincident shell treatment = best
      # For Viz of Loft ##   end
      # For Viz of Loft ## end contact definition frictionless

    end presto region presto_REGION

  end presto procedure presto_PROCEDURE
end sierra vp20
```

B.26.3. Input File - Single Layered Shell Implicit Dynamics

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
# This is the input file for the single layered shell,
# implicit dynamics version of this test.

# shell formulation = bt_shell
```

```
begin sierra vp20


  ###########################################################
  # Aprepro Variables:
  # 0.2  transverse load
  # 30000000  Youngs Modulus
  # 10  length
  # 0.0625  layer thickness
  # 1  width
  # 30000000  Top Shell Modulus
  # 30000000  Bottom Shell Modulus
  # 0.5  Top Shell Loft
  # -0.5  Bottom Shell Loft
  # 0.0001627604167  I
  # 4882.8125  E
  # 0.0512  Tip displacement
  # 10  Base moment
  # 0.002  within 0.2 % of Analytic
  ###########################################################
  begin function ramp
    type is piecewise analytic
    begin expressions
      0.000  "0.5*(1-cos(x * pi))"
      1.000  "1.000"
      10.00  "1.000"
    end expressions
  end function

  begin material mat_1
    density =  1.000E-1
    begin parameters for model elastic
      youngs modulus = 30000000
      poissons ratio = 0.300000
    end parameters for model elastic
  end material mat_1

  begin material mat_2
    density =  1.000E-1
    begin parameters for model elastic
      youngs modulus = 30000000
      poissons ratio = 0.300000
    end parameters for model elastic
  end material mat_2


  begin shell section SHELL_1
    formulation = bt_shell
    thickness = 0.0625
    lofting factor = 0.5
    integration rule = lobatto
  end shell section SHELL_1x

  begin shell section SHELL_2
    formulation = bt_shell
    thickness = 0.0625
    lofting factor = -0.5
    integration rule = lobatto
  end shell section SHELL_2

  begin layered shell section LAYERED_SHELL
    layer 1 = SHELL_2 mat_2 elastic
    layer 2 = SHELL_1 mat_1 elastic
  end


  begin finite element model vp20
```

```
    database name = vp20_layered-cantilever.shell.layered.g
    database type = exodusII

    begin parameters for block block_1
      material mat_1
      model = elastic
      section = LAYERED_SHELL
      hourglass stiffness = 0.0
    end parameters for block block_1

  end finite element model vp20


  begin adagio procedure adagio_PROCEDURE
    begin time control
      begin time stepping block
        start time = 0.0
        begin parameters for adagio region adagio_REGION
          number of time steps = 101
        end
      end time stepping block
      termination time = 1.01
    end time control

    begin adagio region adagio_REGION


      use finite element model vp20

       begin results output adagio_output
        database name = vp20_layered-cantilever.e
        database type = exodusII
        at time 0.0 increment = 1.0e-2
        nodal Variables = displacement as displ
        nodal Variables = rotation as rot
        element variables = stress as stress
        element variables = von_mises as vonm
        nodal varialbes = force_external
        global variables = tip_disp
        global variables = base_moment
        global variables = axial_load
        global variables = transverse_load
        element variables = transform_shell_strain as tss
      end results output adagio_output

     begin user output
        node set = nodelist_3
        compute global tip_disp as max of nodal displacement(2)
      end

      begin user output
        node set = nodelist_1
        compute global base_moment as sum of nodal rotational_reaction(3)
      end

#
#  Verify the solution.  The problem is a cantalever beam with different materials
#
      begin solution verification
        verify global tip_disp = -0.0512
        skip times = 0.0 to 1.0
        relative tolerance = 0.002
        completion file = verif_tip_disp
      end

      begin solution verification
        verify global base_moment = 10
        skip times = 0.0 to 1.0
```

```
      relative tolerance = 0.002
      completion file = verif_M0
    end

    begin fixed displacement
      node set = nodelist_1
      components = X Y Z
    end

    begin fixed rotation
      node set = nodelist_1
      components = x y z
    end

    begin pressure
      surface = surface_1
      function = ramp
      scale factor = 0.2
    end

    # For Viz of Loft ## begin contact definition frictionless
    # For Viz of Loft ##   search      = dash
    # For Viz of Loft ##   enforcement = al
    # For Viz of Loft ##   skin all blocks = on
    # For Viz of Loft ##   begin debug
    # For Viz of Loft ##     visualize contact facets = on
    # For Viz of Loft ##   end
    # For Viz of Loft ##   begin interaction defaults
    # For Viz of Loft ##     self contact = on
    # For Viz of Loft ##     general contact = off
    # For Viz of Loft ##   end
    # For Viz of Loft ##   begin surface options
    # For Viz of Loft ##     coincident shell treatment = best
    # For Viz of Loft ##   end
    # For Viz of Loft ## end contact definition frictionless

    begin solver
      begin cg
        target relative residual = 1.0e-12
        reference = energy
        begin full tangent preconditioner
            tangent diagonal shift = 1.0e-6
        end
      end
    end

  end adagio region adagio_REGION

end adagio procedure adagio_PROCEDURE
end sierra vp20
```

B.26.4. Input File - Single Layered Shell Explicit Dynamics

```
$ Algebraic Preprocessor (Aprepro) version 5.11 (2019/02/27)
# This is the input file for the single layered shell,
# explicit dynamics version of this test.

# shell formulation = bt_shell

begin sierra vp20


  ########################################################
  # Aprepro Variables:
  # 0.2  transverse load
  # 30000000  Youngs Modulus
```

```
# 10  length
# 0.0625  layer thickness
# 1  width
# 30000000  Top Shell Modulus
# 30000000  Bottom Shell Modulus
# 0.5  Top Shell Loft
# -0.5  Bottom Shell Loft
# 0.0001627604167  I
# 4882.8125  E
# 0.0512  Tip displacement
# 10  Base moment
# 0.002  within 0.2 % of Analytic
##########################################################
begin function ramp
  type is piecewise analytic
  begin expressions
    0.000  "0.5*(1-cos(x * pi))"
    1.000  "1.000"
    10.00  "1.000"
  end expressions
end function

begin material mat_1
  density =  1.000E-1
  begin parameters for model elastic
    youngs modulus = 30000000
    poissons ratio = 0.300000
  end parameters for model elastic
end material mat_1

begin material mat_2
  density =  1.000E-1
  begin parameters for model elastic
    youngs modulus = 30000000
    poissons ratio = 0.300000
  end parameters for model elastic
end material mat_2


begin shell section SHELL_1
  formulation = bt_shell
  thickness = 0.0625
  lofting factor = 0.5
  integration rule = lobatto
end shell section SHELL_1x

begin shell section SHELL_2
  formulation = bt_shell
  thickness = 0.0625
  lofting factor = -0.5
  integration rule = lobatto
end shell section SHELL_2

begin layered shell section LAYERED_SHELL
  layer 1 = SHELL_2 mat_2 elastic
  layer 2 = SHELL_1 mat_1 elastic
end


begin finite element model vp20
  database name = vp20_layered-cantilever.shell.layered.g
  database type = exodusII

  begin parameters for block block_1
    material mat_1
    model = elastic
    section = LAYERED_SHELL
    hourglass stiffness = 0.0
```

```
    end parameters for block block_1

  end finite element model vp20


  begin presto procedure presto_PROCEDURE
    begin time control
      begin time stepping block
        start time = 0.0
        begin parameters for presto region presto_REGION
        end
      end time stepping block
      termination time = 1.01
    end time control

    begin presto region presto_REGION


      use finite element model vp20

       begin results output adagio_output
        database name = vp20_layered-cantilever.e
        database type = exodusII
        at time 0.0 increment = 1.0e-2
        nodal Variables = displacement as displ
        nodal Variables = rotation as rot
        element variables = stress as stress
        element variables = von_mises as vonm
        nodal varialbes = force_external
        global variables = tip_disp
        global variables = base_moment
        global variables = axial_load
        global variables = transverse_load
        element variables = transform_shell_strain as tss
      end results output adagio_output

     begin user output
        node set = nodelist_3
        compute global tip_disp as max of nodal displacement(2)
      end

      begin user output
        node set = nodelist_1
        compute global base_moment as sum of nodal rotational_reaction(3)
      end

#
#  Verify the solution.  The problem is a cantalever beam with different materials
#
      begin solution verification
        verify global tip_disp = -0.0512
        skip times = 0.0 to 1.0
        relative tolerance = 0.002
        completion file = verif_tip_disp
      end

      begin solution verification
        verify global base_moment = 10
        skip times = 0.0 to 1.0
        relative tolerance = 0.002
        completion file = verif_M0
      end

      begin fixed displacement
        node set = nodelist_1
        components = X Y Z
      end
```

```
      begin fixed rotation
        node set = nodelist_1
        components = x y z
      end

      begin pressure
        surface = surface_1
        function = ramp
        scale factor = 0.2
      end

      # For Viz of Loft ## begin contact definition frictionless
      # For Viz of Loft ##   search      = dash
      # For Viz of Loft ##   enforcement = al
      # For Viz of Loft ##   skin all blocks = on
      # For Viz of Loft ##   begin debug
      # For Viz of Loft ##     visualize contact facets = on
      # For Viz of Loft ##   end
      # For Viz of Loft ##   begin interaction defaults
      # For Viz of Loft ##     self contact = on
      # For Viz of Loft ##     general contact = off
      # For Viz of Loft ##   end
      # For Viz of Loft ##   begin surface options
      # For Viz of Loft ##     coincident shell treatment = best
      # For Viz of Loft ##   end
      # For Viz of Loft ## end contact definition frictionless

    end presto region presto_REGION

  end presto procedure presto_PROCEDURE
end sierra vp20
```

B.27. LINE WELD FORCE PER UNIT LENGTH

See Section 3.14 for problem description.

B.27.1. Input File - Force Version

```
begin sierra line_weld

  # max_force = {max_force = 1.25}
  # max_disp = {max_disp = 1.0e-4}
  # max_moment   = {max_moment = 0.0}
  # max_rotation = {max_rotation = 1.0e-4}
  # scale_factor = {scale_factor = max_disp}

  begin function translational_FpuL
    type = piecewise linear
    ordinate = force_per_unit_length
    abscissa = displacement
    begin values
      0        0.0
      {1.0e8*max_disp}   {1.0e8*max_force}
    end
  end

  begin function rotational_MpuL
    type = piecewise linear
    ordinate = moment_per_unit_length
    abscissa = rotation
    begin values
      0        0.0
      {1.0e8*max_rotation}   {1.0e8*max_moment}
    end
  end

  begin definition for function cosRamp
    type = analytic
    evaluate expression = "cos_ramp(x, 0.0, 0.80)*{scale_factor}"
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0
  define point center with coordinates 0. 0. 0.
  define axis x_axis with point center direction x
  define axis y_axis with point center direction y
  define axis z_axis with point center direction z

  begin property specification for material steel
    density = 7830
    begin parameters for model elastic
      youngs modulus = 2e11
      poissons ratio = 0.33
    end
  end

  begin truss section truss_section
    area = 1.0
  end

  begin shell section shell_section
    thickness = 0.1
  end

  begin finite element model line_weld
```

```
  database name = {mesh_name}.g
  begin block defaults
    material = steel
    model = elastic
  end

  begin parameters for block block_1
    section = shell_section
  end

  begin parameters for block block_2
    section = shell_section
  end

  begin parameters for block block_3
    section = truss_section
  end
end

begin presto procedure procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region region
        step interval = 100
        number of quasistatic time steps = 10000
      end
    end
    termination time = 1.0
  end

  begin presto region region

    use finite element model line_weld

    begin line weld
      surface = surface_1
      block = block_3
      r displacement function = translational_FpuL
      s displacement function = translational_FpuL
      t displacement function = translational_FpuL
      r rotation function = rotational_MpuL
      s rotation function = rotational_MpuL
      t rotation function = rotational_MpuL
      failure envelope exponent = 1.0
      failure decay cycles = 1
      search tolerance = {1e-4+sqrt(3.0*(0.02*0.02))}
    end

    begin results output output
      database name = force_{mesh_name}_{dir}.e
      at step 0, interval = 1
      nodal variables = displacement as disp
      nodal variables = force_external as f_ext
      nodal variables = force_internal as f_int
      nodal variables = LINE_WELD_force_applied as weld_force_applied
      nodal variables = LINE_WELD_moment_applied as weld_moment_applied
      element variables = LINE_WELD_force_at_death as weld_force_at_death
      element variables = LINE_WELD_death_flag
      element variables = LINE_WELD_weld_active
      element variables = LINE_WELD_force as weld_force
      element variables = LINE_WELD_moment as weld_moment
      global variables = max_disp
      global variables = max_rot
      global variables = max_weld_force
      global variables = max_weld_moment
    end
```

```
      begin user output
        block = block_3
        compute global max_disp as max of nodal displacement({dir})
        compute global max_weld_force as max of element LINE_WELD_force({dir})
        compute global min_weld_force as min of element LINE_WELD_force({dir})
        compute global avg_weld_force as average of element LINE_WELD_force({dir})
        compute global max_rot as max of nodal rotational_displacement({dir})
        compute global max_weld_moment as max of element LINE_WELD_moment({dir})
        compute global min_weld_moment as min of element LINE_WELD_moment({dir})
        compute global avg_weld_moment as average of element LINE_WELD_moment({dir})
      end

      begin solution verification
        verify global max_disp = function cosRamp
        tolerance = 1.0e-7
        completion file = v1
        skip times = 0 to 0.95
      end

      begin solution verification
        verify global max_weld_force = {max_force}
        verify global min_weld_force = {max_force}
        verify global avg_weld_force = {max_force}
        tolerance = 1.0e-4
        completion file = v2
        skip times = 0 to 0.95
      end

      begin fixed displacement
        node set = nodelist_1
        components = x y z
      end

      begin fixed rotation
        block = block_1 block_2
        components = x y z
      end

      begin prescribed displacement
        nodeset = nodeset_2
        component = {dir}
        function = cosRamp
      end

      begin fixed displacement
        node set = nodelist_2
        {if (dir=="x")}
        components = y z
        {elseif (dir=="y")}
        components = x z
        {else}
        components = x y
        {endif}
      end

    end
  end
end
```

B.27.2. Input File - Moment Version

```
begin sierra line_weld

  # max_force = {max_force = 1.25e6}
  # max_disp = {max_disp = 1.0e-4}
  # max_moment   = {max_moment = 1.25}
```

```
# max_rotation = {max_rotation = 1.0e-4}
# scale_factor = {scale_factor = max_rotation}

begin function translational_FpuL
  type = piecewise linear
  ordinate = force_per_unit_length
  abscissa = displacement
  begin values
    0        0.0
    {1.0e8*max_disp}   {1.0e8*max_force}
  end
end

begin function rotational_MpuL
  type = piecewise linear
  ordinate = moment_per_unit_length
  abscissa = rotation
  begin values
    0        0.0
    {1.0e8*max_rotation}   {1.0e8*max_moment}
  end
end

begin definition for function cosRamp
  type = analytic
  evaluate expression = "cos_ramp(x, 0.0, 0.80)*{scale_factor}"
end

define direction x with vector 1.0 0.0 0.0
define direction y with vector 0.0 1.0 0.0
define direction z with vector 0.0 0.0 1.0
define point center with coordinates 0. 0. 0.
define axis x_axis with point center direction x
define axis y_axis with point center direction y
define axis z_axis with point center direction z

begin property specification for material steel
  density = 7830
  begin parameters for model elastic
    youngs modulus = 2e11
    poissons ratio = 0.33
  end
end

begin truss section truss_section
  area = 1.0
end

begin shell section shell_section
  thickness = 0.1
end

begin finite element model line_weld
  database name = {mesh_name}.g
  begin block defaults
    material = steel
    model = elastic
  end

  begin parameters for block block_1
    section = shell_section
  end

  begin parameters for block block_2
    section = shell_section
  end

  begin parameters for block block_3
```

```
    section = truss_section
  end
end

begin presto procedure procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region region
        step interval = 100
        number of quasistatic time steps = 10000
      end
    end
    termination time = 1.0
  end

  begin presto region region

    use finite element model line_weld

    begin line weld
      surface = surface_1
      block = block_3
      r displacement function = translational_FpuL
      s displacement function = translational_FpuL
      t displacement function = translational_FpuL
      r rotation function = rotational_MpuL
      s rotation function = rotational_MpuL
      t rotation function = rotational_MpuL
      failure envelope exponent = 1.0
      failure decay cycles = 1
      search tolerance = {1e-4+sqrt(3.0*(0.02*0.02))}
    end

    begin results output output
      database name = moment_{mesh_name}_{dir}.e
      at step 0, interval = 1
      nodal variables = displacement as disp
      nodal variables = force_external as f_ext
      nodal variables = force_internal as f_int
      nodal variables = LINE_WELD_force_applied as weld_force_applied
      nodal variables = LINE_WELD_moment_applied as weld_moment_applied
      element variables = LINE_WELD_force_at_death as weld_force_at_death
      element variables = LINE_WELD_death_flag
      element variables = LINE_WELD_weld_active
      element variables = LINE_WELD_force as weld_force
      element variables = LINE_WELD_moment as weld_moment
      global variables = max_disp
      global variables = max_rot
      global variables = max_weld_force
      global variables = max_weld_moment
    end

    begin user output
      block = block_3
      compute global max_disp as max of nodal displacement({dir})
      compute global max_weld_force as max of element LINE_WELD_force({dir})
      compute global min_weld_force as min of element LINE_WELD_force({dir})
      compute global avg_weld_force as average of element LINE_WELD_force({dir})
      compute global max_rot as max of nodal rotational_displacement({dir})
      compute global max_weld_moment as max of element LINE_WELD_moment({dir})
      compute global min_weld_moment as min of element LINE_WELD_moment({dir})
      compute global avg_weld_moment as average of element LINE_WELD_moment({dir})
    end

    begin solution verification
      verify global max_rot = function cosRamp
```

```
      tolerance = 1.0e-6
      completion file = v1
      skip times = 0 to 0.95
    end

    begin solution verification
      verify global max_weld_moment = {max_moment}
      verify global min_weld_moment = {max_moment}
      verify global avg_weld_moment = {max_moment}
      tolerance = 2.0e-2
      completion file = v2
      skip times = 0 to 0.95
    end

    begin fixed displacement
      block = block_1
      components = x y z
    end

    begin fixed rotation
      nodeset = nodelist_1
      components = x y z
    end

    begin prescribed displacement
      nodeset = nodeset_2
      cylindrical axis = {dir}_axis
      function = cosRamp
    end

    begin prescribed displacement
      nodeset = nodeset_2
      radial axis = {dir}_axis
      function = SIERRA_CONSTANT_FUNCTION_ZERO
    end

  end
 end
end
```

B.28. LINE WELD CONVERGENCE

See Section 3.15 for problem description.

```
begin sierra line_weld

  # max_force = {max_force = 1.25}
  # max_disp = {max_disp = 1.0e-4}
  # max_moment   = {max_moment = 1.25}
  # max_rotation = {max_rotation = 1.0e-4}
  # scale_factor = {scale_factor = max_disp}

  begin function translational_FpuL
    type = piecewise linear
    ordinate = force_per_unit_length
    abscissa = displacement
    begin values
      0        0.0
      {1.0e8*max_disp}   {1.0e8*max_force}
    end
  end

  begin function rotational_MpuL
    type = piecewise linear
    ordinate = moment_per_unit_length
    abscissa = rotation
    begin values
      0        0.0
      {1.0e8*max_rotation}   {1.0e8*max_moment}
    end
  end

  begin definition for function cosRampLinX
    type = analytic
    expression variable: px = nodal model_coordinates(x)
    expression variable: t = global time
    evaluate expression = "px*cos_ramp(t, 0.0, 0.80)*{scale_factor}"
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0
  define point center with coordinates 0. 0. 0.
  define axis x_axis with point center direction x
  define axis y_axis with point center direction y
  define axis z_axis with point center direction z

  begin property specification for material steel
    density = 7830
    begin parameters for model elastic
      youngs modulus = 2e11
      poissons ratio = 0.33
    end
  end

  begin truss section truss_section
    area = 1.0
  end

  begin shell section shell_section
    thickness = 0.1
  end

  begin rigid body rb1
  end

  begin shell section rb1
```

```
  thickness = 0.1
  rigid body = rb1
end

begin finite element model line_weld
  database name = {mesh_name}.g
  begin block defaults
    material = steel
    model = elastic
  end

  begin parameters for block block_1
    section = rb1
  end

  begin parameters for block block_2
    section = shell_section
  end

  begin parameters for block block_3
    section = truss_section
  end
end

begin presto procedure procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region region
        step interval = 100
        number of quasistatic time steps = 10000
      end
    end
    termination time = 1.0
  end

  begin presto region region

    use finite element model line_weld

    begin line weld
      surface = surface_1
      block = block_3
      r displacement function = translational_FpuL
      s displacement function = translational_FpuL
      t displacement function = translational_FpuL
      r rotation function = rotational_MpuL
      s rotation function = SIERRA_CONSTANT_FUNCTION_ZERO
      t rotation function = SIERRA_CONSTANT_FUNCTION_ZERO
      failure envelope exponent = 1.0
      failure decay cycles = 1
      search tolerance = 1e-4
    end

    begin results output output
      database name = {mesh_name}.e
      at step 0, interval = 1
      nodal variables = displacement as disp
      nodal variables = force_external as f_ext
      nodal variables = force_internal as f_int
      nodal variables = LINE_WELD_force_applied as weld_force_applied
      nodal variables = LINE_WELD_moment_applied as weld_moment_applied
      element variables = LINE_WELD_force_at_death as weld_force_at_death
      element variables = LINE_WELD_death_flag
      element variables = LINE_WELD_weld_active
      element variables = LINE_WELD_force as weld_force
      element variables = LINE_WELD_moment as weld_moment
```

```
      end

      begin fixed displacement
        rigid body = rb1
        components = x y z
      end

      begin fixed rotation
        rigid body = rb1
        components = x y z
      end

      begin prescribed displacement
        nodeset = nodeset_2
        component = y
        function = cosRampLinX
      end

      begin prescribed displacement
        nodeset = nodeset_2
        component = z
        function = cosRampLinX
      end

    end
  end
end
```

B.29. LINE WELD FAILURE

See Section 3.16 for problem description.

B.29.1. Input File - Force Version

```
begin sierra line_weld

  # dt = { dt = 1.0e-5 }
  # num_to_peak = { num_to_peak = 20000 }
  # num_decay = { num_decay = 2000 }
  # ramp_time = { ramp_time = num_to_peak * dt }
  # end_time = { end_time = ramp_time + num_decay * dt }

  # max_force = {max_force = 1.25}
  # max_disp = {max_disp = 1.0e-4}
  # max_moment   = {max_moment = 1.25}
  # max_rotation = {max_rotation = 1.0e-4}
  # scale_factor = {scale_factor = max_disp}

  begin function translational_FpuL
    type = piecewise linear
    ordinate = force_per_unit_length
    abscissa = displacement
    begin values
      0        0.0
      {max_disp}   {max_force}
    end
  end

  begin function rotational_MpuL
    type = piecewise linear
    ordinate = moment_per_unit_length
    abscissa = rotation
    begin values
      0        0.0
      {1.0e8*max_rotation}   {1.0e8*max_moment}
    end
  end

  begin function driveFunc
    type = piecewise linear
    begin values
      0        0.0
      {ramp_time}   {scale_factor}
      {1.0e8*ramp_time}   {1.0e8*scale_factor}
    end
  end

  begin function expectedVal
    type = piecewise linear
    begin values
      0        0.0
      {ramp_time}   {max_force}
      {end_time}   0.0
    end
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0
  define point center with coordinates 0. 0. 0.
  define axis x_axis with point center direction x
  define axis y_axis with point center direction y
```

```
define axis z_axis with point center direction z

begin property specification for material steel
  density = 7830
  begin parameters for model elastic
    youngs modulus = 2e11
    poissons ratio = 0.33
  end
end

begin truss section truss_section
  area = 1.0
end

begin shell section shell_section
  thickness = 0.1
end

begin finite element model line_weld
  database name = {mesh_name}.g
  begin block defaults
    material = steel
    model = elastic
  end

  begin parameters for block block_1
    section = shell_section
  end

  begin parameters for block block_2
    section = shell_section
  end

  begin parameters for block block_3
    section = truss_section
  end
end

begin presto procedure procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region region
        user time step = {dt}
      end
    end
    termination time = {end_time}
  end

  begin presto region region

    use finite element model line_weld

    begin line weld
      surface = surface_1
      block = block_3
      r displacement function = translational_FpuL
      s displacement function = translational_FpuL
      t displacement function = translational_FpuL
      r rotation function = rotational_MpuL
      s rotation function = rotational_MpuL
      t rotation function = rotational_MpuL
      failure envelope exponent = 2.0
      failure decay cycles = {num_decay}
      search tolerance = 1e-4
    end
```

```
begin results output output
  database name = force_{mesh_name}_{dir}.e
  at step 0, interval = 1
  nodal variables = displacement as disp
  nodal variables = force_external as f_ext
  nodal variables = force_internal as f_int
  nodal variables = LINE_WELD_force_applied as weld_force_applied
  nodal variables = LINE_WELD_moment_applied as weld_moment_applied
  element variables = LINE_WELD_force_at_death as weld_force_at_death
  element variables = LINE_WELD_death_flag
  element variables = LINE_WELD_death_step
  element variables = LINE_WELD_weld_active
  element variables = LINE_WELD_force as weld_force
  element variables = LINE_WELD_moment as weld_moment
  global variables = max_disp
  global variables = max_rot
  global variables = max_weld_force
  global variables = max_weld_moment
  global variables = max_death_step
  global variables = max_death_flag
  global variables = max_force_at_death
  global variables = max_active
  global variables = min_disp
  global variables = min_rot
  global variables = min_weld_force
  global variables = min_weld_moment
  global variables = min_death_step
  global variables = min_death_flag
  global variables = min_force_at_death
  global variables = min_active
  global variables = max_weld_force_applied
  global variables = computed_val
  global variables = expected_val
end

begin user output
  block = block_3
  compute global max_disp as max of nodal displacement
  compute global min_disp as min of nodal displacement
  compute global max_weld_force as max of element LINE_WELD_force
  compute global min_weld_force as min of element LINE_WELD_force
  compute global max_rot as max of nodal rotational_displacement
  compute global min_rot as min of nodal rotational_displacement
  compute global max_weld_moment as max of element LINE_WELD_moment
  compute global min_weld_moment as min of element LINE_WELD_moment
  compute global max_death_step as max of element LINE_WELD_death_step
  compute global min_death_step as min of element LINE_WELD_death_step
  compute global max_death_flag as max of element LINE_WELD_death_flag
  compute global min_death_flag as min of element LINE_WELD_death_flag
  compute global max_force_at_death as max of element LINE_WELD_force_at_death
  compute global min_force_at_death as min of element LINE_WELD_force_at_death
  compute global max_active as max of element LINE_WELD_weld_active
  compute global min_active as min of element LINE_WELD_weld_active
  compute global max_weld_force_applied as max of nodal LINE_WELD_force_applied
  compute global computed_val as max of element LINE_WELD_force({dir})
  compute global expected_val as function expectedVal
end

begin solution verification
  verify global computed_val = function expectedVal
  tolerance = {5.0e-2*max_force}
  completion file = v1
end

begin fixed displacement
  node set = nodelist_1
  components = x y z
end
```

```
      begin fixed rotation
        block = block_1 block_2
        components = x y z
      end

      begin prescribed displacement
        nodeset = nodeset_2
        component = {dir}
        function = driveFunc
      end

      begin fixed displacement
        node set = nodelist_2
        {if (dir=="x")}
        components = y z
        {elseif (dir=="y")}
        components = x z
        {else}
        components = x y
        {endif}
      end

    end
  end
end
```

B.29.2. Input File - Moment Version

```
begin sierra line_weld

  # dt = { dt = 1.0e-5 }
  # num_to_peak = { num_to_peak = 20000 }
  # num_decay = { num_decay = 2000 }
  # ramp_time = { ramp_time = num_to_peak * dt }
  # end_time = { end_time = ramp_time + num_decay * dt }

  # max_force = {max_force = 1.25e6}
  # max_disp = {max_disp = 1.0e-4}
  # max_moment   = {max_moment = 1.25}
  # max_rotation = {max_rotation = 1.0e-4}
  # scale_factor = {scale_factor = max_rotation}

  begin function translational_FpuL
    type = piecewise linear
    ordinate = force_per_unit_length
    abscissa = displacement
    begin values
      0        0.0
      {1.0e8*max_disp}   {1.0e8*max_force}
    end
  end

  begin function rotational_MpuL
    type = piecewise linear
    ordinate = moment_per_unit_length
    abscissa = rotation
    begin values
      0        0.0
      {max_rotation}   {max_moment}
    end
  end

  begin function driveFunc
    type = piecewise linear
    begin values
      0        0.0
```

```
    {ramp_time}   {scale_factor}
    {1.0e8*ramp_time}   {1.0e8*scale_factor}
  end
end

begin function expectedVal
  type = piecewise linear
  begin values
    0        0.0
    {ramp_time}   {max_moment}
    {end_time}   0.0
  end
end

define direction x with vector 1.0 0.0 0.0
define direction y with vector 0.0 1.0 0.0
define direction z with vector 0.0 0.0 1.0
define point center with coordinates 0. 0. 0.
define axis x_axis with point center direction x
define axis y_axis with point center direction y
define axis z_axis with point center direction z

begin property specification for material steel
  density = 7830
  begin parameters for model elastic
    youngs modulus = 2e11
    poissons ratio = 0.33
  end
end

begin truss section truss_section
  area = 1.0
end

begin shell section shell_section
  thickness = 0.1
end

begin finite element model line_weld
  database name = {mesh_name}.g
  begin block defaults
    material = steel
    model = elastic
  end

  begin parameters for block block_1
    section = shell_section
  end

  begin parameters for block block_2
    section = shell_section
  end

  begin parameters for block block_3
    section = truss_section
  end
end

begin presto procedure procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for presto region region
        user time step = {dt}
      end
    end
    termination time = {end_time}
```

```
end

begin presto region region

  use finite element model line_weld

  begin line weld
    surface = surface_1
    block = block_3
    r displacement function = translational_FpuL
    s displacement function = translational_FpuL
    t displacement function = translational_FpuL
    r rotation function = rotational_MpuL
    s rotation function = rotational_MpuL
    t rotation function = rotational_MpuL
    failure envelope exponent = 2.0
    failure decay cycles = {num_decay}
    search tolerance = 1e-4
  end

  begin results output output
    database name = moment_{mesh_name}_{dir}.e
    at step 0, interval = 1
    nodal variables = displacement as disp
    nodal variables = force_external as f_ext
    nodal variables = force_internal as f_int
    nodal variables = LINE_WELD_force_applied as weld_force_applied
    nodal variables = LINE_WELD_moment_applied as weld_moment_applied
    element variables = LINE_WELD_force_at_death as weld_force_at_death
    element variables = LINE_WELD_death_flag
    element variables = LINE_WELD_death_step
    element variables = LINE_WELD_weld_active
    element variables = LINE_WELD_force as weld_force
    element variables = LINE_WELD_moment as weld_moment
    global variables = max_disp
    global variables = max_rot
    global variables = max_weld_force
    global variables = max_weld_moment
    global variables = max_death_step
    global variables = max_death_flag
    global variables = max_force_at_death
    global variables = max_active
    global variables = min_disp
    global variables = min_rot
    global variables = min_weld_force
    global variables = min_weld_moment
    global variables = min_death_step
    global variables = min_death_flag
    global variables = min_force_at_death
    global variables = min_active
    global variables = max_weld_force_applied
    global variables = computed_val
    global variables = expected_val
  end

  begin user output
    block = block_3
    compute global max_disp as max of nodal displacement
    compute global min_disp as min of nodal displacement
    compute global max_weld_force as max of element LINE_WELD_force
    compute global min_weld_force as min of element LINE_WELD_force
    compute global max_rot as max of nodal rotational_displacement
    compute global min_rot as min of nodal rotational_displacement
    compute global max_weld_moment as max of element LINE_WELD_moment
    compute global min_weld_moment as min of element LINE_WELD_moment
    compute global max_death_step as max of element LINE_WELD_death_step
    compute global min_death_step as min of element LINE_WELD_death_step
    compute global max_death_flag as max of element LINE_WELD_death_flag
```

```
      compute global min_death_flag as min of element LINE_WELD_death_flag
      compute global max_force_at_death as max of element LINE_WELD_force_at_death
      compute global min_force_at_death as min of element LINE_WELD_force_at_death
      compute global max_active as max of element LINE_WELD_weld_active
      compute global min_active as min of element LINE_WELD_weld_active
      compute global max_weld_force_applied as max of nodal LINE_WELD_force_applied
      compute global computed_val as max of element LINE_WELD_moment({dir})
      compute global expected_val as function expectedVal
    end

    begin solution verification
      verify global computed_val = function expectedVal
      tolerance = {5.0e-2*max_moment}
      completion file = v1
    end

    begin fixed displacement
      block = block_1
      components = x y z
    end

    begin fixed rotation
      nodeset = nodelist_1
      components = x y z
    end

    begin prescribed displacement
      nodeset = nodeset_2
      cylindrical axis = {dir}_axis
      function = driveFunc
    end

    begin prescribed displacement
      nodeset = nodeset_2
      radial axis = {dir}_axis
      function = SIERRA_CONSTANT_FUNCTION_ZERO
    end

  end
 end
end
```

B.30. CONTACT FRICTIONAL ENERGY

See Section 4.1 for problem description.

```
############### contact_energy_friction.explicit.i ##################
begin sierra contact_energy_friction_explicit

  begin function pressure
    type is piecewise linear
    begin values
      0.0       0.0
      0.005     1.0
      1.0e6     1.0
    end values
  end function pressure

  begin function slide
    type is piecewise linear
    begin values
      0.0      0.0
      0.005    0.0
      0.025    0.2
    end values
  end function slide

  begin function analyticContactEnergy
    type is piecewise analytic
    begin expressions
      0.0    "0.0"
      0.005 "- (120.0 * 0.1 )*( x - 0.005 )*( 10.0 ) "
    end
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin material linear_elastic
    density         = 7.4e-4
    begin parameters for model elastic
      youngs modulus  = 30e6
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic

  begin finite element model mesh1
    Database Name = contact_energy_friction.g
    Database Type = exodusII

    begin parameters for block block_1
      material = linear_elastic
      model = elastic
    end parameters for block block_1

    begin parameters for block block_2
      material = linear_elastic
      model = elastic
    end parameters for block block_2


  end finite element model mesh1


  begin presto procedure Presto_Procedure

    begin time control
      begin time stepping block p1
```

```
    start time = 0.0
    begin parameters for presto region presto_region
      time step scale factor = 1.0
      time step increase factor = 2.0
      step interval = 1000
    end parameters for presto region presto_region
  end time stepping block p1
  termination time = 0.015
end time control

begin presto region presto_region
  use finite element model mesh1

  ### output description ###
  begin Results Output output_presto
    Database Name = contact_energy_friction.explicit.e
    Database Type = exodusII
    At Time 0.0, Increment = 0.0015
    nodal Variables = displacement as displ
    nodal Variables = velocity as vel
    nodal variables = force_contact as fcon
    element variables = stress as stress
    global Variables = timestep as timestep
    global Variables = contact_energy
    global variables = external_energy as ExternalEnergy
    global variables = internal_energy as InternalEnergy
    global variables = kinetic_energy as KineticEnergy
    global variables = momentum as Momentum
    global variables = analyticCE
  end


  begin history output hist
    Database Name = contact_energy_friction.explicit.h
    Database Type = exodusII
    At Step 0, interval = 50
    variable = nodal displacement at node 4 as displacement
    variable = global contact_energy
    variable = global fsum1
    variable = global fsum2
    variable = global analyticCE
  end history output hist

  begin user output
    compute global analyticCE as function analyticContactEnergy
  end


  ### definition of BCs ###

  begin prescribed displacement
    block = block_1
    component = x
    function = slide
  end prescribed displacement

  begin fixed displacement
    block = block_1
    components = y z
  end

  begin fixed displacement
    surface = surface_20
    component = x
  end

  begin user output
    block = block_1
```

```
        compute global fsum1 as sum of nodal force_contact
      end

      begin user output
        block = block_2
        compute global fsum2 as sum of nodal force_contact
      end

      begin prescribed displacement
        surface = surface_20
        component = y
        function = pressure
        scale factor = -4.0e-6
      end prescribed displacement

      begin contact definition
        search = dash
        contact surface surf_10  contains surface_10
        contact surface surf_2 contains surface_2

        begin constant friction model cFric
          friction coefficient = 0.1
        end constant friction model cFric

        begin interaction
          master = surf_2
          slave = surf_10
          normal tolerance = 0.01
          friction model = cFric
        end interaction

        begin enforcement options
          momentum balance iterations = 100
        end

      end contact definition

      begin solution verification
        verify global contact_energy = function analyticContactEnergy
        relative tolerance = 0.03
        skip times = 0.0 to 0.006
        completion file = contact_energy_friction.explicit.verif
      end

    end presto region presto_region
  end presto procedure Presto_Procedure

end sierra contact_energy_friction_explicit
################ contact_energy_friction.impd.i ##################
begin sierra contact_energy_friction_impd

  begin function pressure
    type is piecewise linear
    begin values
      0.0       0.0
      0.005     1.0
      1.0e6     1.0
    end values
  end function pressure

  begin function slide
    type is piecewise linear
    begin values
      0.0      0.0
      0.005    0.0
      0.025    0.2
    end values
  end function slide
```

```
begin function analyticContactEnergy
  type is piecewise analytic
  begin expressions
    0.0    "0.0"
    0.005 "- (120.0 * 0.1 )*( x - 0.005 )*( 10.0 ) "
  end
end

define direction x with vector 1.0 0.0 0.0
define direction y with vector 0.0 1.0 0.0
define direction z with vector 0.0 0.0 1.0

begin material linear_elastic
  density         = 7.4e-4
  begin parameters for model elastic
    youngs modulus  = 30e6
    poissons ratio  = 0.0
  end parameters for model elastic
end material linear_elastic

begin finite element model mesh1
  Database Name = contact_energy_friction.g
  Database Type = exodusII

  begin parameters for block block_1
    material = linear_elastic
    model = elastic
  end parameters for block block_1

  begin parameters for block block_2
    material = linear_elastic
    model = elastic
  end parameters for block block_2

end finite element model mesh1

begin adagio procedure Adagio_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for adagio region adagio_region
        number of time steps = 400
      end parameters for adagio region adagio_region
    end time stepping block p1
    termination time = 0.015
  end time control

  begin adagio region adagio_region
    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = contact_energy_friction.impd.e
      Database Type = exodusII
      At Time 0.0, Increment = 0.0015
      nodal Variables = displacement as displ
      nodal Variables = velocity as vel
      nodal variables = force_contact as fcon
      element variables = stress as stress
      global Variables = timestep as timestep
      global Variables = contact_energy
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
      global variables = analyticCE
```

```
end


begin history output hist
  Database Name = contact_energy_friction.impd.h
  Database Type = exodusII
  At Step 0, interval = 50
  variable = nodal displacement at node 4 as displacement
  variable = global contact_energy
  variable = global fsum1
  variable = global fsum2
  variable = global analyticCE
end history output hist

begin user output
  compute global analyticCE as function analyticContactEnergy
end


### definition of BCs ###

begin prescribed displacement
  block = block_1
  component = x
  function = slide
end prescribed displacement

begin fixed displacement
  block = block_1
  components = y z
end

begin fixed displacement
  surface = surface_20
  component = x
end

begin user output
  block = block_1
  compute global fsum1 as sum of nodal force_contact
end

begin user output
  block = block_2
  compute global fsum2 as sum of nodal force_contact
end

begin prescribed displacement
  surface = surface_20
  component = y
  function = pressure
  scale factor = -4.0e-6
end prescribed displacement

begin contact definition
  search = dash
  contact surface surf_10  contains surface_10
  contact surface surf_2 contains surface_2

  begin constant friction model cFric
    friction coefficient = 0.1
  end constant friction model cFric

  begin interaction
    master = surf_2
    slave = surf_10
    normal tolerance = 0.01
    friction model = cFric
```

```
      end interaction

    end contact definition

    begin solution verification
      verify global contact_energy = function analyticContactEnergy
      relative tolerance = 0.125
      skip times = 0.0 to 0.006
      completion file = contact_energy_friction.impd.verif
    end

    begin implicit dynamics
    end

    begin solver
      begin control contact
        target residual = 1.0e-6
        lagrange adaptive penalty           = uniform
        lagrange initialize                 = multiplier
      end
      begin cg
        target residual = 1.0e-7
        iteration print = 10
        begin full tangent preconditioner
          small number of iterations = 10
        end
      end
    end

  end adagio region adagio_region
end adagio procedure Adagio_Procedure

end sierra contact_energy_friction_impd
```

B.31. CONTACT ENERGY WITHOUT FRICTION

See Section 4.2 for problem description.

```
############### contact_energy_frictionless.explicit.i ##################
begin sierra contact_energy_frictionless_explicit

  begin function pressure
    type is piecewise linear
    begin values
      0.0       0.0
      0.005     1.0
      1.0e6     1.0
    end values
  end function pressure

  begin function slide
    type is piecewise linear
    begin values
      0.0      0.0
      0.005    0.0
      0.025    0.2
    end values
  end function slide

  begin function analyticContactEnergy
    type is piecewise analytic
    begin expressions
      0.0    "0.0"
      0.005  "0.0"
    end
  end

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin material linear_elastic
    density         = 7.4e-4
    begin parameters for model elastic
      youngs modulus  = 30e6
      poissons ratio  = 0.0
    end parameters for model elastic
  end material linear_elastic

  begin finite element model mesh1
    Database Name = contact_energy_frictionless.g
    Database Type = exodusII

    begin parameters for block block_1
      material = linear_elastic
      model = elastic
    end parameters for block block_1

    begin parameters for block block_2
      material = linear_elastic
      model = elastic
    end parameters for block block_2


  end finite element model mesh1


  begin presto procedure Presto_Procedure

    begin time control
      begin time stepping block p1
```

```
    start time = 0.0
    begin parameters for presto region presto_region
      time step scale factor = 1.0
      time step increase factor = 2.0
      step interval = 1000
    end parameters for presto region presto_region
  end time stepping block p1
  termination time = 0.015
end time control

begin presto region presto_region
  use finite element model mesh1

  ### output description ###
  begin Results Output output_presto
    Database Name = contact_energy_frictionless.explicit.e
    Database Type = exodusII
    At Time 0.0, Increment = 0.0015
    nodal Variables = displacement as displ
    nodal Variables = velocity as vel
    nodal variables = force_contact as fcon
    element variables = stress as stress
    global Variables = timestep as timestep
    global Variables = contact_energy
    global variables = external_energy as ExternalEnergy
    global variables = internal_energy as InternalEnergy
    global variables = kinetic_energy as KineticEnergy
    global variables = momentum as Momentum
    global variables = analyticCE
  end


  begin history output hist
    Database Name = contact_energy_frictionless.explicit.h
    Database Type = exodusII
    At Step 0, interval = 50
    variable = nodal displacement at node 4 as displacement
    variable = global contact_energy
    variable = global fsum1
    variable = global fsum2
    variable = global analyticCE
  end history output hist

  begin user output
    compute global analyticCE as function analyticContactEnergy
  end


  ### definition of BCs ###

  begin prescribed displacement
    block = block_1
    component = x
    function = slide
  end prescribed displacement

  begin fixed displacement
    block = block_1
    components = y z
  end

  begin fixed displacement
    block = block_2
    component = x
  end

  begin user output
    block = block_1
```

```
        compute global fsum1 as sum of nodal force_contact
      end

      begin user output
        block = block_2
        compute global fsum2 as sum of nodal force_contact
      end

      begin prescribed displacement
        surface = surface_20
        component = y
        function = pressure
        scale factor = -4.0e-6
      end prescribed displacement

      begin contact definition
        search = dash
        contact surface surf_10  contains surface_10
        contact surface surf_2 contains surface_2

        begin interaction
          master = surf_2
          slave = surf_10
          normal tolerance = 0.01
          friction model = frictionless
        end interaction

        begin enforcement options
          momentum balance iterations = 100
        end

      end contact definition

      begin solution verification
        verify global contact_energy = function analyticContactEnergy
        tolerance = 1.0e-2
        completion file = contact_energy_frictionless.explicit.verif
      end

    end presto region presto_region
  end presto procedure Presto_Procedure

end sierra contact_energy_frictionless_explicit
################ contact_energy_frictionless.impd.i ##################
begin sierra contact_energy_frictionless_impd

  begin function pressure
    type is piecewise linear
    begin values
      0.0       0.0
      0.005     1.0
      1.0e6     1.0
    end values
  end function pressure

  begin function slide
    type is piecewise linear
    begin values
      0.0      0.0
      0.005    0.0
      0.025    0.2
    end values
  end function slide

  begin function analyticContactEnergy
    type is piecewise analytic
    begin expressions
      0.0    "0.0"
```

```
    0.005  "0.0"
  end
end

define direction x with vector 1.0 0.0 0.0
define direction y with vector 0.0 1.0 0.0
define direction z with vector 0.0 0.0 1.0

begin material linear_elastic
  density         = 7.4e-4
  begin parameters for model elastic
    youngs modulus  = 30e6
    poissons ratio  = 0.0
  end parameters for model elastic
end material linear_elastic

begin finite element model mesh1
  Database Name = contact_energy_frictionless.g
  Database Type = exodusII

  begin parameters for block block_1
    material = linear_elastic
    model = elastic
  end parameters for block block_1

  begin parameters for block block_2
    material = linear_elastic
    model = elastic
  end parameters for block block_2


end finite element model mesh1


begin adagio procedure Adagio_Procedure

  begin time control
    begin time stepping block p1
      start time = 0.0
      begin parameters for adagio region adagio_region
        number of time steps = 500
      end parameters for adagio region adagio_region
    end time stepping block p1
    termination time = 0.015
  end time control

  begin adagio region adagio_region
    use finite element model mesh1

    ### output description ###
    begin Results Output output_adagio
      Database Name = contact_energy_frictionless.impd.e
      Database Type = exodusII
      At Time 0.0, Increment = 0.0015
      nodal Variables = displacement as displ
      nodal Variables = velocity as vel
      nodal variables = force_contact as fcon
      element variables = stress as stress
      global Variables = timestep as timestep
      global Variables = contact_energy
      global variables = external_energy as ExternalEnergy
      global variables = internal_energy as InternalEnergy
      global variables = kinetic_energy as KineticEnergy
      global variables = momentum as Momentum
      global variables = analyticCE
    end
```

```
begin history output hist
  Database Name = contact_energy_frictionless.impd.h
  Database Type = exodusII
  At Step 0, interval = 50
  variable = nodal displacement at node 4 as displacement
  variable = global contact_energy
  variable = global fsum1
  variable = global fsum2
  variable = global analyticCE
end history output hist

begin user output
  compute global analyticCE as function analyticContactEnergy
end


### definition of BCs ###

begin prescribed displacement
  block = block_1
  component = x
  function = slide
end prescribed displacement

begin fixed displacement
  block = block_1
  components = y z
end

begin fixed displacement
  block = block_2
  component = x
end

begin user output
  block = block_1
  compute global fsum1 as sum of nodal force_contact
end

begin user output
  block = block_2
  compute global fsum2 as sum of nodal force_contact
end

begin prescribed displacement
  surface = surface_20
  component = y
  function = pressure
  scale factor = -4.0e-6
end prescribed displacement

begin contact definition
  search = dash
  contact surface surf_10  contains surface_10
  contact surface surf_2 contains surface_2

  begin interaction
    master = surf_2
    slave = surf_10
    normal tolerance = 0.01
    friction model = frictionless
  end interaction


end contact definition

begin solution verification
  verify global contact_energy = function analyticContactEnergy
```

```
        tolerance = 1.5e-2
        completion file = contact_energy_frictionless.impd.verif
      end

      begin implicit dynamics
      end

      begin solver
        begin control contact
          target residual = 1.0e-6
        end
        begin cg
          target residual = 1.0e-8
        end
      end

    end adagio region adagio_region
  end adagio procedure Adagio_Procedure

end sierra contact_energy_frictionless_impd
```

B.32. EXTERNAL ENERGY DUE TO APPLIED FORCE

See Section 4.3 for problem description.

```
############### external_energy_nograv.explicit.i ##################
begin sierra external_energy_nograv
  title external energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin function CONSTANT
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
      -500.00         1.0
      0.00          1.0
      500.00          1.0
    end values
  end function CONSTANT

  ## a = 9.81 (acceleration)
  ## mass = volume * density
  ## mass = 1.0 * 1000.0
  ## distance = 0.5 * 1 * t^2
  ## force = mass * a
  begin function analyticExternalEnergy
    type is analytic
    evaluate expression = "0.5 * 1000.0 * 1.0 * 9.81 * 9.81 * x * x"
  end

  begin function FORCE
    type is piecewise linear
    ordinate is force
    abscissa is time
    begin values
      0.0     2452.5   ## force is 9810 N divided over 4 nodes
      500.00   2452.5
    end values
  end function FORCE

####### Material Definition #########

  begin material steel
    density = 1000    # actually, 7900.0 kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e+9 # GPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

####### BV Problem Definition #########

  begin solid section solid_1
    strain incrementation = strongly_objective
  end solid section solid_1

  begin finite element model model_1
    database name = external_energy_nograv.g

    begin parameters for block block_1
      material = steel
      model = elastic
```

```
      section = solid_1
    end parameters for block block_1

  end finite element model model_1

  begin presto procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for presto region region_1
        time step scale factor =  1.0
          step interval = 1000
        end parameters for presto region region_1
      end time stepping block time_control_1
      termination time = 0.1
    end time control

    begin presto region region_1
      use finite element model model_1

######## definition of BCs ###########

      begin prescribed force
        node set = nodelist_1
        direction = y
        function = FORCE
        scale factor = 1.0
        active periods = time_control_1
      end prescribed force

######### output description ##############

      begin results output output_1
        database name = external_energy_nograv.explicit.e
        at step 0 increment = 1000
        nodal variables = displacement
        nodal variables = acceleration
        nodal variables = force_external
        nodal variables = force_internal
        element variables = stress
        element variables = hourglass_energy
        global variables = artificial_energy
        global variables = external_energy
        global variables = internal_energy
        global variables = kinetic_energy
        global variables = strain_energy
        global variables = analyticEE
      end results output output_1

      begin history output history_1
        database name = external_energy_nograv.explicit.h
        at step 1 increment = 1
        ####   variable = nodal  acceleration
        variable = global artificial_energy
        variable = global external_energy
        variable = global internal_energy
        variable = global kinetic_energy
        variable = global strain_energy
        variable = global analyticEE
      end history output history_1

      begin user output
        compute global analyticEE as function analyticExternalEnergy
      end

      begin solution verification
        verify global external_energy = function analyticExternalEnergy
```

```
        relative tolerance = 0.005
        skip times = 0.0 to 0.01
        completion file = external_energy_nograve.explicit.verif
      end

    end presto region region_1

  end presto procedure procedure_1

end sierra external_energy_nograv
############### external_energy_nograv.impd.i ##################
begin sierra external_energy_nograv
  title external energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin function CONSTANT
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
      -500.00         1.0
      0.00          1.0
      500.00          1.0
    end values
  end function CONSTANT

  ## a = 9.81 (acceleration)
  ## mass = volume * density
  ## mass = 1.0 * 1000.0
  ## distance = 0.5 * 1 * t^2
  ## force = mass * a
  begin function analyticExternalEnergy
    type is analytic
    evaluate expression = "0.5 * 1000.0 * 1.0 * 9.81 * 9.81 * x * x"
  end

  begin function FORCE
    type is piecewise linear
    ordinate is force
    abscissa is time
    begin values
      0.0     2452.5   ## force is 9810 N divided over 4 nodes
      500.00   2452.5
    end values
  end function FORCE

####### Material Definition #########

  begin material steel
    density = 1000    # actually, 7900.0 kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e+9 # GPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

####### BV Problem Definition #########

  begin solid section solid_1
    strain incrementation = strongly_objective
  end solid section solid_1

  begin finite element model model_1
```

```
    database name = external_energy_nograv.g

    begin parameters for block block_1
      material = steel
      model = elastic
      section = solid_1
    end parameters for block block_1

  end finite element model model_1

  begin adagio procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for adagio region region_1
          time increment = 0.001
        end parameters for adagio region region_1
      end time stepping block time_control_1
      termination time = 0.1
    end time control

    begin adagio region region_1
      use finite element model model_1

######## definition of BCs ###########

      begin prescribed force
        node set = nodelist_1
        direction = y
        function = FORCE
        scale factor = 1.0
        active periods = time_control_1
      end prescribed force

######### output description ##############

      begin results output output_1
        database name = external_energy_nograv.impd.e
        at step 0 increment = 1
        nodal variables = displacement
        nodal variables = acceleration
        nodal variables = force_external
        nodal variables = force_internal
        element variables = stress
        element variables = hourglass_energy
        global variables = artificial_energy
        global variables = external_energy
        global variables = internal_energy
        global variables = kinetic_energy
        global variables = strain_energy
        global variables = analyticEE
      end results output output_1

      begin history output history_1
        database name = external_energy_nograv.impd.h
        at step 1 increment = 1
        ####   variable = nodal  acceleration
        variable = global artificial_energy
        variable = global external_energy
        variable = global internal_energy
        variable = global kinetic_energy
        variable = global strain_energy
        variable = global analyticEE
      end history output history_1

      begin user output
        compute global analyticEE as function analyticExternalEnergy
```

```
      end

      begin solution verification
        verify global external_energy = function analyticExternalEnergy
        relative tolerance = 0.005
        skip times = 0.0 to 0.01
        completion file = external_energy_nograve.explicit.verif
      end

      begin implicit dynamics
        alpha = 0.0
        beta = 0.25
        gamma = 0.5
      end

      begin solver
      end

    end adagio region region_1

  end adagio procedure procedure_1

end sierra external_energy_nograv
```

B.33. EXTERNAL ENERGY DUE TO GRAVITY

See Section 4.4 for problem description.

```
################ external_energy_wgrav.explicit.i ##################
begin sierra external_energy_wgrav
  title external energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin function CONSTANT
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
      -500.00  1.0
      0.00     1.0
      500.00   1.0
    end values
  end function CONSTANT

  ## a = 9.81 (acceleration)
  ## mass = volume * density
  ## mass = 1.0 * 1000.0
  ## distance = 0.5 * a * t^2
  ## force = mass * a
  ## external energy = force * distance
  begin function analyticExternalEnergy
    type is analytic
    evaluate expression = "0.5 * 1000.0 * 1.0 * 9.81 * 9.81 * x * x"
  end


####### Material Definition #########

  begin material steel
    density = 1000    # actually, 7900.0 kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e+9 # GPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

####### BV Problem Definition #########

  begin solid section solid_1
    strain incrementation = strongly_objective
  end solid section solid_1

  begin finite element model model_1
    database name = external_energy_wgrav.g

    begin parameters for block block_1
      material = steel
      model = elastic
      section = solid_1
    end parameters for block block_1

  end finite element model model_1

  begin presto procedure procedure_1

    begin time control
```

```
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for presto region region_1
        time step scale factor =  1.0
          step interval = 1
          user time step = 0.1
        end parameters for presto region region_1
      end time stepping block time_control_1
      termination time = 0.1
    end time control

    begin presto region region_1
      use finite element model model_1

######## definition of BCs ###########

      begin gravity
        include all blocks
        direction = y
        function = CONSTANT
        gravitational constant = 9.81
        scale factor = 1.0
        active periods = time_control_1
      end gravity

######### output description ##############

      begin results output output_1
        database name = external_energy_wgrav.explicit.e
        at step 0 increment = 1
        nodal variables = displacement
        nodal variables = acceleration
        element variables = stress
        element variables = hourglass_energy
        global variables = artificial_energy
        global variables = external_energy
        global variables = internal_energy
        global variables = kinetic_energy
        global variables = strain_energy
        global variables = analyticEE
      end results output output_1

      begin history output history_1
        database name = external_energy_wgrav.explicit.h
        at step 1 increment = 1
        ####   variable = nodal  acceleration
        variable = global artificial_energy
        variable = global external_energy
        variable = global internal_energy
        variable = global kinetic_energy
        variable = global strain_energy
        variable = global analyticEE
      end history output history_1

      begin user output
        compute global analyticEE as function analyticExternalEnergy
      end

      begin solution verification
        verify global external_energy = function analyticExternalEnergy
        relative tolerance = 0.005
        skip times = 0.0 to 0.01
        completion file = external_energy_wgrav.explicit.verif
      end

    end presto region region_1

  end presto procedure procedure_1
```

```
end sierra external_energy_wgrav
############### external_energy_wgrav.impd.i ##################
begin sierra external_energy_wgrav
  title external energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin function CONSTANT
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
      -500.00  1.0
      0.00     1.0
      500.00   1.0
    end values
  end function CONSTANT

  ## a = 9.81 (acceleration)
  ## mass = volume * density
  ## mass = 1.0 * 1000.0
  ## distance = 0.5 * a * t^2
  ## force = mass * a
  ## external energy = force * distance
  begin function analyticExternalEnergy
    type is analytic
    evaluate expression = "0.5 * 1000.0 * 1.0 * 9.81 * 9.81 * x * x"
  end


######## Material Definition #########

  begin material steel
    density = 1000    # actually, 7900.0 kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e+9 # GPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

######## BV Problem Definition #########

  begin solid section solid_1
    strain incrementation = strongly_objective
  end solid section solid_1

  begin finite element model model_1
    database name = external_energy_wgrav.g

    begin parameters for block block_1
      material = steel
      model = elastic
      section = solid_1
    end parameters for block block_1

  end finite element model model_1

  begin adagio procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for adagio region region_1
```

```
        time increment = 0.001
      end parameters for adagio region region_1
    end time stepping block time_control_1
    termination time = 0.1
  end time control

  begin adagio region region_1
    use finite element model model_1

######## definition of BCs ###########

    begin gravity
      include all blocks
      direction = y
      function = CONSTANT
      gravitational constant = 9.81
      scale factor = 1.0
      active periods = time_control_1
    end gravity

######### output description ##############

    begin results output output_1
      database name = external_energy_wgrav.impd.e
      at step 0 increment = 1
      nodal variables = displacement
      nodal variables = acceleration
      element variables = stress
      element variables = hourglass_energy
      global variables = artificial_energy
      global variables = external_energy
      global variables = internal_energy
      global variables = kinetic_energy
      global variables = strain_energy
      global variables = analyticEE
    end results output output_1

    begin history output history_1
      database name = external_energy_wgrav.impd.h
      at step 1 increment = 1
      ####   variable = nodal  acceleration
      variable = global artificial_energy
      variable = global external_energy
      variable = global internal_energy
      variable = global kinetic_energy
      variable = global strain_energy
      variable = global analyticEE
    end history output history_1

    begin user output
      compute global analyticEE as function analyticExternalEnergy
    end

    begin solution verification
      verify global external_energy = function analyticExternalEnergy
      relative tolerance = 0.005
      skip times = 0.0 to 0.01
      completion file = external_energy_wgrav.impd.verif
    end

    begin implicit dynamics
      alpha = 0.0
      beta = 0.25
      gamma = 0.5
    end

    begin solver
    end
```

```
    end adagio region region_1
  end adagio procedure procedure_1
end sierra external_energy_wgrav
```

B.34. HOURGLASS ENERGY FOR UNIFORM GRADIENT HEX ELEMENT WITH MIDPOINT INCREMENT FORMULATION

See Section 4.5 for problem description.

```
begin sierra hourglass_ughex_mi_stiff_1elem
  title hourglass energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin function expectedHourglassEnergy
    type is analytic
    evaluate expression = "4.61546e8*x*x*{reductionFactor}"
  end

  begin function upperHourglassEnergy
    type is analytic
    evaluate expression = "4.61546e8*x*x*{upperFactor}"
  end

  begin function lowerHourglassEnergy
    type is analytic
    evaluate expression = "4.61546e8*x*x*{lowerFactor}"
  end

  begin function disp
    type is piecewise linear
    begin values
      0.0  0.0
      1.0  0.01
    end values
  end function disp

####### Material Definition #########

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e+9 # GPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

####### BV Problem Definition #########

  begin solid section solid_1
    strain incrementation = midpoint_increment
  end solid section solid_1

  begin finite element model model_1
    database name = hourglass_ughex_mi_stiff.{meshSize}elem.g

    begin parameters for block block_1
      material = steel
      model = elastic
      section = solid_1
      hourglass stiffness = 0.05
      hourglass viscosity = 0.0
    end parameters for block block_1
```

```
  end finite element model model_1

  begin presto procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for presto region region_1
          user time step = 2.0e-5
          time step scale factor = 1.0
          time step increase factor = 1.0
          step interval = 1000
        end parameters for presto region region_1
      end time stepping block time_control_1
      termination time = 0.25
    end time control

    begin presto region region_1
      use finite element model model_1

######## definition of BCs ###########

      begin prescribed displacement
        node set = nodelist_1
        node set subroutine = hourglass_sideset
        scale factor = 0.2
      end prescribed displacement

      begin prescribed displacement
        node set = nodelist_2
        node set subroutine = hourglass_sideset
        scale factor = -0.2
      end prescribed displacement

######### output description ##############

      begin results output output_1
        database name = hourglass_ughex_mi_stiff.explicit.{meshSize}elem.e
        at step 0 increment = 1
        nodal variables = displacement
        element variables = stress
        nodal variables = hourglass_energy
        global variables = artificial_energy
        global variables = contact_energy
        global variables = external_energy
        global variables = hourglass_energy
        global variables = internal_energy
        global variables = kinetic_energy
        global variables = strain_energy
        global variables = expectedHGE
      end results output output_1

      begin history output history_1
        database name = hourglass_ughex_mi_stiff.explicit.{meshSize}elem.h
        at step 1 increment = 1
        variable = global artificial_energy
        variable = global contact_energy
        variable = global external_energy
        variable = global hourglass_energy
        variable = global internal_energy
        variable = global kinetic_energy
        variable = global strain_energy
        variable = global expectedHGE
        variable = global upperHGE
        variable = global lowerHGE
      end history output history_1
```

```
      begin user output
        compute global expectedHGE as function expectedHourglassEnergy
        compute global upperHGE as function upperHourglassEnergy
        compute global lowerHGE as function lowerHourglassEnergy
      end

      begin solution verification
        verify global hourglass_energy <= function upperHourglassEnergy
        verify global hourglass_energy >= function lowerHourglassEnergy
        skip times = 0.0 to 1.0e-3
        completion file = hourglass_ughex_mi_stiff.{meshSize}elem.verif
      end

    end presto region region_1

  end presto procedure procedure_1

end sierra hourglass_ughex_mi_stiff_1elem
```

B.35. HOURGLASS ENERGY FOR UNIFORM GRADIENT HEX ELEMENT WITH STRONGLY OBJECTIVE FORMULATION

See Section 4.6 for problem description.

```
begin sierra hourglass_ughex_so_stiff_1elem
  title hourglass energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  begin function disp
    type is piecewise linear
    begin values
      0.0  0.0
      1.0  0.01
    end values
  end function disp

####### Material Definition #########

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e+9 # GPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

####### BV Problem Definition #########

  begin solid section solid_1
      strain incrementation = strongly_objective
  end solid section solid_1

  begin finite element model model_1
    database name = hourglass_ughex_so_stiff_1elem.g

    begin parameters for block block_1
       material = steel
       model = elastic
       section = solid_1
       hourglass stiffness = 0.05
       hourglass viscosity = 0.0
    end parameters for block block_1

  end finite element model model_1

  begin presto procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for presto region region_1
        time step scale factor =  1.0
          step interval = 1000
        end parameters for presto region region_1
      end time stepping block time_control_1
      termination time = 0.25
    end time control
```

```
    begin presto region region_1
      use finite element model model_1

######## definition of BCs ###########

      begin prescribed displacement
        node set = nodelist_1
        node set subroutine = hourglass_sideset
        scale factor = 0.2
      end prescribed displacement

      begin prescribed displacement
        node set = nodelist_2
        node set subroutine = hourglass_sideset
        scale factor = -0.2
      end prescribed displacement

######### output description ##############

      begin results output output_1
        database name = hourglass_ughex_so_stiff_1elem.e
        at step 0 increment = 1
        nodal variables = displacement
        element variables = stress
        element variables = hourglass_energy
        global variables = artificial_energy
        global variables = contact_energy
        global variables = external_energy
        global variables = hourglass_energy
        global variables = internal_energy
        global variables = kinetic_energy
        global variables = strain_energy
      end results output output_1

      begin history output history_1
        database name = hourglass_ughex_so_stiff_1elem.h
        at step 1 increment = 1
        variable = global artificial_energy
        variable = global contact_energy
        variable = global external_energy
        variable = global hourglass_energy
        variable = global internal_energy
        variable = global kinetic_energy
        variable = global strain_energy
      end history output history_1

    end presto region region_1

  end presto procedure procedure_1

end sierra hourglass_ughex_so_stiff_1elem
```

B.36. HOURGLASS ENERGY WITH VISCOSITY CONTROL FOR UNIFORM GRADIENT HEX ELEMENT WITH STRONGLY OBJECTIVE FORMULATION

See Section 4.7 for problem description.

```
begin sierra hourglass_ughex_so_viscous_1elem
  title hourglass energy test

  user subroutine file = faces.F

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

####### Material Definition #########

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e+9 # GPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

####### BV Problem Definition #########

  begin solid section solid_1
      strain incrementation = strongly_objective
  end solid section solid_1

  begin finite element model model_1
    database name = hourglass_ughex_so_viscous_1elem.g

    begin parameters for block block_1
       material = steel
       model = elastic
       section = solid_1
       hourglass stiffness = 0.0
       hourglass viscosity = 0.002
    end parameters for block block_1

  end finite element model model_1

  begin presto procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for presto region region_1
        time step scale factor =  1.0
          step interval = 1000
        end parameters for presto region region_1
      end time stepping block time_control_1
      termination time = 0.01
    end time control

    begin presto region region_1
      use finite element model model_1

######## definition of BCs ###########
```

```
      begin prescribed displacement
        node set = nodelist_1
        node set subroutine = hourglass_sideset
        scale factor = 7.0
      end prescribed displacement

      begin prescribed displacement
        node set = nodelist_2
        node set subroutine = hourglass_sideset
        scale factor = -7.0
      end prescribed displacement

######### output description ##############

      begin results output output_1
        database name = hourglass_ughex_so_viscous_1elem.e
        at step 0 increment = 1
        nodal variables = displacement
        element variables = stress
        element variables = hourglass_energy
        global variables = artificial_energy
        global variables = contact_energy
        global variables = external_energy
        global variables = hourglass_energy
        global variables = internal_energy
        global variables = kinetic_energy
        global variables = strain_energy
      end results output output_1

      begin history output history_1
        database name = hourglass_ughex_so_viscous_1elem.h
        at step 1 increment = 1
        variable = global artificial_energy
        variable = global contact_energy
        variable = global external_energy
        variable = global hourglass_energy
        variable = global internal_energy
        variable = global kinetic_energy
        variable = global strain_energy
      end history output history_1

    end presto region region_1

  end presto procedure procedure_1

end sierra hourglass_ughex_so_viscous_1elem
```

B.37. INTERNAL ENERGY – EXPLICIT AND IMPLICIT DYNAMICS

See Section 4.8 for problem description.

```
############### internal_energy.explicit.i ##################
begin sierra internal_energy
  title internal energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  ## area = 1.0
  ## modulus 200000.0
  ## strain_rate = 0.0001
  ## strain = log(1-time*strain_rate)
  ## internal_energy = 0.5 * area * modulus * strain * strain

  begin function strainRate
    type is analytic
    evaluate expression = "0.0001*x"
  end

  begin function analyticInternalEnergy
    type is analytic
    evaluate expression = "0.5*1.0*200000.0*log(1.0-x*0.0001)*log(1.0-x*0.0001);"
  end

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e3 # MPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

  begin finite element model model_1
    database name = internal_energy.g

    begin block defaults
      material = steel
      model = elastic
    end

    begin parameters for block block_1
    end parameters for block block_1

  end finite element model model_1

  begin presto procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for presto region region_1
          user time step = 0.149
          time step scale factor = 1.0
          step interval = 1
        end parameters for presto region region_1
      end time stepping block time_control_1
      termination time = 100.0
    end time control
```

```
begin presto region region_1

  global energy reporting = exact

  use finite element model model_1

  begin fixed displacement
    node set = nodelist_1
    components = z
  end fixed displacement

  begin fixed displacement
    node set = nodelist_2
    components = x y
  end fixed displacement

  begin fixed displacement
    node set = nodelist_3
    components = y
  end fixed displacement

  begin prescribed displacement
    node set = nodelist_4
    direction = z
    function = strainRate
    scale factor = -1.0
  end prescribed displacement

  begin initial velocity
    node set = nodelist_4
    component = z
    magnitude = -0.0001
  end initial velocity

  begin results output output_1
    database name = internal_energy.explicit.e
    at step 0 increment = 1
    nodal variables = displacement
    nodal variables = velocity
    nodal variables = force_internal
    element variables = stress
    element variables = hourglass_energy
    global variables = contact_energy
    global variables = external_energy
    global variables = hourglass_energy
    global variables = internal_energy
    global variables = kinetic_energy
    global variables = strain_energy
    global variables = analyticIE
  end results output output_1

  begin history output history_1
    database name = internal_energy.explicit.h
    at step 1 increment = 1
    variable = global contact_energy
    variable = global external_energy
    variable = global hourglass_energy
    variable = global internal_energy
    variable = global kinetic_energy
    variable = global strain_energy
    variable = global analyticIE
  end history output history_1

  begin user output
    compute global analyticIE as function analyticInternalEnergy
  end

  begin solution verification
```

```
        verify global internal_energy = function analyticInternalEnergy
        relative tolerance = 0.10
        skip times = 0.0 to 1.1
        completion file = internal_energy.explicit.verif
      end

    end presto region region_1

  end presto procedure procedure_1

end sierra internal_energy
############### internal_energy.impd.i ##################
begin sierra internal_energy
  title internal energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  ## area = 1.0
  ## modulus 200000.0
  ## strain_rate = 0.0001
  ## strain = log(1-time*strain_rate)
  ## internal_energy = 0.5 * area * modulus * strain * strain

  begin function strainRate
    type is analytic
    evaluate expression = "0.0001*x"
  end

  begin function analyticInternalEnergy
    type is analytic
    evaluate expression = "0.5*1.0*200000.0*log(1.0-x*0.0001)*log(1.0-x*0.0001);"
  end

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e3 # MPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

  begin finite element model model_1
    database name = internal_energy.g

    begin block defaults
      material = steel
      model = elastic
    end

    begin parameters for block block_1
    end parameters for block block_1

  end finite element model model_1

  begin adagio procedure procedure_1

    begin time control
      begin time stepping block time_control_1
        start time =  0.0
        begin parameters for adagio region region_1
          time increment = 0.09
        end parameters for adagio region region_1
      end time stepping block time_control_1
      termination time = 100.0
```

```
end time control

begin adagio region region_1

  global energy reporting = exact

  use finite element model model_1

  begin fixed displacement
    node set = nodelist_1
    components = z
  end fixed displacement

  begin fixed displacement
    node set = nodelist_2
    components = x y
  end fixed displacement

  begin fixed displacement
    node set = nodelist_3
    components = y
  end fixed displacement

  begin prescribed displacement
    node set = nodelist_4
    direction = z
    function = strainRate
    scale factor = -1.0
  end prescribed displacement

  begin initial velocity
    node set = nodelist_4
    component = z
    magnitude = -0.0001
  end initial velocity

  begin results output output_1
    database name = internal_energy.impd.e
    at step 0 increment = 1
    nodal variables = displacement
    nodal variables = velocity
    nodal variables = force_internal
    element variables = stress
    element variables = hourglass_energy
    global variables = artificial_energy
    global variables = contact_energy
    global variables = external_energy
    global variables = hourglass_energy
    global variables = internal_energy
    global variables = kinetic_energy
    global variables = strain_energy
    global variables = analyticIE
  end results output output_1

  begin history output history_1
    database name = internal_energy.impd.h
    at step 1 increment = 1
    variable = global artificial_energy
    variable = global contact_energy
    variable = global external_energy
    variable = global hourglass_energy
    variable = global internal_energy
    variable = global kinetic_energy
    variable = global strain_energy
    variable = global analyticIE
  end history output history_1

  begin user output
```

```
        compute global analyticIE as function analyticInternalEnergy
      end

      begin solution verification
        verify global internal_energy = function analyticInternalEnergy
        relative tolerance = 0.0027
        skip times = 0.0 to 1.0
        completion file = internal_energy.impd.verif
      end

      begin implicit dynamics
        alpha = 0.0
        beta = 0.25
        gamma = 0.5
      end

      begin solver
      end

    end adagio region region_1

  end adagio procedure procedure_1

end sierra internal_energy
############### internal_energy.impd.default.i ##################
begin sierra internal_energy
  title internal energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  ## area = 1.0
  ## modulus 200000.0
  ## strain_rate = 0.0001
  ## strain = log(1-time*strain_rate)
  ## internal_energy = 0.5 * area * modulus * strain * strain

  begin function strainRate
    type is analytic
    evaluate expression = "0.0001*x"
  end

  begin function analyticInternalEnergy
    type is analytic
    evaluate expression = "0.5*1.0*200000.0*log(1.0-x*0.0001)*log(1.0-x*0.0001);"
  end

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e3 # MPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

  begin finite element model model_1
    database name = internal_energy.g

    begin block defaults
      material = steel
      model = elastic
    end

    begin parameters for block block_1
    end parameters for block block_1
```

```
end finite element model model_1

begin adagio procedure procedure_1

  begin time control
    begin time stepping block time_control_1
      start time =  0.0
      begin parameters for adagio region region_1
        time increment = 0.09
      end parameters for adagio region region_1
    end time stepping block time_control_1
    termination time = 100.0
  end time control

  begin adagio region region_1

    global energy reporting = exact

    use finite element model model_1

    begin fixed displacement
      node set = nodelist_1
      components = z
    end fixed displacement

    begin fixed displacement
      node set = nodelist_2
      components = x y
    end fixed displacement

    begin fixed displacement
      node set = nodelist_3
      components = y
    end fixed displacement

    begin prescribed displacement
      node set = nodelist_4
      direction = z
      function = strainRate
      scale factor = -1.0
    end prescribed displacement

    begin initial velocity
      node set = nodelist_4
      component = z
      magnitude = -0.0001
    end initial velocity

    begin results output output_1
      database name = internal_energy.impd.default.e
      at step 0 increment = 1
      nodal variables = displacement
      nodal variables = velocity
      nodal variables = force_internal
      element variables = stress
      element variables = hourglass_energy
      global variables = artificial_energy
      global variables = contact_energy
      global variables = external_energy
      global variables = hourglass_energy
      global variables = internal_energy
      global variables = kinetic_energy
      global variables = strain_energy
      global variables = analyticIE
    end results output output_1

    begin history output history_1
```

```
      database name = internal_energy.impd.default.h
      at step 1 increment = 1
      variable = global artificial_energy
      variable = global contact_energy
      variable = global external_energy
      variable = global hourglass_energy
      variable = global internal_energy
      variable = global kinetic_energy
      variable = global strain_energy
      variable = global analyticIE
    end history output history_1

    begin user output
      compute global analyticIE as function analyticInternalEnergy
    end

    begin solution verification
      verify global internal_energy = function analyticInternalEnergy
      relative tolerance = 0.0027
      skip times = 0.0 to 1.0
      completion file = internal_energy.impd.default.verif
    end

    begin implicit dynamics
    end

    begin solver
    end

  end adagio region region_1

end adagio procedure procedure_1

end sierra internal_energy
################ internal_energy.impd.approx.i ###################
begin sierra internal_energy
  title internal energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  ## area = 1.0
  ## modulus 200000.0
  ## strain_rate = 0.0001
  ## strain = log(1-time*strain_rate)
  ## internal_energy = 0.5 * area * modulus * strain * strain

  begin function strainRate
    type is analytic
    evaluate expression = "0.0001*x"
  end

  begin function analyticInternalEnergy
    type is analytic
    evaluate expression = "0.5*1.0*200000.0*log(1.0-x*0.0001)*log(1.0-x*0.0001);"
  end

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e3 # MPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel
```

```
begin finite element model model_1
  database name = internal_energy.g

  begin block defaults
    material = steel
    model = elastic
  end

  begin parameters for block block_1
  end parameters for block block_1

end finite element model model_1

begin adagio procedure procedure_1

  begin time control
    begin time stepping block time_control_1
      start time =  0.0
      begin parameters for adagio region region_1
        time increment = 0.09
      end parameters for adagio region region_1
    end time stepping block time_control_1
    termination time = 100.0
  end time control

  begin adagio region region_1

    global energy reporting = approximate

    use finite element model model_1

    begin fixed displacement
      node set = nodelist_1
      components = z
    end fixed displacement

    begin fixed displacement
      node set = nodelist_2
      components = x y
    end fixed displacement

    begin fixed displacement
      node set = nodelist_3
      components = y
    end fixed displacement

    begin prescribed displacement
      node set = nodelist_4
      direction = z
      function = strainRate
      scale factor = -1.0
    end prescribed displacement

    begin initial velocity
      node set = nodelist_4
      component = z
      magnitude = -0.0001
    end initial velocity

    begin results output output_1
      database name = internal_energy.impd.approx.e
      at step 0 increment = 1
      nodal variables = displacement
      nodal variables = velocity
      nodal variables = force_internal
      element variables = stress
      element variables = hourglass_energy
      global variables = artificial_energy
```

```
        global variables = contact_energy
        global variables = external_energy
        global variables = hourglass_energy
        global variables = internal_energy
        global variables = kinetic_energy
        global variables = strain_energy
        global variables = analyticIE
      end results output output_1

      begin history output history_1
        database name = internal_energy.impd.approx.h
        at step 1 increment = 1
        variable = global artificial_energy
        variable = global contact_energy
        variable = global external_energy
        variable = global hourglass_energy
        variable = global internal_energy
        variable = global kinetic_energy
        variable = global strain_energy
        variable = global analyticIE
      end history output history_1

      begin user output
        compute global analyticIE as function analyticInternalEnergy
      end

      begin solution verification
        verify global internal_energy = function analyticInternalEnergy
        relative tolerance = 0.0027
        skip times = 0.0 to 1.0
        completion file = internal_energy.impd.approx.verif
      end

      begin implicit dynamics
        alpha = 0.0
        beta = 0.25
        gamma = 0.5
      end

      begin solver
      end

    end adagio region region_1

  end adagio procedure procedure_1

end sierra internal_energy
################ internal_energy.impd.approx.default.i ##################
begin sierra internal_energy
  title internal energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  ## area = 1.0
  ## modulus 200000.0
  ## strain_rate = 0.0001
  ## strain = log(1-time*strain_rate)
  ## internal_energy = 0.5 * area * modulus * strain * strain

  begin function strainRate
    type is analytic
    evaluate expression = "0.0001*x"
  end

  begin function analyticInternalEnergy
    type is analytic
```

```
  evaluate expression = "0.5*1.0*200000.0*log(1.0-x*0.0001)*log(1.0-x*0.0001);"
end

begin material steel
  density = 7900.0 # kg/m^3

  begin parameters for model elastic
    youngs modulus  = 200.0e3 # MPa
    poissons ratio  = 0.3
  end parameters for model elastic

end material steel

begin finite element model model_1
  database name = internal_energy.g

  begin block defaults
    material = steel
    model = elastic
  end

  begin parameters for block block_1
  end parameters for block block_1
end finite element model model_1

begin adagio procedure procedure_1

  begin time control
    begin time stepping block time_control_1
      start time =  0.0
      begin parameters for adagio region region_1
        time increment = 0.09
      end parameters for adagio region region_1
    end time stepping block time_control_1
    termination time = 100.0
  end time control

  begin adagio region region_1

    global energy reporting = approximate

    use finite element model model_1

    begin fixed displacement
      node set = nodelist_1
      components = z
    end fixed displacement

    begin fixed displacement
      node set = nodelist_2
      components = x y
    end fixed displacement

    begin fixed displacement
      node set = nodelist_3
      components = y
    end fixed displacement

    begin prescribed displacement
      node set = nodelist_4
      direction = z
      function = strainRate
      scale factor = -1.0
    end prescribed displacement

    begin initial velocity
      node set = nodelist_4
      component = z
```

```
      magnitude = -0.0001
    end initial velocity

    begin results output output_1
      database name = internal_energy.impd.approx.default.e
      at step 0 increment = 1
      nodal variables = displacement
      nodal variables = velocity
      nodal variables = force_internal
      element variables = stress
      element variables = hourglass_energy
      global variables = artificial_energy
      global variables = contact_energy
      global variables = external_energy
      global variables = hourglass_energy
      global variables = internal_energy
      global variables = kinetic_energy
      global variables = strain_energy
      global variables = analyticIE
    end results output output_1

    begin history output history_1
      database name = internal_energy.impd.approx.default.h
      at step 1 increment = 1
      variable = global artificial_energy
      variable = global contact_energy
      variable = global external_energy
      variable = global hourglass_energy
      variable = global internal_energy
      variable = global kinetic_energy
      variable = global strain_energy
      variable = global analyticIE
    end history output history_1

    begin user output
      compute global analyticIE as function analyticInternalEnergy
    end

    begin solution verification
      verify global internal_energy = function analyticInternalEnergy
      relative tolerance = 0.0027
      skip times = 0.0 to 1.0
      completion file = internal_energy.impd.approx.default.verif
    end

    begin implicit dynamics
    end

    begin solver
    end

  end adagio region region_1

end adagio procedure procedure_1

end sierra internal_energy
```

B.38. INTERNAL (STRAIN) ENERGY – QUASISTATICS

See Section 4.9 for problem description.

```
############### strain_energy.qs.i ##################
begin sierra strain_energy
  title strain energy test

  define direction x with vector 1.0 0.0 0.0
  define direction y with vector 0.0 1.0 0.0
  define direction z with vector 0.0 0.0 1.0

  ## modulus 200000.0
  ## strain_rate = 0.0001 (ext) or 0.00001/2 (shear)
  ## strain = log(1-time*strain_rate)
  ## strain energy density = 0.5 * modulus * strain * strain

  begin function strain
    type is piecewise linear
    begin values
      0.0   0.0
      50.0  0.005
      100.0 0.0
      150.0 0.0005
      200.0 0.0
    end values
  end

  begin function analyticInternalEnergyExt
    type is piecewise analytic
    begin expressions
      0.0   "0.5 * 200000.0 * log(1.0 - (x        ) * 0.0001) * log(1.0 - (x
) * 0.0001)"
      50.0
"0.5 * 200000.0 * log(1.0 - (100.0 - x) * 0.0001) * log(1.0 - (100.0 - x) * 0.0001)"
      100.0 "0.0"
    end
  end

  begin function analyticInternalEnergyShr
    type is piecewise analytic
    begin expressions
      0.0   "0.0"
      100.0 "0.25 * 200000.0 / (1.0 + 0.3) * log(1.0 - (x - 100.0) * 0.00001) * log(1.0 - (x - 100.0) * 0.0000
      150.0 "0.25 * 200000.0 / (1.0 + 0.3) * log(1.0 - (200.0 - x) * 0.00001) * log(1.0 - (200.0 - x) * 0.0000
    end
  end

  begin material steel
    density = 7900.0 # kg/m^3

    begin parameters for model elastic
      youngs modulus  = 200.0e3 # MPa
      poissons ratio  = 0.3
    end parameters for model elastic

  end material steel

  begin finite element model model_1
    database name = strain_energy.g
    begin parameters for block block_1
      material = steel
      model = elastic
    end parameters for block block_1
  end finite element model model_1

  begin adagio procedure procedure_1
```

```
begin time control

  begin time stepping block time_control_1
    start time =  0.0
    begin parameters for adagio region region_1
      time increment = 5.26
    end parameters for adagio region region_1
  end time stepping block time_control_1

  begin time stepping block time_control_2
    start time =  100.0
    begin parameters for adagio region region_1
      time increment = 5.26
    end parameters for adagio region region_1
  end time stepping block time_control_2

  termination time = 150.0

end time control

begin adagio region region_1

  global energy reporting = exact

  use finite element model model_1

  begin fixed displacement
    node set = nodelist_1
    components = z
  end fixed displacement

  begin fixed displacement
    node set = nodelist_2
    components = x y
  end fixed displacement

  begin fixed displacement
    node set = nodelist_3
    components = y
  end fixed displacement

  begin prescribed displacement
    active periods = time_control_1
    node set = nodelist_4
    direction = z
    function = strain
    scale factor = -1.0
  end prescribed displacement

  begin prescribed displacement
    active periods = time_control_2
    node set = nodelist_4
    direction = y
    function = strain
    scale factor = 1.0
  end prescribed displacement

  begin fixed displacement
    active periods = time_control_2
    node set = nodelist_4
    components = x z
  end fixed displacement

  begin fixed displacement
    active periods = time_control_2
    node set = nodelist_1 nodelist_2 nodelist_3
    components = x y z
```

```
  end fixed displacement

  begin initial velocity
    node set = nodelist_4
    component = z
    magnitude = -0.0001
  end initial velocity

  begin results output output_1
    database name = strain_energy.qs.e
    at step 0 increment = 1
    nodal variables = displacement
    nodal variables = velocity
    nodal variables = force_internal
    element variables = stress
    element variables = strain
    element variables = hourglass_energy
    element variables = strain_energy
    global variables = artificial_energy
    global variables = contact_energy
    global variables = external_energy
    global variables = hourglass_energy
    global variables = internal_energy
    global variables = kinetic_energy
    global variables = strain_energy
    global variables = analyticIEExt
    global variables = analyticIEShr
  end results output output_1

  begin history output history_1
    database name = strain_energy.qs.h
    at step 1 increment = 1
    variable = global artificial_energy
    variable = global contact_energy
    variable = global external_energy
    variable = global hourglass_energy
    variable = global internal_energy
    variable = global kinetic_energy
    variable = global strain_energy
    variable = global analyticIEExt
    variable = global analyticIEShr
  end history output history_1

  begin user output
    compute at every step
    compute global analyticIEExt as function analyticInternalEnergyExt
    compute global analyticIEShr as function analyticInternalEnergyShr
#    compute global strainEnergy as sum of element strain_energy
#    compute global hgEnergy as sum of element hourglass_energy
  end

  begin solution verification
#    verify global strainEnergy = function analyticInternalEnergyExt
    verify global internal_energy = function analyticInternalEnergyExt
    relative tolerance = 0.0014
    skip times = 0.0 to 1.0
    skip times = 50.0 to 150.0
    completion file = strain_energy_ext.qs.verif
  end

  begin solution verification
#    verify global strainEnergy = function analyticInternalEnergyShr
    verify global internal_energy = function analyticInternalEnergyShr
    relative tolerance = 0.00051
    skip times = 0.0 to 101.0
    completion file = strain_energy_shr.qs.verif
  end
```

```
#      begin solution verification
#        verify global hgEnergy = 0
#        tolerance = 1.0e-12
#        skip times = 0.0 to 1.0
#        completion file = hg_energy.qs.verif
#      end

      begin solver
      end

    end adagio region region_1

  end adagio procedure procedure_1

end sierra strain_energy
```

B.39. KINETIC ENERGY

See Section 4.10 for problem description.

```
############### kinetic_energy.explicit.i ##################
begin sierra kinietic_energy

  begin function sine
    type is analytic
    evaluate expression is "amplitude=2;    \#
                            frequency=2*pi; \#
                            phase=0;        \#
                            amplitude * sin(frequency*x + phase)"
  end

  begin function analyticKE
    type is analytic
    # KE = 0.5 * M * V^2
    # M = p * V
    evaluate expression is " 0.5 * (100.0 * 1.0) *
( 2.0 * sin(2.0*pi*x) * 2.0 * sin(2.0*pi*x) ) "
  end

  begin function zero
    type is analytic
    evaluate expression is "0.0"
  end

  begin material flubber
    density = 100.0
    begin parameters for model elastic
      youngs modulus = 30.0e6
      poissons ratio = 0.3
    end parameters for model elastic
  end material flubber

  #{ Ax = 1.0 }
  #{ Ay = 1.0 }
  #{ Az = 1.0 }
  define direction offaxis with vector {Ax} {Ay} {Az}
  # offaxis crossed with 1 0 0
  #{ Bx =  0.0 }
  #{ By =  1.0 }
  #{ Bz = -1.0 }
  define direction perpOne with vector {Bx} {By} {Bz}
  # A cross B
  #{ Cx =  Ay*Bz - Az*By }
  #{ Cy =  Az*Bx - Ax*Bz }
  #{ Cz =  Ax*By - Ay*Bx }
  define direction perpTwo with vector {Cx} {Cy} {Cz}

  begin finite element model kinietic_energy1
    database name = kinetic_energy.g
    database type = exodusII

    begin parameters for block block_1
      material = flubber
      model = elastic
    end

  end finite element model kinietic_energy1

  begin adagio procedure rigidBody

    begin time control
      begin time stepping block p1
        start time = 0.0
```

```
      begin parameters for presto region adagio
        time step scale factor = 1.0
        user time step = 2.24e-2
        step interval = 1
      end
    end time stepping block p1
    termination time = 1.0
  end time control

  begin adagio region adagio
    use finite element model kinietic_energy1

    begin user output
      include all blocks
      compute global analyticKE as function analyticKE
      compute at every step
    end

    begin solution verification
      completion file = VerifGlobalInternal
      verify global kinetic_energy = function analyticKE
      relative tolerance = 0.005
      ## Skip crossings of zero
      skip times = 0.000 to 0.001
      skip times = 0.499 to 0.501
      skip times = 0.999 to 1.001
    end

    begin history output frederica
      database name = kinetic_energy.explicit.h
      database type = exodusII
      at step 0, increment = 1
      variable = global kinetic_energy as KineticEnergy
      variable = global analyticKE
    end history output frederica

    begin prescribed velocity
      block = block_1
      direction = offaxis
      function = sine
    end

    begin prescribed velocity
      block = block_1
      direction = perpOne
      function = zero
    end

    begin prescribed velocity
      block = block_1
      direction = perpTwo
      function = zero
    end

  end adagio region adagio
 end adagio procedure rigidBody
end sierra kinietic_energy
############### kinetic_energy.impd.i ##################
begin sierra kinietic_energy

 begin function sine
   type is analytic
   evaluate expression is "amplitude=2;    \#
                           frequency=2*pi; \#
                           phase=0;        \#
                           amplitude * sin(frequency*x + phase)"
 end
```

```
  begin function analyticKE
    type is analytic
    # KE = 0.5 * M * V^2
    # M = p * V
    evaluate expression is " 0.5 * (100.0 * 1.0) *
( 2.0 * sin(2.0*pi*x) * 2.0 * sin(2.0*pi*x) ) "
  end

  begin function zero
    type is analytic
    evaluate expression is "0.0"
  end

  begin material flubber
    density = 100.0
    begin parameters for model elastic
      youngs modulus = 30.0e6
      poissons ratio = 0.3
    end parameters for model elastic
  end material flubber

  #{ Ax = 1.0 }
  #{ Ay = 1.0 }
  #{ Az = 1.0 }
  define direction offaxis with vector {Ax} {Ay} {Az}
  # offaxis crossed with 1 0 0
  #{ Bx =  0.0 }
  #{ By =  1.0 }
  #{ Bz = -1.0 }
  define direction perpOne with vector {Bx} {By} {Bz}
  # A cross B
  #{ Cx =  Ay*Bz - Az*By }
  #{ Cy =  Az*Bx - Ax*Bz }
  #{ Cz =  Ax*By - Ay*Bx }
  define direction perpTwo with vector {Cx} {Cy} {Cz}

  begin finite element model kinietic_energy1
    database name = kinetic_energy.g
    database type = exodusII

    begin parameters for block block_1
      material = flubber
      model = elastic
    end

  end finite element model kinietic_energy1

  begin adagio procedure fred

    begin time control
      begin time stepping block p1
        start time = 0.0
        begin parameters for adagio region adagio
          time increment = 0.005
        end
      end
      termination time = 1.0
    end

    begin adagio region adagio
      use finite element model kinietic_energy1

      begin user output
        include all blocks
        compute global analyticKE as function analyticKE
        compute at every step
      end
```

```
      begin solution verification
        completion file = VerifGlobalInternal
        verify global kinetic_energy = function analyticKE
        relative tolerance = 0.005
        ## Skip crossings of zero
        skip times = 0.000 to 0.001
        skip times = 0.499 to 0.501
        skip times = 0.999 to 1.001
      end

      begin history output fred
        database name = kinetic_energy.impd.h
        database type = exodusII
        at step 0, increment = 1
        variable = global kinetic_energy as KineticEnergy
        variable = global analyticKE
      end

      begin prescribed velocity
        block = block_1
        direction = offaxis
        function = sine
      end

      begin prescribed velocity
        block = block_1
        direction = perpOne
        function = zero
      end

      begin prescribed velocity
        block = block_1
        direction = perpTwo
        function = zero
      end

      begin implicit dynamics
        alpha = 0.0
        beta = 0.25
        gamma = 0.5
      end

      begin solver
        begin cg
          reference = belytschko
          target residual = 1.0e-8
        end
      end

    end
  end
end
```

DISTRIBUTION

Email—Internal [redacted]

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.