SAND2024-08021

SANDIA REPORT
Unlimited Release
Printed June 26, 2024

Sierra/Aria Verification Manual – Version 5.20

Brian R. Carnes

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 portion of the tests that exist in the Sierra Thermal/Fluids verification test suite. Each of these tests is run nightly with the Sierra/TF code suite and the results of the test checked under mesh refinement against the correct analytic result. For each of the tests presented in this document the test setup, derivation of the analytic solution, and comparison of the code results to the analytic solution is provided. This document can be used to confirm that a given code capability is verified or referenced as a compilation of example problems.

ACKNOWLEDGMENTS

This document's authors acknowledge the help of the Sierra/TF team in creating and maintaining these verification tests.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The Sierra/TF Verification Manual is divided into chapters based on related capabilities. Each section of a chapter represents a distinct verification test. Some problems that are not yet fully documented are listed at the end of each chapter.

All of these verification tests are run nightly by the development team to continually verify code accuracy under mesh refinement. The graphics and charts in this document are automatically generated by the nightly test runs.

The test files for these problems may be found in the Sierra regression test repository. Most are in the sub-directory called "verification."

```
aria_rtest/verification
```

All tests are assigned the keyword "verification". Those that appear in this document also have the keyword "self-documenting".

For each test, the approximate finite element solution T_h is compared to the exact solution T using several global norms, and in some cases using response quantities of interest. This is repeated over a series of uniformly refined meshes (not necessarily nested) with mesh sizes $\{h_i\}$, giving a sequence of errors $\{E_i\}$. For each pair of meshes, a convergence rate is estimated using the formula

$$r_i \equiv \log(E_i/E_{i-1})/\log(h_i/h_{i-1}). \tag{1.1}$$

The convergence of r_i to the expected rate is monitored as the mesh is refined. A test passes if all of the estimated convergence rates on the finest pair of meshes are within a given tolerance of the expected rates.

This page intentionally left blank.

2. BASIC THERMAL TESTS

2.1. STEADY HEAT CONDUCTION: HEX8 MESHES

This problem tests basic steady state heat conduction in a 3D domain. The geometry consists of a unit cube. A variety of different source terms and boundary conditions are simultaneously applied. The exact solution is a manufactured solution.

2.1.1. Features Tested

Basic heat conduction on Hex8 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.1.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine).

2.1.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

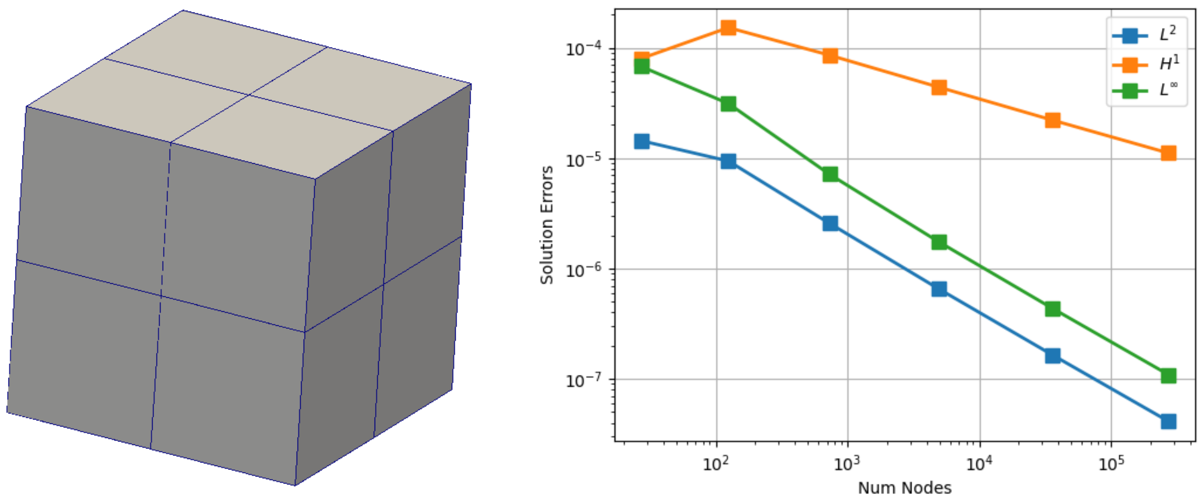
2.1.4. Verification of Solution

A manufactured solution is chosen as

$$T(x, y, z) = 1 + (x - x^2)^2(y - y^2)^2(z - z^2)^2.$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).



Coarse Mesh   Error Norms

Figure 2.1-1.. Steady Heat Conduction: Hex8 Meshes

Table 2.1-1.. Steady Heat Conduction: Convergence Rates for Hex8 Meshes

Num Dofs	L^2	H^1	L^∞
125	0.83	-1.27	1.52
729	2.20	0.98	2.51
4913	2.15	1.05	2.21
35937	2.08	1.04	2.10
274625	2.04	1.02	2.05

For input decks see Appendix 12.1.1.

2.2. STEADY HEAT CONDUCTION: HEX20 MESHES

This problem tests basic steady state heat conduction in a 3D domain. The geometry consists of a unit cube. A variety of different source terms and boundary conditions are simultaneously applied. The exact solution is a manufactured solution.

2.2.1. Features Tested

Basic heat conduction on Hex20 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.2.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine).

2.2.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

2.2.4. Verification of Solution

A manufactured solution is chosen as

$$T(x, y, z) = 1 + (x - x^2)^2(y - y^2)^2(z - z^2)^2.$$

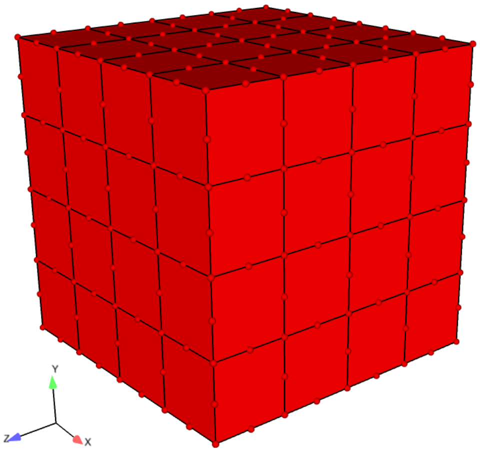
The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

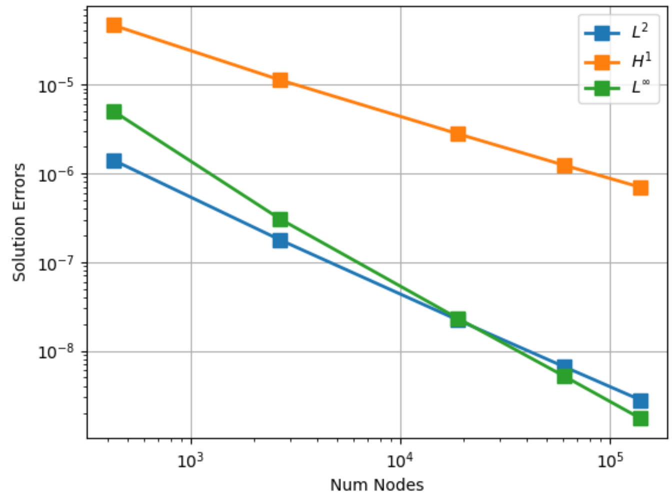
Table 2.2-1.. Steady Heat Conduction: Convergence Rates for Hex20 Meshes

Num Dofs	L^2	H^1	L^∞
2673	3.39	2.32	4.58
18785	3.19	2.15	3.95
60625	3.11	2.08	3.85
140481	3.08	2.06	3.91

For input decks see Appendix 12.1.2.



Coarse Mesh



Error Norms

Figure 2.2-1.. Steady Heat Conduction: Hex20 Meshes

2.3. STEADY HEAT CONDUCTION: HEX27 MESHES

This problem tests basic steady state heat conduction in a 3D domain. The geometry consists of a unit cube. A variety of different source terms and boundary conditions are simultaneously applied. The exact solution is a manufactured solution.

2.3.1. Features Tested

Basic heat conduction on Hex27 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.3.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine).

2.3.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

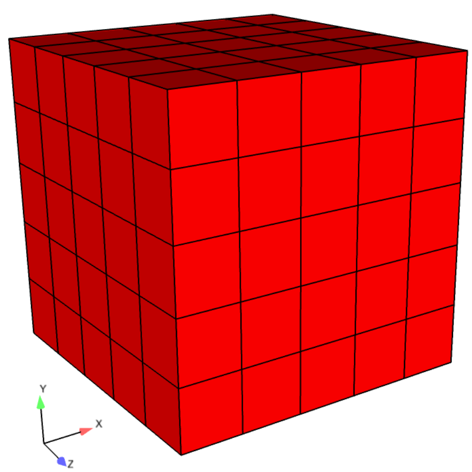
2.3.4. Verification of Solution

A manufactured solution is chosen as

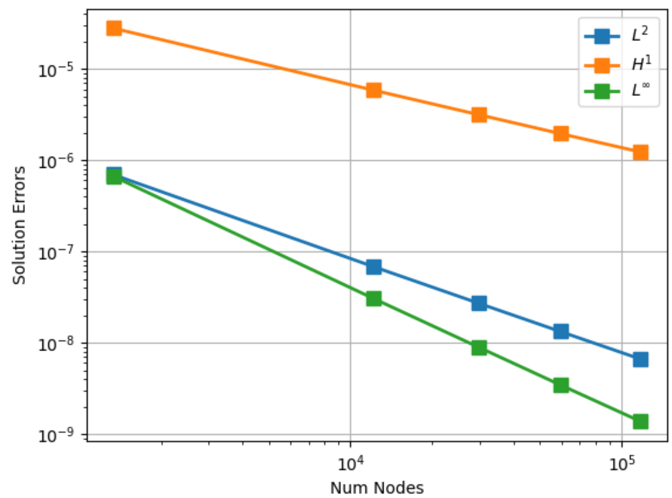
$$T(x, y, z) = 1 + (x - x^2)^2(y - y^2)^2(z - z^2)^2.$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).



Coarse Mesh



Error Norms

Figure 2.3-1.. Steady Heat Conduction: Hex27 Meshes

Table 2.3-1.. Steady Heat Conduction: Convergence Rates for Hex27 Meshes

Num Dofs	L^2	H^1	L^∞
12167	3.15	2.12	4.18
29791	3.10	2.07	4.13
59319	3.08	2.06	4.10
117649	3.06	2.05	4.08

For input decks see Appendix 12.1.3.

2.4. STEADY HEAT CONDUCTION: TET4 MESHES

This problem is identical to the one in Section 2.1 except that unstructured Tet4 meshes are used instead. The meshes are obtained from Cubit.

2.4.1. Features Tested

Basic heat conduction on Tet4 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.4.2. Boundary Conditions

Same as in Section 2.1.

2.4.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

2.4.4. Verification of Solution

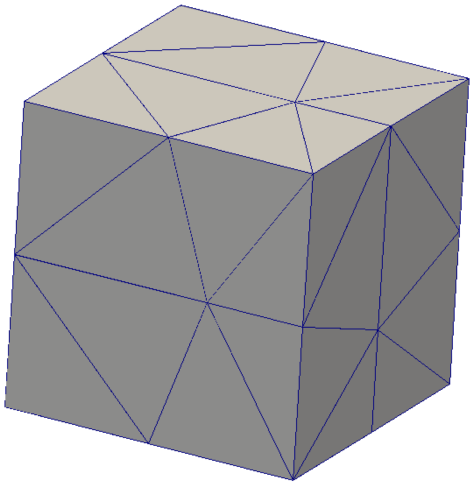
Same as in Section 2.1.

Unlike the Hex8 case, we have observed in many cases, and in this test, that the convergence rate for the temperature in the L^∞ norm is somewhat less than 2, in this case about 1.9. The exact reason for this behavior is unclear.

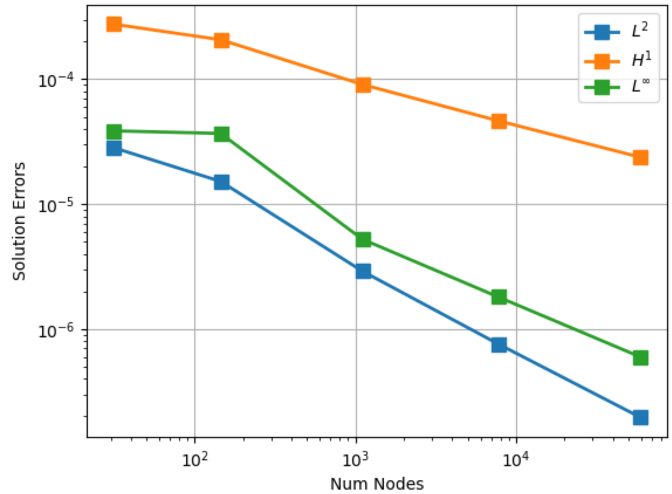
Table 2.4-1.. Steady Heat Conduction: Convergence Rates for Tet4 Meshes

Num Dofs	L^2	H^1	L^∞
145	1.21	0.56	0.10
1104	2.45	1.22	2.88
7725	2.07	1.03	1.63
59636	1.99	0.99	1.63

For input decks see Appendix 12.1.4.



Coarse Mesh



Error Norms

Figure 2.4-1.. Steady Heat Conduction: Tet4 Meshes

2.5. STEADY HEAT CONDUCTION: TET4TET10 MESHES

This problem is identical to the one in Section 2.1 with the exception of constant thermal conductivity and use of unstructured Tet10 meshes. The meshes are obtained from Cubit.

2.5.1. Features Tested

Basic heat conduction with Tet4 solution on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.5.2. Boundary Conditions

Same as in Section 2.1.

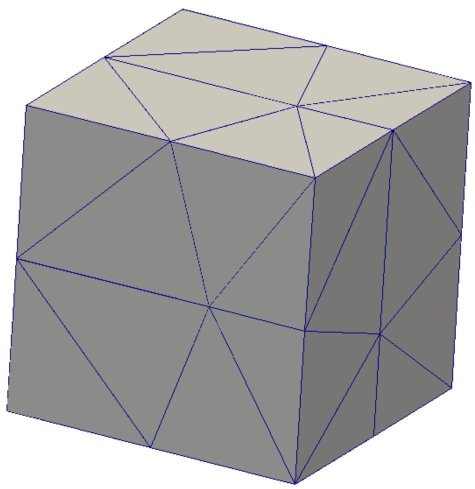
2.5.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

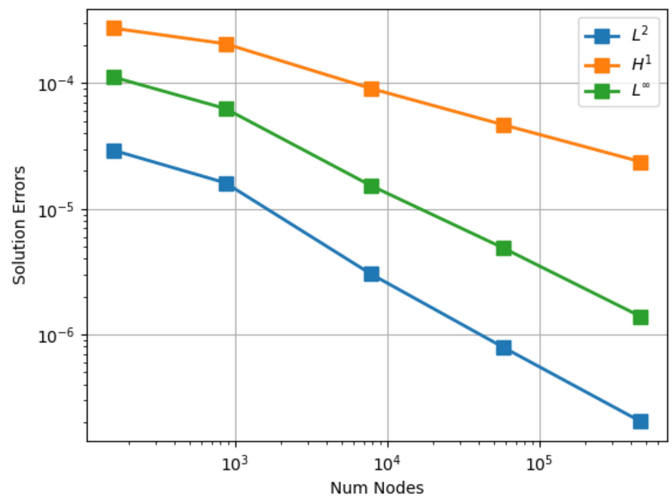
2.5.4. Verification of Solution

Same as in Section 2.1.

Unlike the Hex8 case, we have observed in many cases, and in this test, that the convergence rate for the temperature in the L^∞ norm is somewhat less than 2, in this case about 1.9. The exact reason for this behavior is unclear.



Coarse Mesh



Error Norms

Figure 2.5-1.. Steady Heat Conduction: Tet4 Solutions on Tet10 Meshes

Table 2.5-1.. Steady Heat Conduction: Convergence Rates for Tet4Tet10 Meshes

Num Dofs	L^2	H^1	L^∞
865	1.06	0.50	1.02
7831	2.27	1.12	1.92
58211	2.01	1.00	1.71
464414	1.96	0.98	1.81

For input decks see Appendix 12.1.5.

2.6. STEADY HEAT CONDUCTION: TET10 MESHES

This problem is identical to the one in Section 2.1 except that unstructured Tet10 meshes are used instead. The meshes are obtained from Cubit.

2.6.1. Features Tested

Basic heat conduction on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.6.2. Boundary Conditions

Same as in Section 2.1.

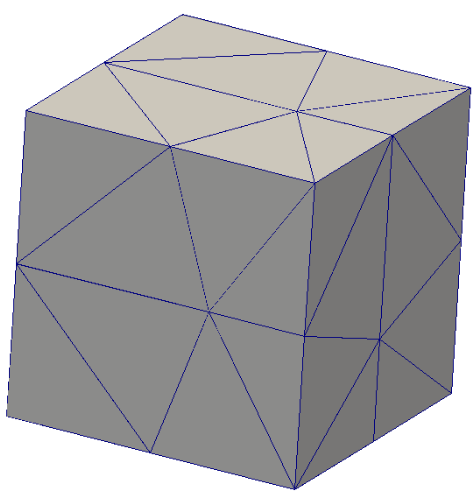
2.6.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

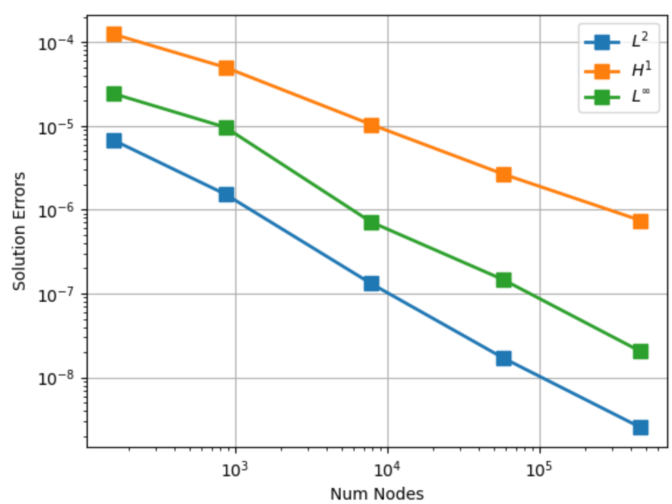
2.6.4. Verification of Solution

Same as in Section 2.1.

Unlike the Hex8 case, we have observed in many cases, and in this test, that the convergence rate for the temperature in the L^∞ norm is somewhat less than 3, in this case about 2.7. The exact reason for this behavior is unclear.



Coarse Mesh



Error Norms

Figure 2.6-1.. Steady Heat Conduction: Tet10 Meshes

For input decks see Appendix 12.1.6.

Table 2.6-1.. Steady Heat Conduction: Convergence Rates for Tet10 Meshes

Num Dofs	L^2	H^1	L^∞
865	2.63	1.61	1.64
7831	3.34	2.13	3.54
58211	3.06	2.05	2.38
464414	2.75	1.84	2.84

2.7. TRANSIENT HEAT CONDUCTION: HEX8 MESHES

This problem tests basic transient heat conduction in a 3D domain. The geometry consists of a unit cube.

2.7.1. Features Tested

Basic transient heat conduction on Hex8 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.7.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine).

2.7.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

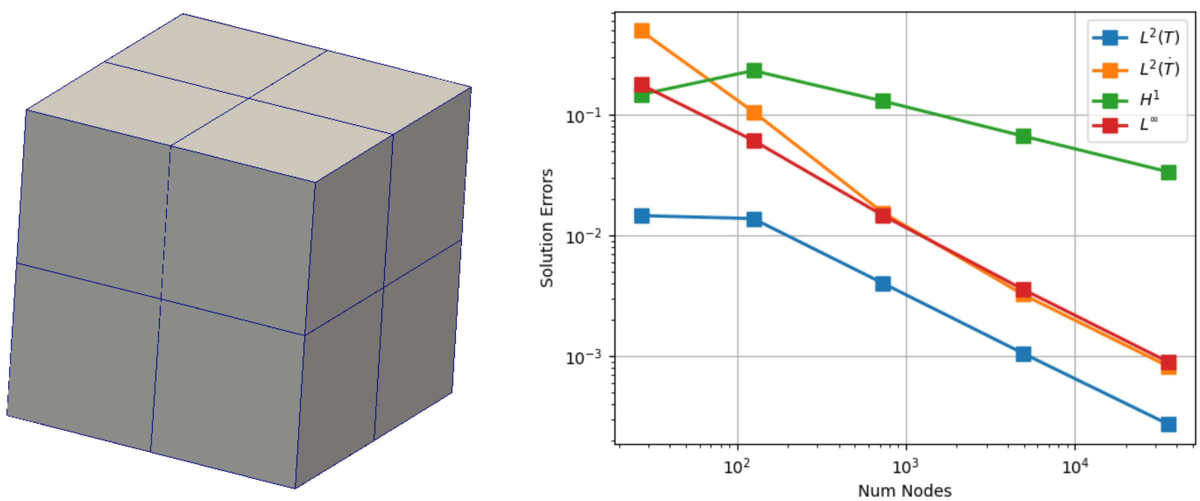
2.7.4. Verification of Solution

A manufactured solution is chosen as

$$\begin{aligned} T(x,y,z,t) &= (x-x^2)^2\,(y-y^2)^2\,(z-z^2)^2 m(t) + 1, \\ m(t) &= 10^4\,(1-\exp(-t) + t\,\exp(-(t-1)^2)) \end{aligned}$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution at final time are computed in the L^2 norm of T and $\dot{T}$, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, 2 and 1, respectively (within a tolerance).



Coarse Mesh | Error Norms

Figure 2.7-1.. Transient Heat Conduction: Hex8 Meshes

Table 2.7-1.. Transient Heat Conduction: Convergence Rates for Hex8 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
125	0.12	3.02	-0.89	2.08
729	2.09	3.28	0.98	2.42
4913	2.09	2.46	1.05	2.22
35937	2.06	2.07	1.04	2.10

For input decks see Appendix 12.1.7.

2.8. TRANSIENT HEAT CONDUCTION: TET4 MESHES

This problem tests basic transient heat conduction in a 3D domain as in Section 2.7. The geometry consists of a unit cube and a single bulk fluid element.

2.8.1. Features Tested

Basic transient heat conduction on Tet4 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; bulk fluid element; heat flux and source term from Encore user subroutines.

2.8.2. Boundary Conditions

Identical to Section 2.7 except one convective flux boundary condition is now connected to a bulk fluid element.

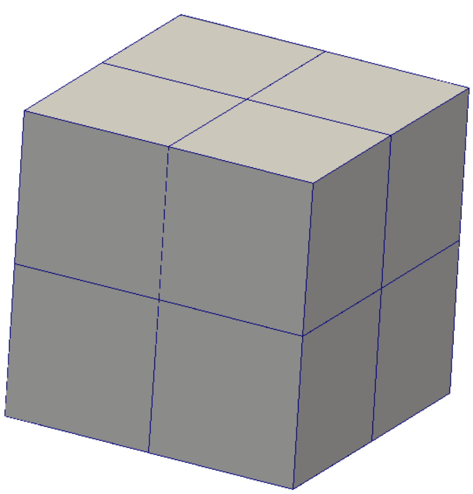
2.8.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

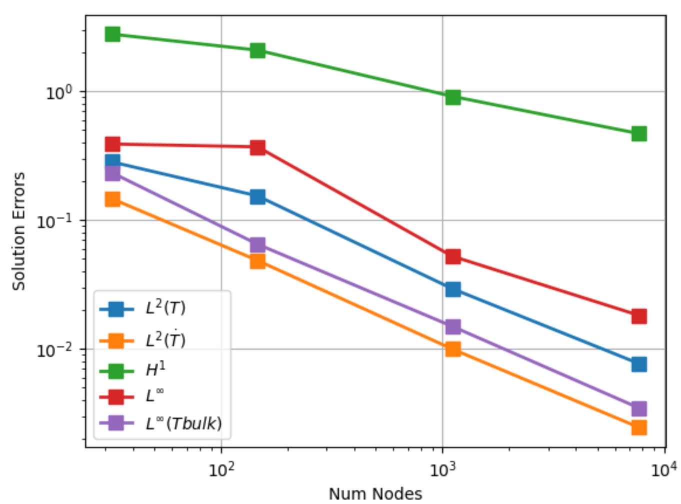
2.8.4. Verification of Solution

A manufactured solution is chosen as in Section 2.7.

For each mesh, the errors in the temperature solution at final time are computed in the L^2 norm of T and $\dot{T}$, L^∞ and H^1 norms. As in Section 2.4, we see convergence rates for L^∞ that are slightly less than 2.



Coarse Mesh



Error Norms

Figure 2.8-1.. Transient Heat Conduction: Tet4 Meshes

For input decks see Appendix 12.1.8.

Table 2.8-1.. Transient Heat Conduction: Convergence Rates for Tet4 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞	$L^\infty(Tbulk)$
146	1.20	2.19	0.57	0.10	2.53
1105	2.45	2.34	1.22	2.89	2.17
7726	2.07	2.16	1.03	1.64	2.26

2.9. TRANSIENT HEAT CONDUCTION: TET4TET10 MESHES

This problem tests basic transient heat conduction in a 3D domain as in Section 2.8. The geometry consists of a unit cube.

2.9.1. Features Tested

Basic transient heat conduction Tet4 analysis on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.9.2. Boundary Conditions

Identical to Section 2.8

2.9.3. Material Parameters

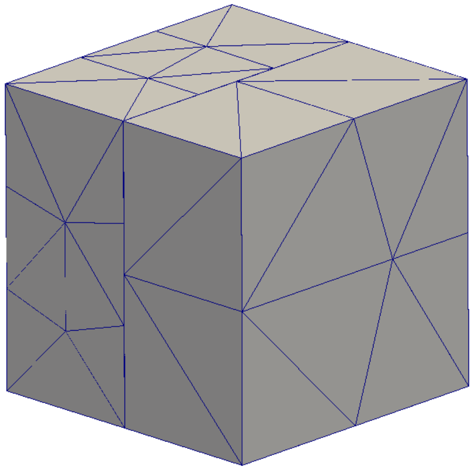
The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

2.9.4. Verification of Solution

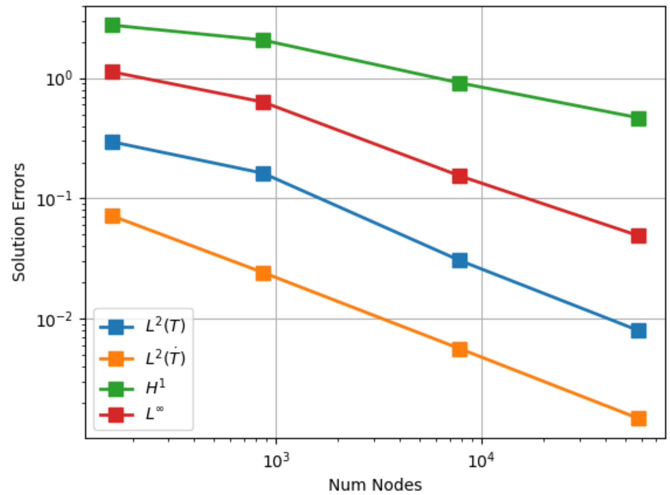
A manufactured solution is chosen as in Section 2.8.

For each mesh, the errors in the temperature solution at final time are computed in the L^2 norm of T and $\dot{T}$, L^∞ and H^1 norms. As in Section 2.8, we see convergence rates for L^∞ that are slightly less than 2.

For input decks see Appendix 12.1.9.



Coarse Mesh



Error Norms

Figure 2.9-1.. Transient Heat Conduction: Tet4 Solution on Tet10 Meshes

Table 2.9-1.. Transient Heat Conduction: Convergence Rates for Tet4 Solution on Tet10 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
865	1.06	1.93	0.50	1.02
7831	2.27	1.98	1.12	1.92
58211	2.01	2.00	1.00	1.72

2.10. TRANSIENT HEAT CONDUCTION: TET10 MESHES

This problem tests basic transient heat conduction in a 3D domain as in Section 2.7. The geometry consists of a unit cube.

2.10.1. Features Tested

Basic transient heat conduction on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

2.10.2. Boundary Conditions

Identical to Section 2.7

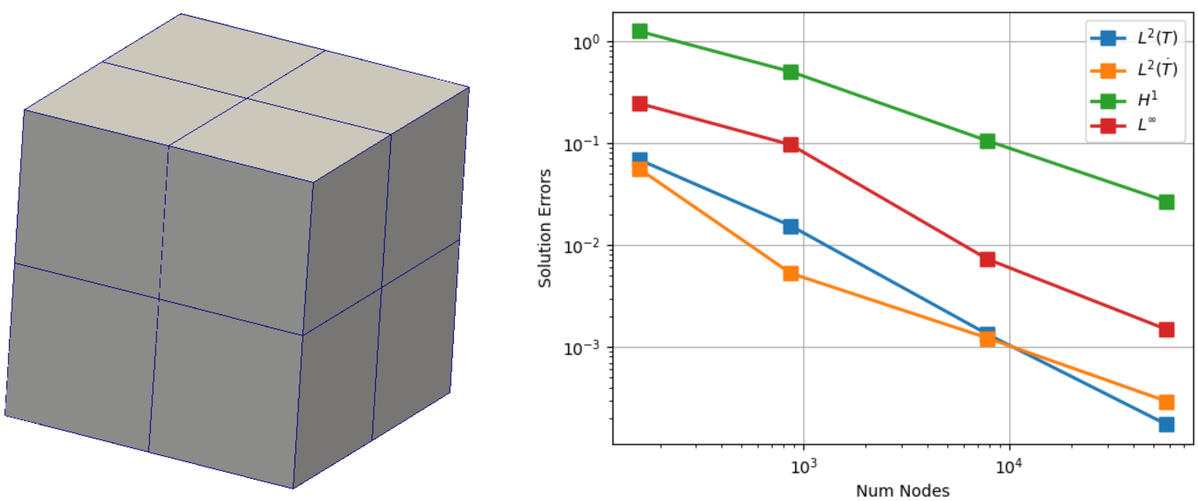
2.10.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

2.10.4. Verification of Solution

A manufactured solution is chosen as in Section 2.7.

For each mesh, the errors in the temperature solution at final time are computed in the L^2 norm of T and $\dot{T}$, L^∞ and H^1 norms. As in Section 2.6, we see convergence rates for L^∞ that are slightly less than 2.



Coarse Mesh Error Norms

Figure 2.10-1.. Transient Heat Conduction: Tet10 Meshes

Table 2.10-1.. Transient Heat Conduction: Convergence Rates for Tet10 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
865	2.64	4.17	1.61	1.65
7831	3.33	1.99	2.13	3.52
58211	3.06	2.14	2.05	2.38

For input decks see Appendix 12.1.10.

2.11. POSTPROCESS MIN/MAX

2.11.1. Problem Description

This problem tests the min/max postprocessors in Aria.

2.11.2. Features Tested

min max postprocessors

2.11.3. Boundary Conditions

Dirichlet BCs are specified using the exact solution on surfaces 1-4.

A source term is applied within all blocks based on substituting the exact solution into the heat conduction operator.

2.11.4. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant.

2.11.5. Verification of Solution

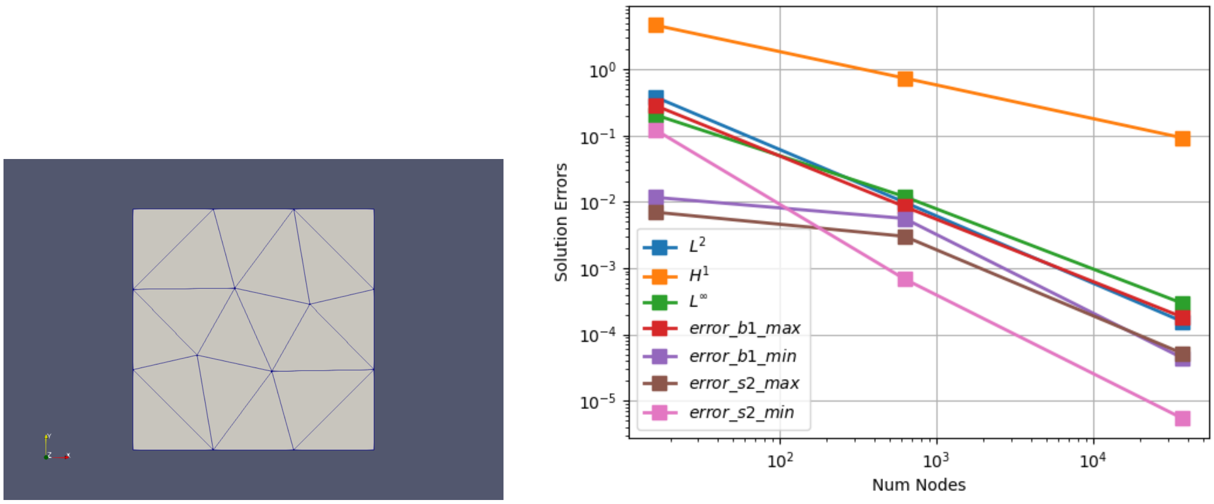
The manufactured solution is

$$\sin(7x)\sin(8y).$$

For each uniformly refined mesh, the errors in the temperature solution are computed in the L^2, H^1, and L^∞ norms and for various postprocessors. Additionally, the nodal maximum and minimum values on both block 1 and surface 2 are computed using Encore postprocessors and the convergence of these values is compared as well. Since the maximum and minimums are nodal, the location of the nodes will reflect the max/min values produced for a given mesh. Provided that the mesh is uniformly refined (without smoothing that may shift the nodal locations), every mesh refinement will produce a better result, dependent on how much closer to the maximum/minimum true solution the new nodes are.

Table 2.11-1.. Min Max Postprocess: Convergence Rates

Num Dofs	L^2	H^1	L^∞	$error\_b1\_max$	$error\_b1\_min$	$error\_s2\_max$	$error\_s2\_min$
625	2.00	1.00	1.55	1.92	0.40	0.46	2.83
37249	2.03	1.02	1.81	1.88	2.37	1.99	2.37



Coarse Mesh Error Norms

Figure 2.11-1.. Min Max Postprocess

2.12. ADAPTIVITY

This problem is identical to the one in Section 2.4 except that we use adaptive mesh refinement to refine from a coarse base mesh obtained from Cubit.

2.12.1. Features Tested

Basic heat conduction on Tet4 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines; adaptive mesh refinement; local error indicators based on jump in heat flux.

2.12.2. Boundary Conditions

Same as in Section 2.1.

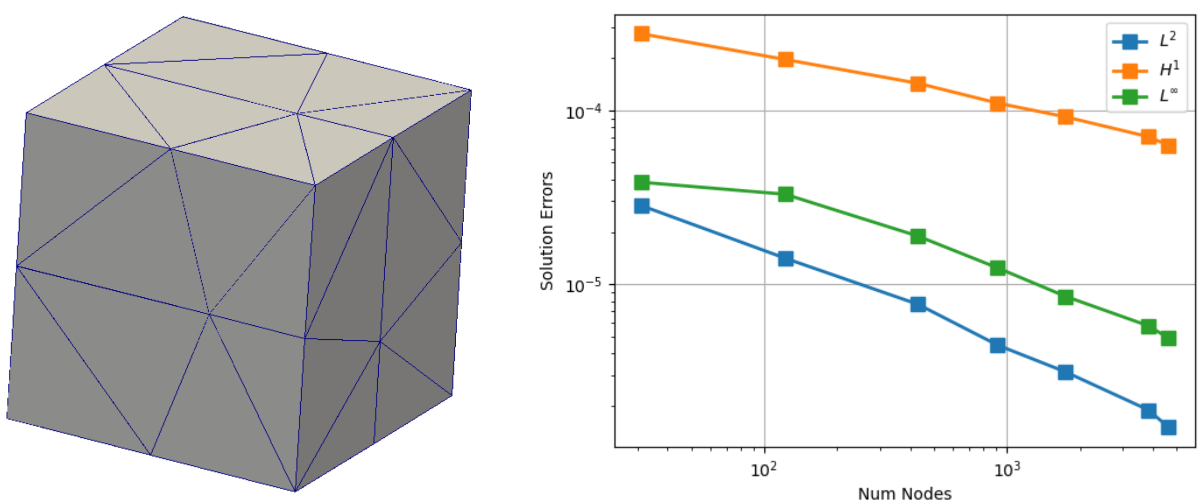
2.12.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

2.12.4. Verification of Solution

The mesh is adapted using code from Sierra/Percept that refines tetrahedral meshes without any hanging nodes (conformal meshes only). The element error indicator is computed using a residual-based error indicator in Encore, that computes the integrated jump in the normal heat flux across inter-element faces. The input file is configured to refine elements so that the sum of the error in the refined elements is approximately 75% of the total error in all elements.

Because of variability in the meshes, we expect the error reduction to be noisy. In this case, we use linear least squares to estimate the slope of the error on a log-log plot against mesh size. Since the solution is smooth we also expect the meshes to eventually refine everywhere. We estimate convergence in the usual error norms and observe rates close to the theoretical ones (second order convergence for the L^2 and L^∞ norms and first order convergence for the H^1 norm). Mesh size is estimated using the formula $h \approx N^{-1/3}$, where N is the number of nodes in the mesh.



Coarse Mesh

Error Norms

Figure 2.12-1.. Steady Heat Conduction: Tet4 Meshes (Adaptive Mesh Refinement)

Documentation for the following tests is in progress:

```
nlin_verify1/1dnonlin_verify1.test|np8
o_2d/aniso_2d.test|np8
o_3d/aniso_3d.test|np8
shell_2d/cyl_shell_2d.test|np8
shell_3d/cyl_shell_3d.test|np8
in_C_fi/nonlin_C_fi.test|np1
in_C_trap/nonlin_C_trap.test|np1
ce_parab/source_parab.test|np1
ce_parab_2d/source_parab_2d.test|np1
```

```
shell_axi/sph_shell_axi.test|np1
rical_shell/spherical_shell.test|np4
11_nonlin/x11b11_nonlin.test|np1
```

This page intentionally left blank.

3. THERMAL BOUNDARY CONDITIONS

3.1. RADIATIVE HEAT FLUX

This problem tests the radiative flux boundary condition under steady state heat conduction in a 2D domain. The geometry consists of a unit square.

3.1.1. Features Tested

Basic heat conduction on Quad4 meshes; radiative flux boundary conditions with constant emissivity and reference temperature; radiation form factor from C-style user subroutine; temperature boundary conditions from C-style user subroutine and constant values.

3.1.2. Boundary Conditions

At surface 3, the temperature is prescribed from a C-style user subroutine. On surfaces 2 and 4, a constant temperatre boundary condition is used. On surface 1, a radiative heat flux condition is prescribed. No source term is needed.

3.1.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant.

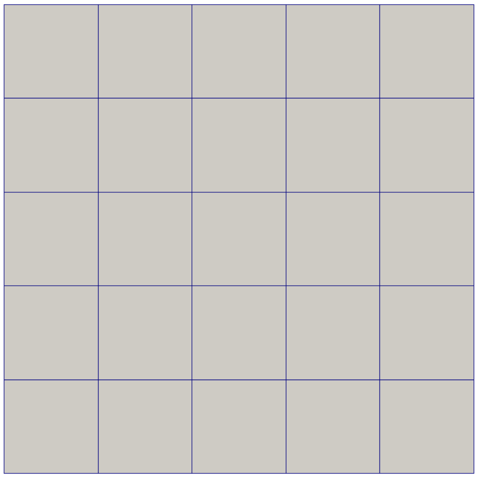
3.1.4. Verification of Solution

A manufactured solution is chosen as

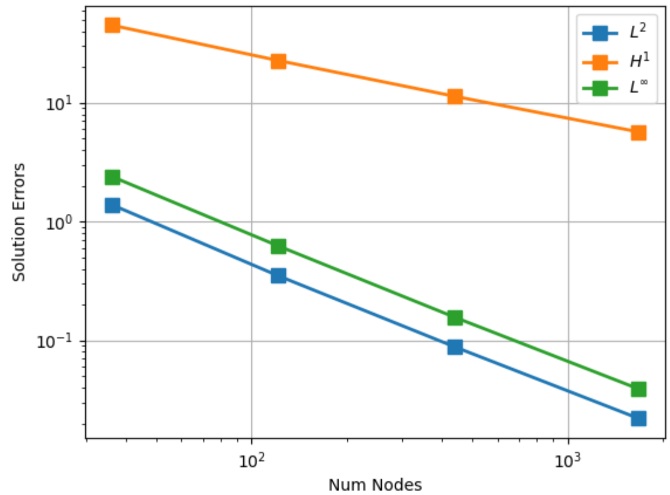
$$T(x, y) = 200\ \exp(-\pi y)\ \sin(\pi x) + 600$$

For each mesh, the errors in the temperature solution are computed in the L^2, H^1 and L^∞ norms. The test passes, only if the observed rates of convergence in these norms are 2, 1, and 2, respectively (within a tolerance).

For input decks see Appendix 12.2.1.



Coarse Mesh



Error Norms

Figure 3.1-1.. Radiative Heat Flux

Table 3.1-1.. Radiative Heat Flux: Convergence Rates for Hex8 Meshes

Num Dofs	L^2	H^1	L^∞
121	2.27	1.13	2.21
441	2.14	1.07	2.15
1681	2.07	1.04	2.07

3.2. RADIATIVE HEAT FLUX FROM FORTRAN USER SUBROUTINE

This test verifies that a user-supplied subroutine for convective coefficient and reference temperature (restricted to a surface patch) produces the same results as the equivalent input syntax with constant values. The user subroutine is applied to the entire exterior surface, while the case using constant values must be applied only to specific sidesets that span a portion of the exterior surface.

3.2.1. Features Tested

Basic heat conduction on a Hex8 mesh; convective and radiative flux BCs, Fortran user subroutines.

3.2.2. Boundary Conditions

Convective and radiative flux BCs are applied to the exterior boundary.

3.2.3. Material Parameters

The values of density, thermal conductivity, emissivity and specific heat are all constant.

3.2.4. Verification of Solution

The test compares Exodus output between two input files. The first does not use any user subroutines and instead relies on sidesets to apply the correct convective and radiative boundary conditions with constant coefficients. The second uses a single convective boundary condition with user subroutines for both the convective coefficient and reference temperature. The two input files produce results that agree to the default tolerances in the exodiff script.

For input decks see Appendix 12.2.2.

3.3. CONVECTIVE HEAT FLUX

This problem tests the convective flux boundary condition under transient heat conduction in a 2D domain. The geometry consists of a unit square.

3.3.1. Features Tested

Transient heat conduction on Quad4 meshes; convective flux boundary conditions with user subroutines for convective coefficient and reference temperature; temperature boundary conditions from C-style user subroutine and constant values.

3.3.2. Boundary Conditions

At surface 3, the temperature is prescribed from a C-style user subroutine. On surfaces 2 and 4, a constant temperature boundary condition is used. On surface 1, a convective heat flux condition is prescribed. No source term is needed. The initial condition is provided by a C-style user subroutine

3.3.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant.

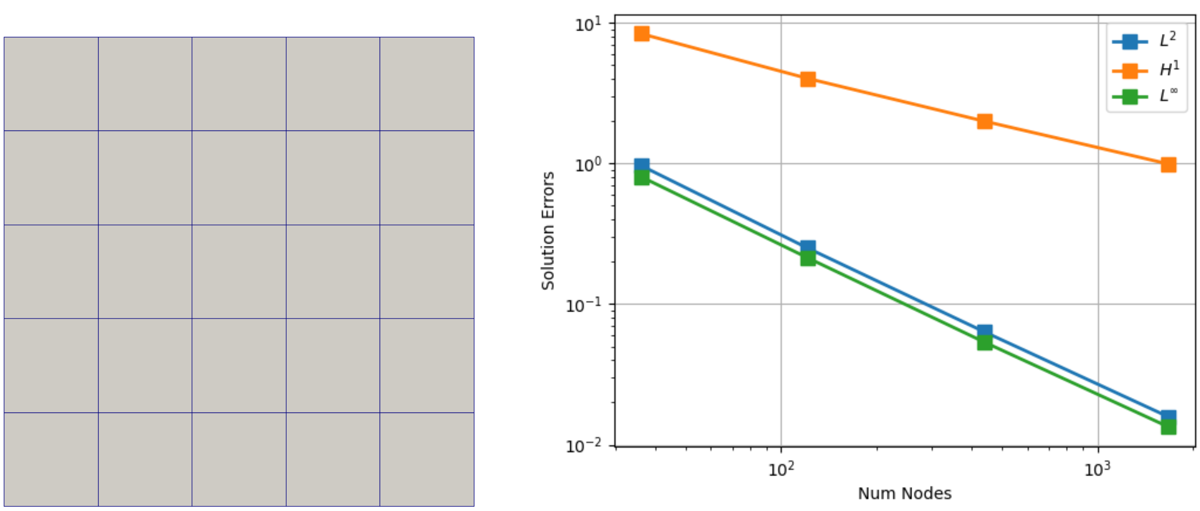
3.3.4. Verification of Solution

A manufactured solution is chosen as

$$T(x,y,t) = 100\ \exp(-2\pi^2 t)\ \sin(\pi x)\,(\cos(\pi y) + \sin(\pi y))$$

Because the solution is based on eigenfunctions, it satisfies the heat equation with no source term.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).



Coarse Mesh Error Norms

Figure 3.3-1.. Convective Heat Flux

Table 3.3-1.. Convective Heat Flux: Convergence Rates for Hex8 Meshes

Num Dofs	L^2	H^1	L^∞
121	2.22	1.21	2.19
441	2.13	1.09	2.15
1681	2.07	1.04	2.07

For input decks see Appendix 12.2.3.

3.4. THERMAL CONVECTIVE FLUX (FORTRAN SUB-ROUTINE)

3.4.1. Problem Description

This problem tests the convective flux boundary condition with a convective coefficient Fortran subroutine for a steady thermal problem in a 3D domain whose geometry consists of a unit-sized cube.

3.4.2. Features Tested

Convective Flux BC, Convective Coefficient Fortran Subroutine, user subroutine, integrated flux, integrated power

3.4.3. Boundary Conditions

Convective flux boundary conditions are imposed on surfaces 1 and 2. Dirichlet BCs are specified using the exact solution on surfaces 3-6. A source term is applied within all blocks based on substituting the exact solution into the heat conduction operator.

3.4.4. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant.

3.4.5. Verification of Solution

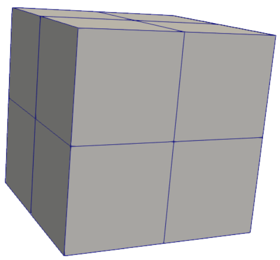
The manufactured solution is

$$T(x, y, z) = (x - x^2)^2(y - y^2)^2(z - z^2)^2 + (z + z^2).$$

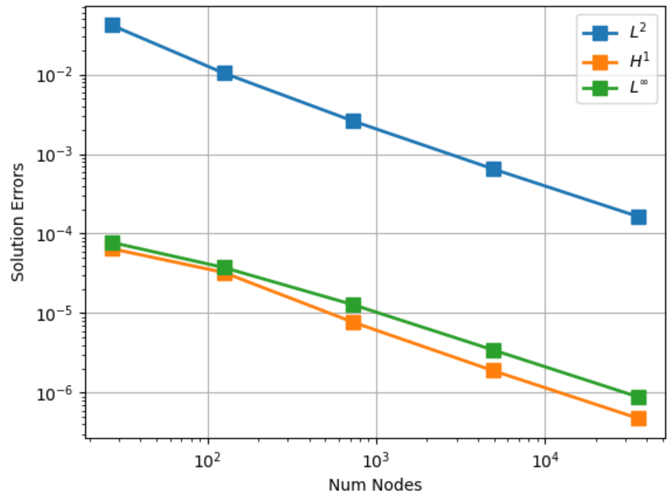
For each mesh, the errors in the temperature solution are computed in the L^2, H^1, and L^∞ norms.

Table 3.4-1.. Thermal Convective BC: Convergence Rates

Num Dofs	L^2	H^1	L^∞
125	2.71	1.35	1.43
729	2.36	2.45	1.83
4913	2.18	2.20	2.06
35937	2.09	2.09	2.06



Coarse Mesh



Error Norms

Figure 3.4-1.. Convergence for 3D thermal steady convective flux BCs.

3.5. THERMAL CONVECTIVE FLUX (USER FIELD FROM EXODUS READ-IN)

3.5.1. Problem Description

This problem evalutates a convective flux boundary condition with a convective coefficient and a reference temperature from an exodus file for a steady thermal problem in a 3D domain whose geometry consists of a unit-sized cube.

3.5.2. Features Tested

Convective Flux BC, Convective Coefficient, transfers, user subroutine, integrated flux, integrated power

3.5.3. Boundary Conditions

Convective flux boundary conditions are imposed on surfaces 1 and 2. Dirichlet BCs are specified using the exact solution on surfaces 3-6. A source term is applied within all blocks based on substituting the exact solution into the heat conduction operator.

3.5.4. Material Parameters

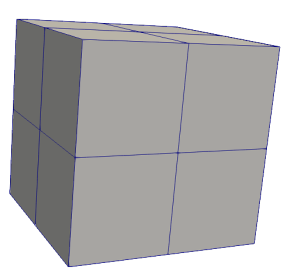
The values of density, thermal conductivity, and specific heat are all constant.

3.5.5. Verification of Solution

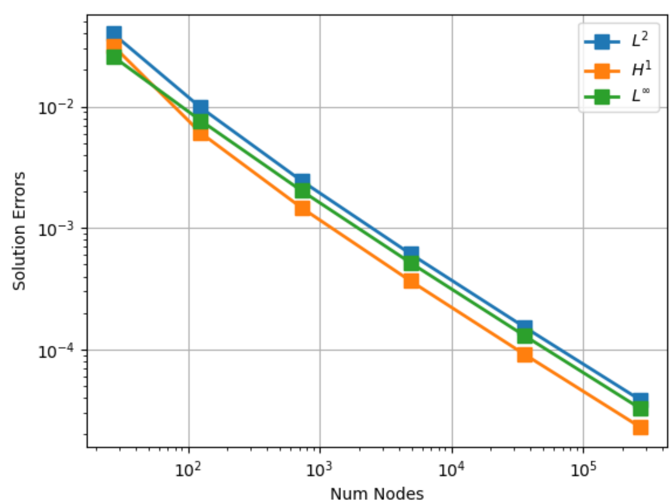
The manufactured solution is

$$T(x, y, z) = (x - x^2)^2(y - y^2)^2(z - z^2)^2 + (z^2 + z).$$

For each mesh, the errors in the temperature solution are computed in the L^2, H^1, and L^∞ norms.



Coarse Mesh



Error Norms

Figure 3.5-1.. Convergence for 3D thermal steady convective flux BCs.

Table 3.5-1.. Thermal Convective BC: Convergence Rates

Num Dofs	L^2	H^1	L^∞
125	2.74	3.24	2.38
729	2.36	2.41	2.26
4913	2.18	2.19	2.15
35937	2.09	2.09	2.08
274625	2.05	2.05	2.04

3.6. THERMAL RADIATIVE HEAT FLUX

3.6.1. Basic Calore-Style BC

3.6.1.1. Problem Description

This problem evaluates a steady thermal solution with radiative heat flux boundary conditions on a 3D unit cube domain.

3.6.1.2. Features Tested

Basic heat conduction, Calore style radiative heat flux BCs, Integrated Flux Output, Integrated Power Output, Hex8 meshes.

3.6.1.3. Boundary Conditions

Dirichlet BCs are specified using the exact solution on surfaces 2-6. On surface 1, a radiative heat flux BC is specified with constant emissivity and a radiation form factor of 0.2. A source term is applied within all blocks based on substituting the exact solution into the heat conduction operator.

3.6.1.4. Material Parameters

The values of density, thermal conductivity, specific heat, and emissivity are all constant values.

3.6.1.5. Verification of Solution

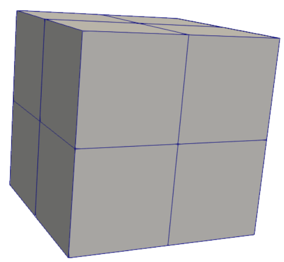
The manufactured solution is

$$T(x,y,x) = (x-x^2)^2(y-y^2)^2(z-z^2)^2 + T - \frac{\partial T}{\partial n}(z^2 - z).$$

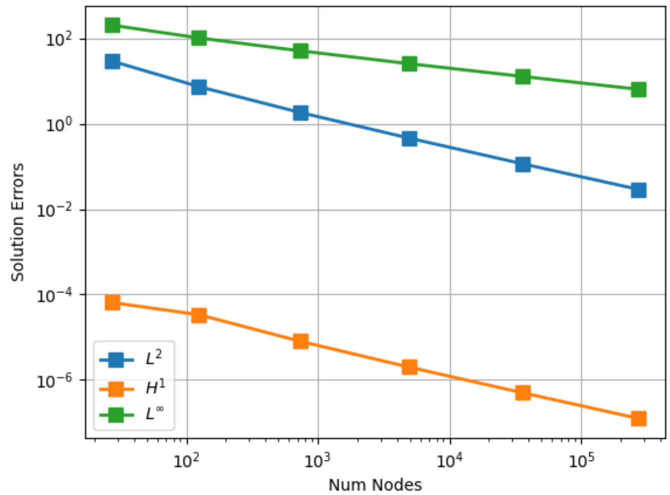
For each discretization, the errors in the temperature solution are computed in the L^2, H^1, and L^∞ norms. The observed rates of convergence are 2 (except for the L^∞ norm, with convergence order 1).

Table 3.6-1.. Thermal Radiative Flux BC: Convergence Rates

Num Dofs	L^2	L^∞	H^1
125	2.71	1.29	1.36
729	2.36	2.43	1.18
4913	2.18	2.19	1.09
35937	2.09	2.09	1.05
274625	2.05	2.05	1.02



Coarse Mesh



Error Norms

Figure 3.6-1.. Thermal Radiative Flux

3.6.2. With Fortran Subroutines

3.6.2.1. Problem Description

This problem evaluates a steady thermal solution with radiative heat flux boundary conditions using Fortran subroutines on a 3D unit cube domain.

3.6.2.2. Features Tested

Basic heat conduction, Calore style radiative heat flux BCs, Integrated Flux Output, Integrated Power Output, Hex8 meshes, Fortran subroutines.

3.6.2.3. Boundary Conditions

Dirichlet BCs are specified using the exact solution on surfaces 2-6. On surface 1, a radiative heat flux BC is specified with emissivity, reference temperature, and radiation form factor of provided by Fortran subroutines. A source term is applied within all blocks based on substituting the exact solution into the heat conduction operator.

3.6.2.4. Material Parameters

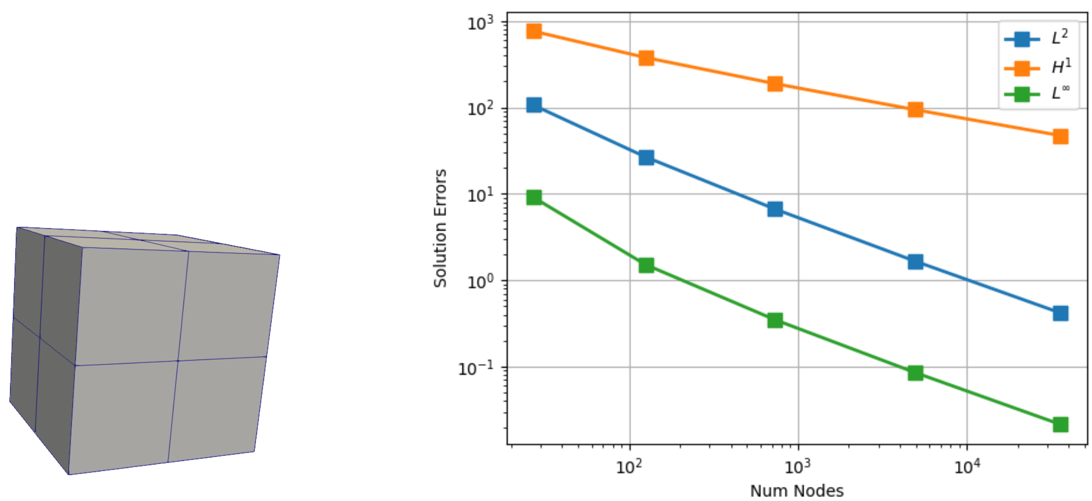
The values of density, thermal conductivity, specific heat, and emissivity are all constant values.

3.6.2.5. Verification of Solution

The manufactured solution is

$$T(x,y,x) = (x-x^2)^2(y-y^2)^2(z-z^2)^2 + T - \frac{\partial T}{\partial n}(z^2 - z).$$

For each discretization, the errors in the temperature solution are computed in the L^2, H^1, and L^∞ norms. The observed rates of convergence are 2 (except for the L^∞ norm, with convergence order 1).



Coarse Mesh Error Norms

Figure 3.6-2.. Thermal Radiative Flux

Table 3.6-2.. Thermal Radiative Flux BC: Convergence Rates

Num Dofs	L^2	L^∞	H^1
125	2.72	1.38	3.52
729	2.36	1.18	2.50
4913	2.18	1.09	2.21
35937	2.09	1.05	2.10

3.6.3. With User Subroutines

3.6.3.1. Problem Description

This problem evaluates a steady thermal solution with radiative heat flux boundary conditions with user subroutines on a 3D unit cube domain.

3.6.3.2. Features Tested

Basic heat conduction, Calore style radiative heat flux BCs, Integrated Flux Output, Integrated Power Output, Hex8 meshes, user subroutines.

3.6.3.3. Boundary Conditions

Dirichlet BCs are specified using the exact solution on surfaces 2-6. On surface 1, a radiative heat flux BC is specified with emissivity, reference termperature and radiation form factor provided by user subroutines. A source term is applied within all blocks based on substituting the exact solution into the heat conduction operator.

3.6.3.4. Material Parameters

The values of density, thermal conductivity, specific heat, and emissivity are all constant values.

3.6.3.5. Verification of Solution

The manufactured solution is

$$T(x,y,x) = (x-x^2)^2(y-y^2)^2(z-z^2)^2 + T - \frac{\partial T}{\partial n}(z^2 - z).$$

For each discretization, the errors in the temperature solution are computed in the L^2, H^1, and L^∞ norms. The observed rates of convergence are 2 (except for the L^∞ norm, with convergence order 1).

Table 3.6-3.. Thermal Radiative Flux BC: Convergence Rates

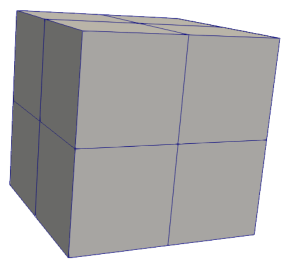
Num Dofs	L^2	L^∞	H^1
125	2.72	1.38	3.52
729	2.36	1.18	2.50
4913	2.18	1.09	2.21
35937	2.09	1.05	2.10

3.7. ADVECTIVE BAR

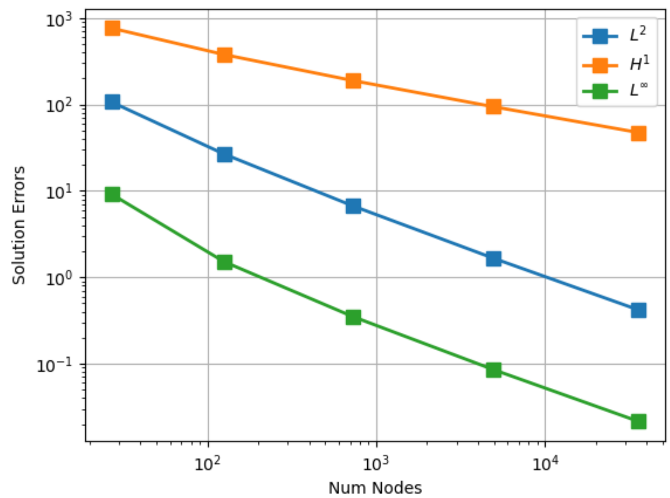
Advective bar model verification tests.

3.7.1. Steady Advection-Diffusion

The three dimensional Bar2 meshes of one element block are generated in Cubit.



Coarse Mesh



Error Norms

Figure 3.6-3.. Thermal Radiative Flux

3.7.2. Features Tested

Steady heat conduction on 3D Bar2 meshes, Dirichlet boundary conditions, constant source term, advection and SUPG stabilization.

3.7.3. Boundary Conditions

$$T(0) = T(1) = 0$$

3.7.4. Material Parameters

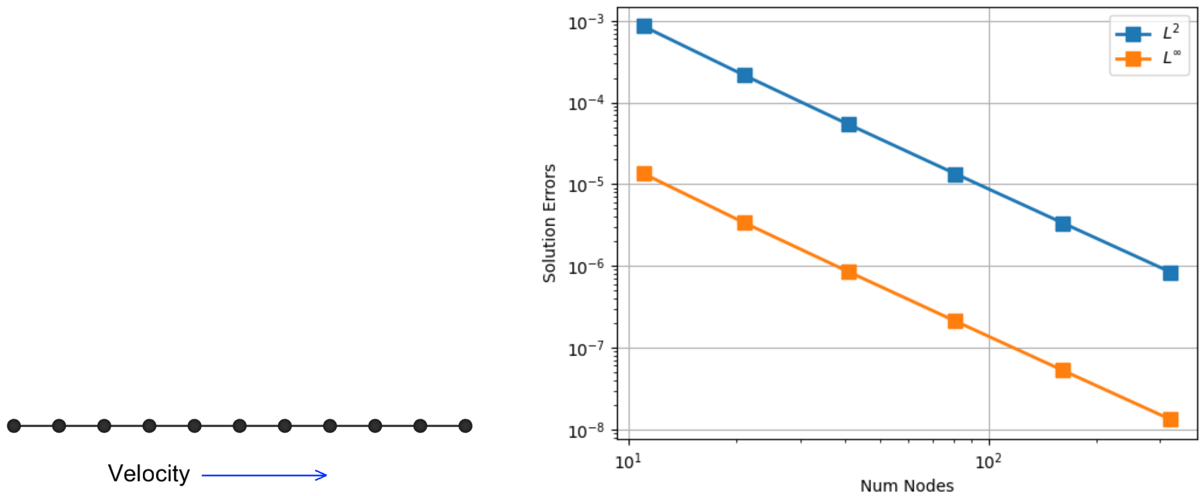
The values of density, thermal conductivity, and specific heat are all constant, equal to one in the block.

3.7.5. Verification of Solution

Solution verification is carried out by computing the error in the numerical solution based upon comparison with the analytic solution.

$$T(x) = \frac{1}{\rho CV}\left[x - \frac{1-\exp(x\gamma)}{1-\exp(\gamma)}\right]$$

where $\gamma = \rho C V / k$ where ρ is the density, C is specific heat, k is the thermal conductivity and V is the advection velocity. In this test, we find that the convergence rate for the temperature in the L^∞ and L^2 norms are 2.



Coarse Mesh Error Norms

Figure 3.7-1.. Steady Advective Conduction: 3D Bar2 Meshes

Table 3.7-1.. Steady Advective Conduction: Convergence Rates for 3D Bar2 Meshes

Num Dofs	L^2	L^∞
21	2.14	2.14
41	2.07	2.07
81	2.04	2.04
161	2.02	2.02
321	2.01	2.01

3.7.6. Transient Advection-Diffusion

The three dimensional Bar2 meshes of one element block are generated in Cubit.

3.7.7. Features Tested

Transient heat conduction on 3D Bar2 meshes, Dirichlet boundary conditions and Encore function source term.

3.7.8. Boundary Conditions

Dirichlet boundary conditions on the bar ends based upon the manufactured solution $T(x)$

$$T(0) = T(1) = T_i$$

3.7.9. Material Parameters

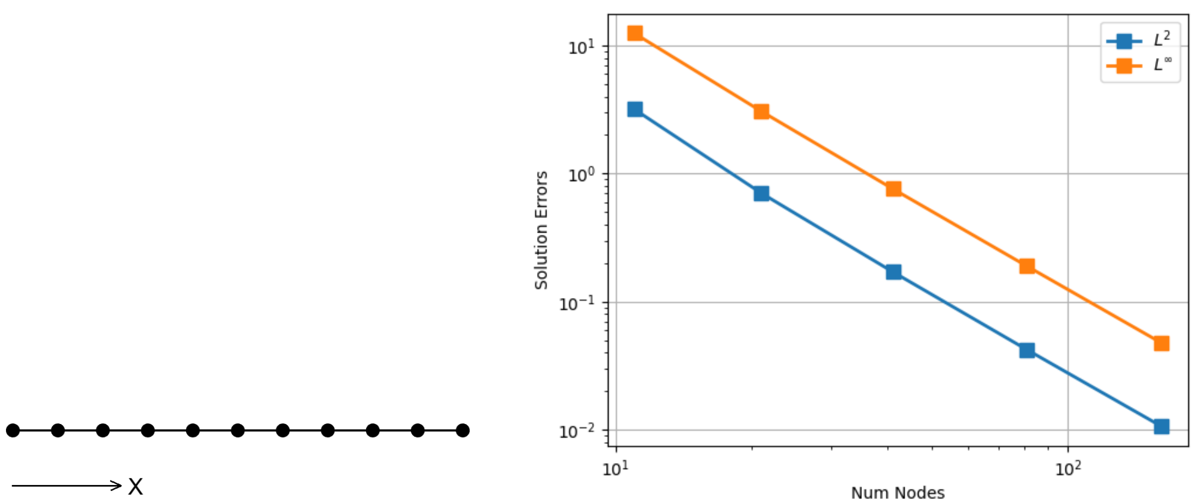
The values of density, thermal conductivity, and specific heat are all constant, equal to one in the bar block.

3.7.10. Verification of Solution

Solution verification is carried out by computing the error in the numerical solution based upon comparison with the analytic solution.

$$T(x) = T_i + Atx(x-1)\exp(-Bt)\exp(-Bx)$$

In this test, we find that the convergence rate for the temperature in the L^∞ and L^2 norms are 2.



Coarse Mesh   Error Norms

Figure 3.7-2.. Transient Heat Conduction: 3D Bar2 Meshes

3.7.11. Transient Advection-Diffusion in 2D

The two dimensional Bar2 meshes of one elment block are generated in Cubit.

Table 3.7-2.. Transient Heat Conduction: Convergence Rates for 3D Bar2 Meshes

Num Dofs	L^2	L^∞
21	2.33	2.17
41	2.12	2.08
81	2.05	2.04
161	2.02	2.02

3.7.12. Features Tested

Transient heat conduction on 2D Bar2 meshes, Dirichlet boundary conditions and Encore function source term.

3.7.13. Boundary Conditions

Dirichlet boundary conditions on the bar ends

$$T(0) = T(1) = T_i$$

3.7.14. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in the bar block.

3.7.15. Verification of Solution

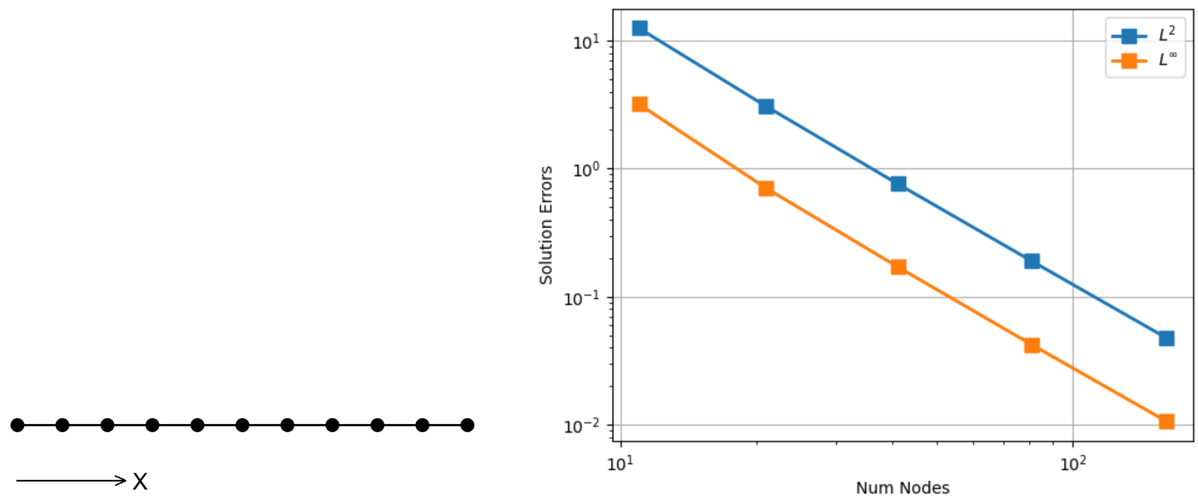
Solution verification is carried out by computing the error in the numerical solution based upon comparison with the analytic solution.

$$T(x) = T_i + Atx(x-1)\exp(-Bt)\exp(-Bx)$$

In this test, we find that the convergence rate for the temperature in the L^∞ and L^2 norms are 2.

Table 3.7-3.. Transient Heat Conduction: Convergence Rates for 2D Bar2 Meshes

Num Dofs	L^2	L^∞
21	2.17	2.33
41	2.08	2.12
81	2.04	2.05
161	2.02	2.02



Coarse Mesh Error Norms

Figure 3.7-3.. Transient Heat Conduction: Bar2 Meshes

3.8. SOLUTION VERIFICATION

This test is for a Mock AFF (including a metal case, foam, mock components, and temperature-dependent properties) that uses extrapolation to determine an approximation to the exact solution as a function of the results from three levels of meshes.

3.8.1. Features Tested

Extrapolation, Radiative flux boundary condition

3.8.2. Material Parameters

Constant density, emissivity. Temperature dependent user functions for specific hear and thermal conductivity.

3.8.3. Verification of Solution

Quantities of interest are the maximum, minimum, and average temperatures on both blocks and points. There is no manufactured solution in this case, instead an extrapolated solution is calculated and used to measure convergence and approximate the absolute error for a given mesh resolution.

Documentation for the following tests is in progress:

```
nic_material_decomposition/organic_material_decomposition.test|np4
```

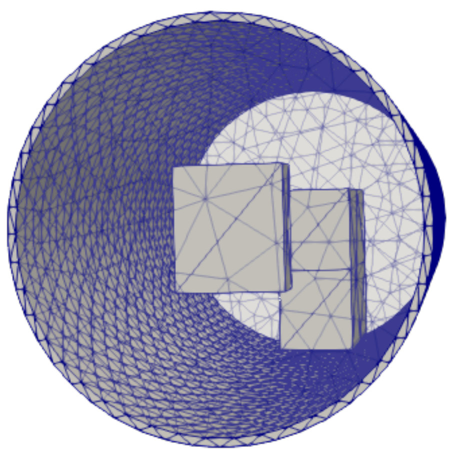


Figure 3.8-1.. Mock AFF Solution Verification

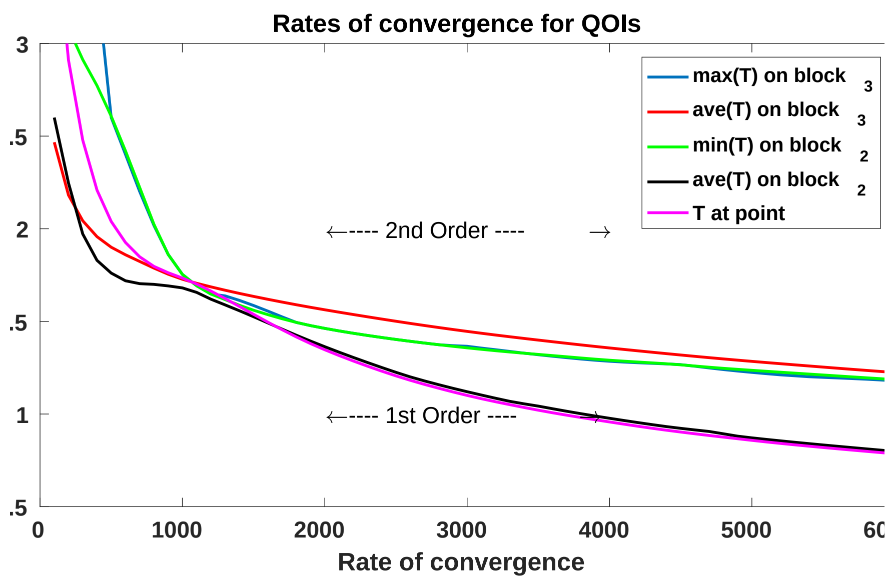


Figure 3.8-2.. The convergence rates can vary over time and between QOIs

4. THERMAL CONTACT

4.1. 1D FLAT CONTACT

This problem tests thermal contact along a flat surface using 3D domains. The geometry consists of two thick blocks, which are in contact along a common flat surface. The mesh nodes on either side of the contact surface are not aligned in general.

In this problem we observe sub-optimal convergence rates in the L^∞ norm when using Tet elements. This is a known issue with unknown cause.

The contact search tolerances are fixed for all meshes, with a zero tangential and normal tolerances.

4.1.1. Features Tested

Basic heat conduction, tied and resistance thermal contact between non-matching meshes (Hex-Hex, Tet-Tet, Hex-Tet).

4.1.2. Boundary Conditions

The interface between the two blocks is a thermal contact boundary condition. Both tied contact and resistance contact (with finite contact resistance) are tested. The left and right boundary conditions are prescribed using constant values. The remaining boundary conditions are adiabatic. A constant source term is applied in each block (with different signs).

4.1.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.1.4. Verification of Solution

A manufactured solution is chosen based on the contact interface at $x = 0$:

$$T(x, y, z) = \begin{cases} \frac{1}{2}(1 + x)(\gamma + x), & x < 0, \\ 1 + \frac{1}{2}(1 - x)(-\gamma + x), & x > 0 \end{cases}$$

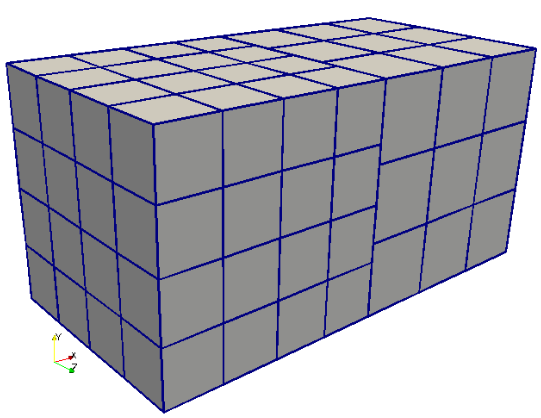
where $\gamma = (2 - R)/(2 + R)$ is a constant depending on the thermal contact resistance R. Here R is the inverse of the contact conductance that is provided as a code input. In the case of tied contact, $R = 0$ and therefore $\gamma = 1$. We note that when $R > 0$, this exact solution exhibits a jump in temperature across the contact interface.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

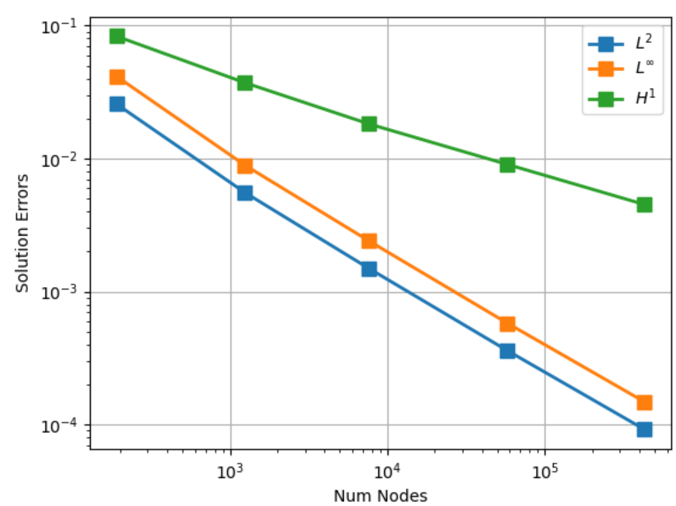
These rates are observed for the Hex-Hex case; however, both of the cases involving Tet meshes exhibit a reduced order of convergence in the L^∞ norms (convergence rate about 1.7).

For input decks see Appendix 12.3.1.

4.1.5. Results: Hex8 Tied



Coarse Mesh



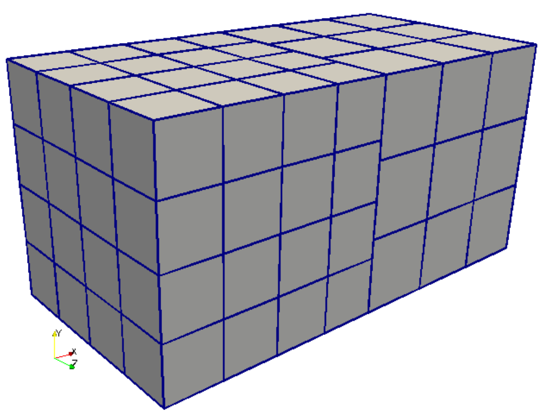
Error Norms

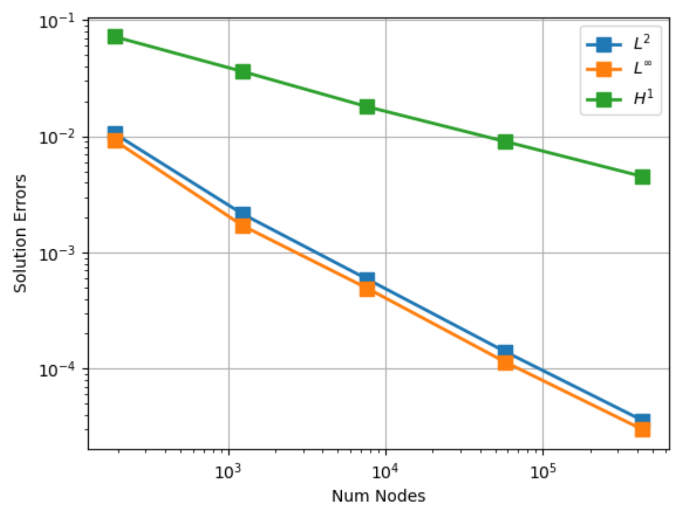
Figure 4.1-1.. 1D Flat Contact: Hex8 Tied

4.1.6. Results: Hex8 Resistance

Table 4.1-1.. 1D Flat Contact: Convergence Rates for Hex8 Tied

Num Dofs	L^2	L^∞	H^1
1241	2.45	2.46	1.29
7657	2.17	2.16	1.18
57889	2.11	2.11	1.04
432089	2.04	2.04	1.04


Coarse Mesh



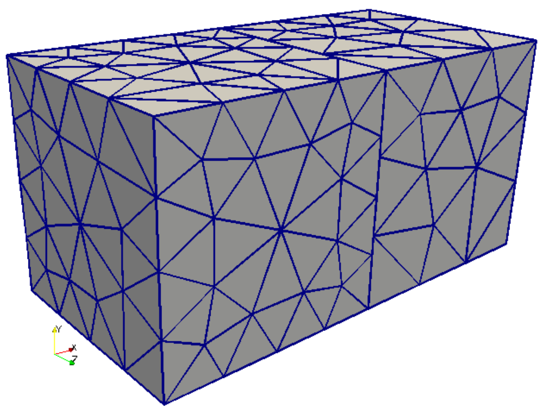
Error Norms

Figure 4.1-2.. 1D Flat Contact: Hex8 Resistance

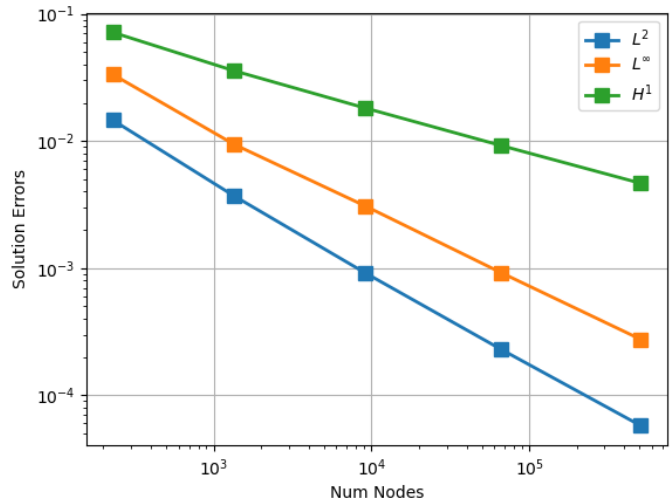
Table 4.1-2.. 1D Flat Contact: Convergence Rates for Hex8 Resistance

Num Dofs	L^2	L^∞	H^1
1241	2.55	2.70	1.10
7657	2.12	2.04	1.14
57889	2.13	2.17	1.03
432089	2.04	2.00	1.03

4.1.7. Results: Tet4 Tied



Coarse Mesh



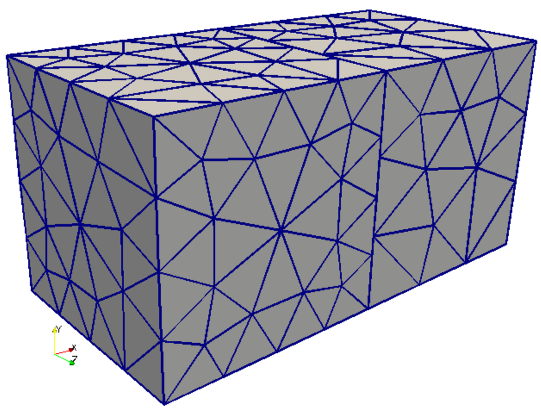
Error Norms

Figure 4.1-3.. 1D Flat Contact: Tet4 Tied

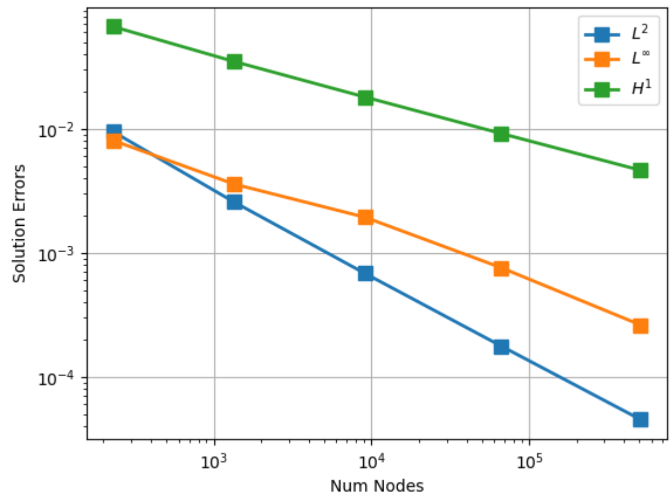
Table 4.1-3.. 1D Flat Contact: Convergence Rates for Tet4 Tied

Num Dofs	L^2	L^∞	H^1
1348	2.33	2.15	1.18
9102	2.19	1.75	1.06
66618	2.09	1.83	1.02
509170	2.04	1.78	1.01

4.1.8. Results: Tet4 Resistance



Coarse Mesh



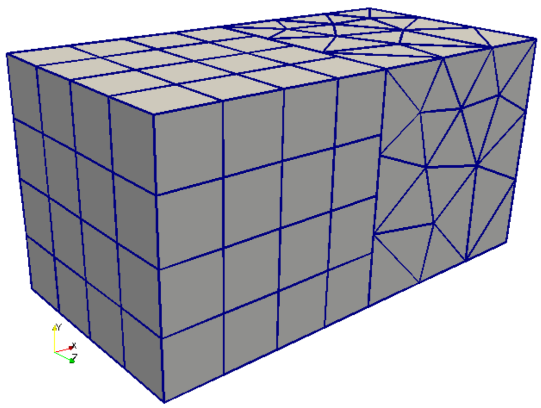
Error Norms

Figure 4.1-4.. 1D Flat Contact: Tet4 Resistance

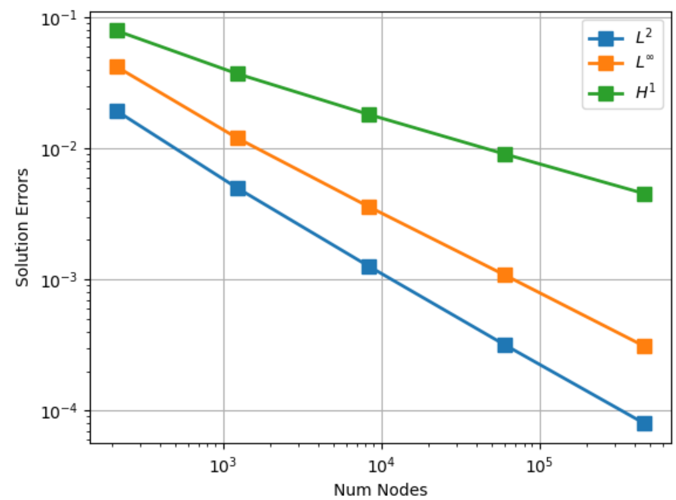
Table 4.1-4.. 1D Flat Contact: Convergence Rates for Tet4 Resistance

Num Dofs	L^2	L^∞	H^1
1348	2.22	1.37	1.10
9102	2.08	0.96	1.04
66618	2.04	1.42	1.02
509170	2.01	1.57	1.01

4.1.9. Results: Hex8-Tet4 Tied



Coarse Mesh



Error Norms

Figure 4.1-5.. 1D Flat Contact: Hex8-Tet4 Tied

Table 4.1-5.. 1D Flat Contact: Convergence Rates for Hex8-Tet4 Tied

Num Dofs	L^2	L^∞	H^1
1231	2.32	2.15	1.29
8284	2.16	1.89	1.12
60566	2.09	1.82	1.05
462762	2.04	1.84	1.02

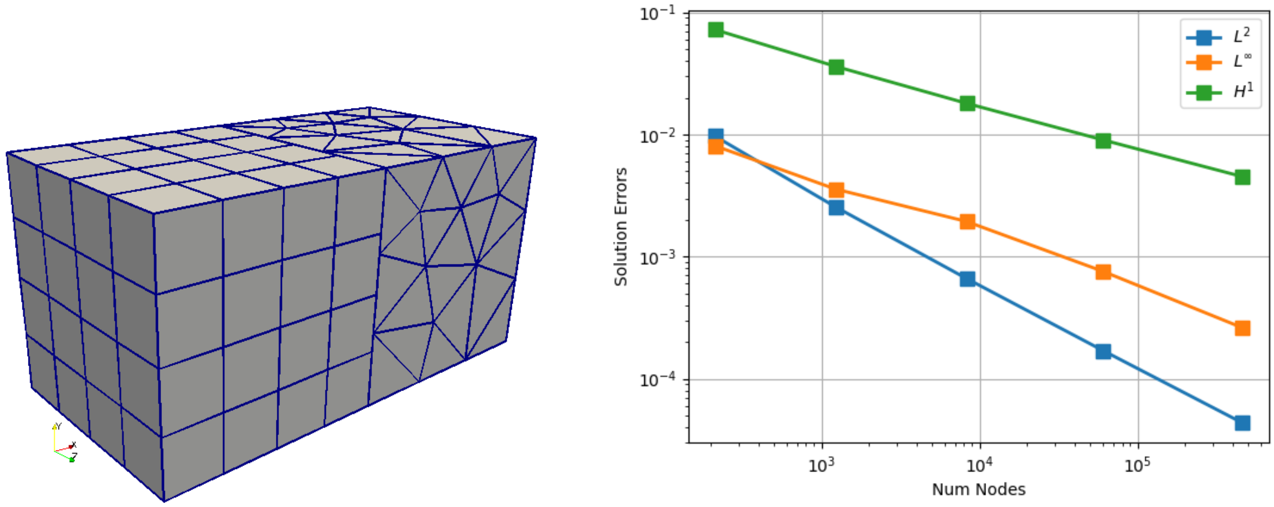
4.1.10. Results: Hex8-Tet4 Resistance

4.2. 3D CURVED CONTACT

This problem tests thermal contact along a curved surface in 3D. The geometry consists of two thick spherical shells, which are in contact along a shared surface. The mesh nodes on either side of the contact surface are not aligned in general.

In this problem we observe sub-optimal convergence rates in the L^∞ norm when using tet elements. This is a known issue with unknown cause.

The contact search tolerances are fixed for all meshes, with a zero tangential tolerance and a normal tolerance large enough to insure a proper contact search on the coarsest mesh.



Coarse Mesh Error Norms

Figure 4.1-6.. 1D Flat Contact: Hex8-Tet4 Resistance

Table 4.1-6.. 1D Flat Contact: Convergence Rates for Hex8-Tet4 Resistance

Num Dofs	L^2	L^∞	H^1
1231	2.28	1.39	1.18
8284	2.12	0.96	1.09
60566	2.05	1.42	1.05
462762	2.01	1.57	1.02

4.2.1. Features Tested

Basic heat conduction, tied thermal contact between non-matching meshes (hex-hex, tet-tet, hex-tet).

4.2.2. Boundary Conditions

The interface between the two blocks is a tied thermal contact boundary condition. The outer and inner boundary conditions are prescribed at the nodes using the analytic solution.

4.2.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.2.4. Verification of Solution

A manufactured solution is chosen as

$$T(x, y, z) = -3x^2z - 3y^2z + 2z^3$$

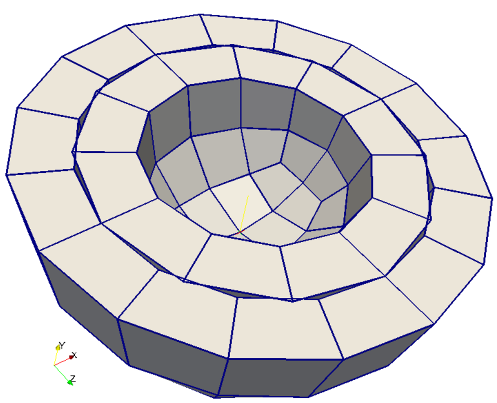
This solution is harmonic, implying that no source term is needed for the steady state heat equation with constant conductivity.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

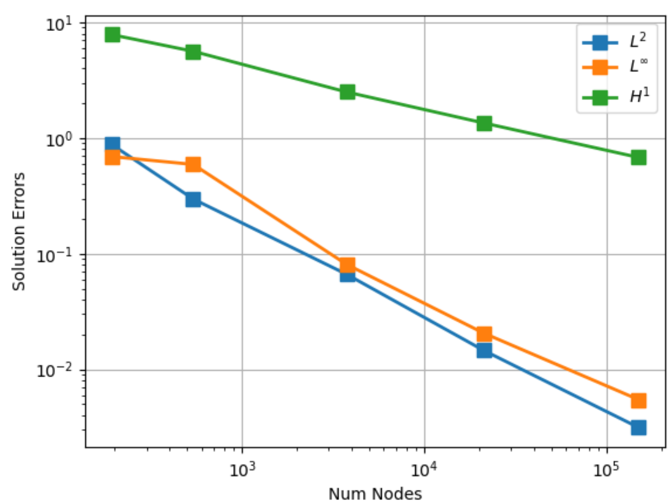
These rates are observed for the hex-hex case; however, both of the cases involving tet meshes exhibit a reduced order of convergence in the L^∞ norms (convergence rate about 1.7).

For input decks see Appendix 12.3.2.

4.2.5. Results: Hex8-Hex8 Contact



Coarse Mesh



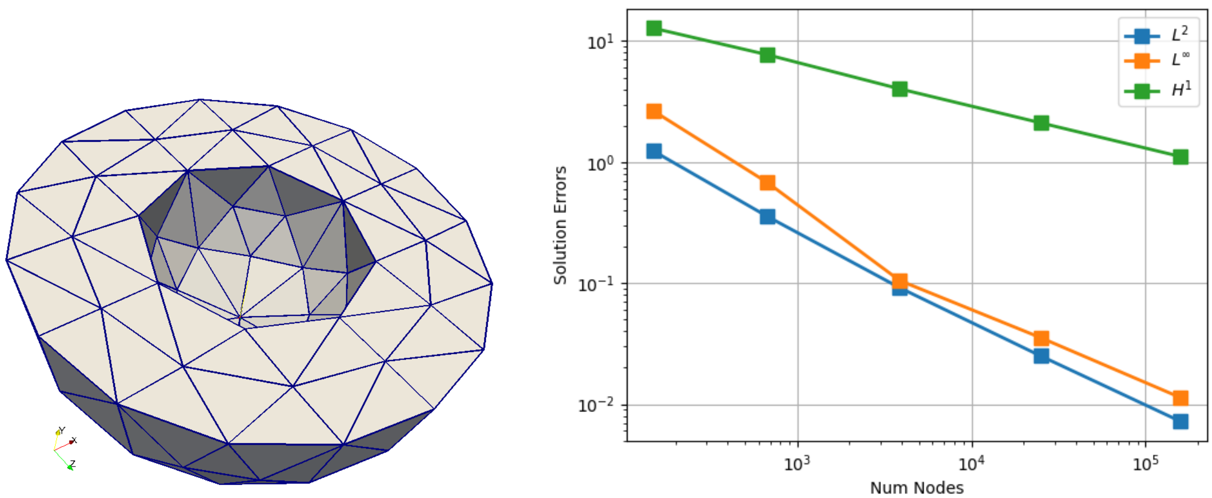
Error Norms

Figure 4.2-1.. 3D Curved Contact: Hex8-Hex8 Case

Table 4.2-1.. 3D Curved Contact: Convergence Rates for Hex8-Hex8

Num Dofs	L^2	L^∞	H^1
540	3.16	0.46	0.96
3752	2.32	3.08	1.25
21216	2.62	2.37	1.07
150744	2.35	2.02	1.06

4.2.6. Results: Tet4-Tet4 Contact



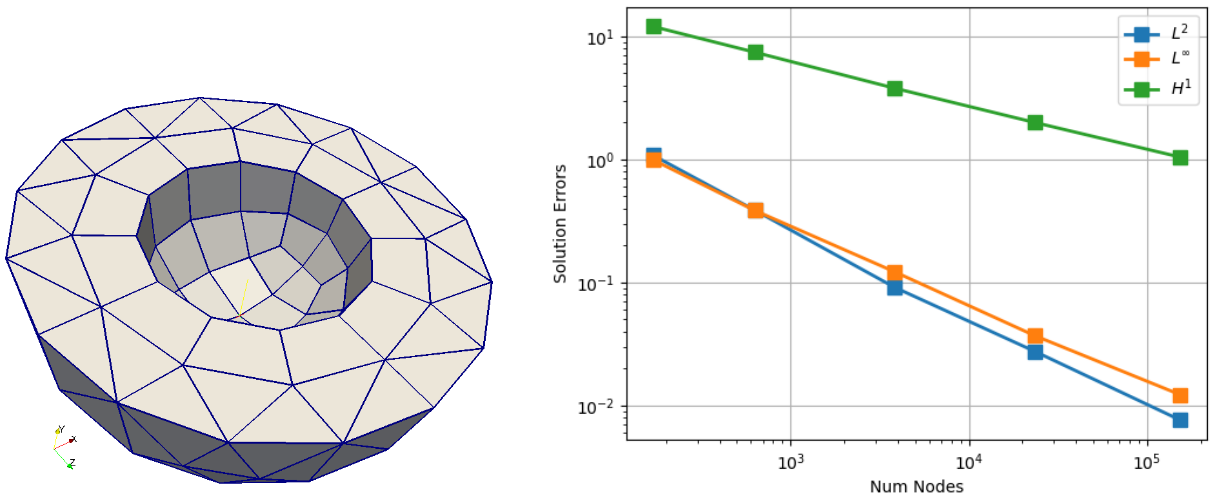
Coarse Mesh Error Norms

Figure 4.2-2.. 3D Curved Contact: Tet4-Tet4 Case

Table 4.2-2.. 3D Curved Contact: Convergence Rates for Tet4-Tet4

Num Dofs	L^2	L^∞	H^1
674	2.47	2.71	1.00
3881	2.33	3.20	1.11
25008	2.09	1.75	1.04
159147	2.02	1.85	1.04

4.2.7. Results: Hex8-Tet4 Contact



Coarse Mesh | Error Norms

Figure 4.2-3.. 3D Curved Contact: Hex8-Tet4 Case

Table 4.2-3.. 3D Curved Contact: Convergence Rates for Hex8-Tet4

Num Dofs	L^2	L^∞	H^1
630	2.34	2.16	1.11
3830	2.40	1.91	1.12
23416	1.98	1.98	1.06
153671	2.06	1.77	1.04

4.3. STEADY HEX8 CONTACT

This problem tests basic steady state heat conduction in a 3D domain. The geometry consists of a unit cube.

4.3.1. Features Tested

Basic heat conduction on Hex8 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.3.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine).

4.3.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.3.4. Verification of Solution

A manufactured solution is chosen as

$$T(x, y, z) = 1 + (x - x^2)^2(y - y^2)^2(z - z^2)^2.$$

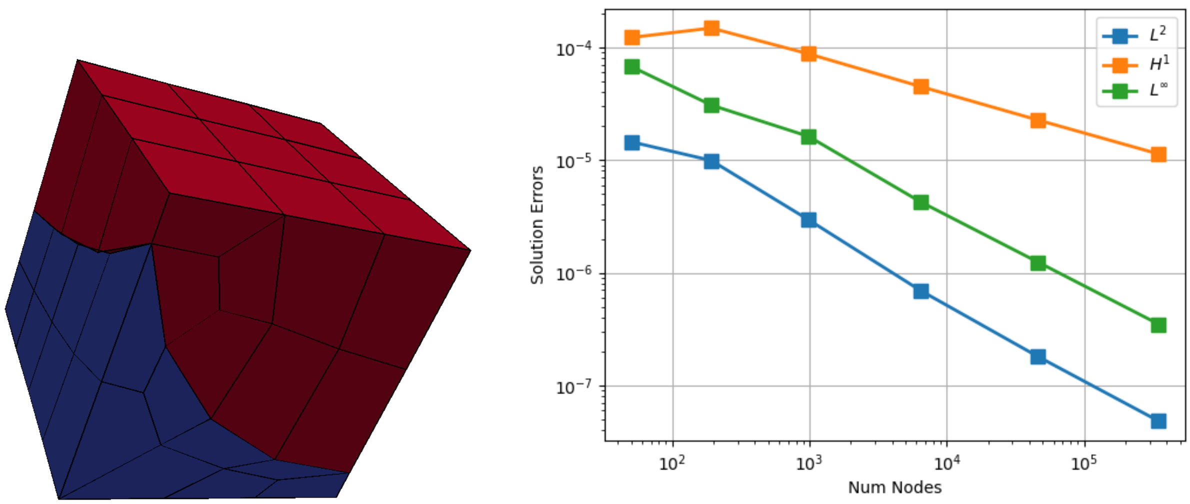
The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

Table 4.3-1.. Steady Tied Contact: Convergence Rates for Hex8 Meshes

Num Dofs	L^2	H^1	L^∞
192	0.86	-0.44	1.76
982	2.22	0.97	1.18
6419	2.31	1.07	2.13
46277	2.06	1.04	1.87
350649	1.95	1.02	1.88

For input decks see Appendix 12.3.3.



Coarse Mesh Error Norms

Figure 4.3-1.. Steady Tied Contact: Hex8 Meshes

4.4. STEADY HEX20 CONTACT

This problem tests basic steady state heat conduction in a 3D domain. The geometry consists of a unit cube. A variety of different source terms and boundary conditions are simultaneously applied. The exact solution is a manufactured solution.

4.4.1. Features Tested

Basic heat conduction on Hex20 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.4.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine).

4.4.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

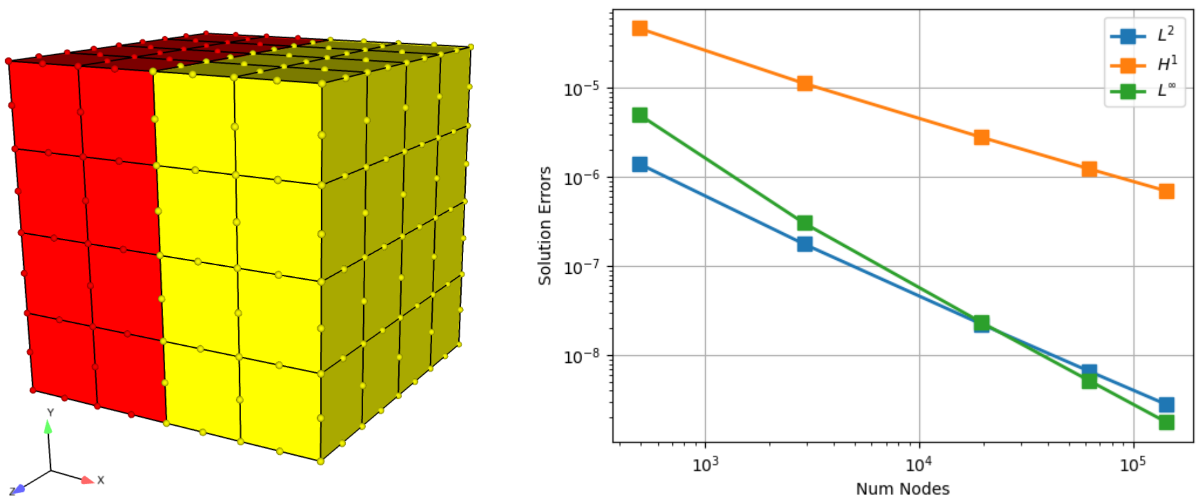
4.4.4. Verification of Solution

A manufactured solution is chosen as

$$T(x, y, z) = 1 + (x - x^2)^2(y - y^2)^2(z - z^2)^2.$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).



Coarse Mesh Error Norms

Figure 4.4-1.. Steady Heat Conduction: Hex20 Meshes

Table 4.4-1.. Steady Heat Conduction: Convergence Rates for Hex20 Meshes

Num Dofs	L^2	H^1	L^∞
2898	3.50	2.40	4.74
19618	3.25	2.19	4.03
62450	3.15	2.10	3.89
143682	3.10	2.07	3.87

For input decks see Appendix 12.3.4.

4.5. STEADY HEX27 CONTACT

This problem tests basic steady state heat conduction in a 3D domain. The geometry consists of a unit cube. A variety of different source terms and boundary conditions are simultaneously applied. The exact solution is a manufactured solution.

4.5.1. Features Tested

Basic heat conduction on Hex27 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.5.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine).

4.5.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.5.4. Verification of Solution

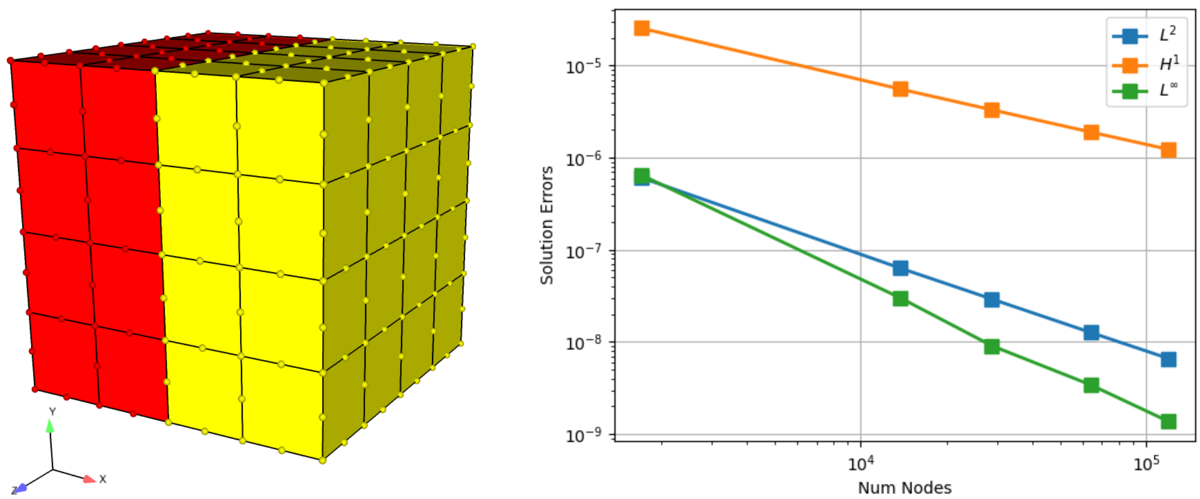
A manufactured solution is chosen as

$$T(x, y, z) = 1 + (x - x^2)^2(y - y^2)^2(z - z^2)^2.$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

For input decks see Appendix 12.3.5.



Coarse Mesh Error Norms

Figure 4.5-1.. Steady Heat Conduction: Hex27 Meshes

Table 4.5-1.. Steady Heat Conduction: Convergence Rates for Hex27 Meshes

Num Dofs	L^2	H^1	L^∞
13754	3.25	2.18	4.41
28830	3.13	2.10	4.85
63882	3.14	2.09	3.68
120050	3.11	2.07	4.38

4.6. STEADY TET4 CONTACT

This problem is identical to the one in Section 2.1 except that unstructured Tet4 meshes are used instead.

4.6.1. Features Tested

Basic heat conduction on Tet4 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.6.2. Boundary Conditions

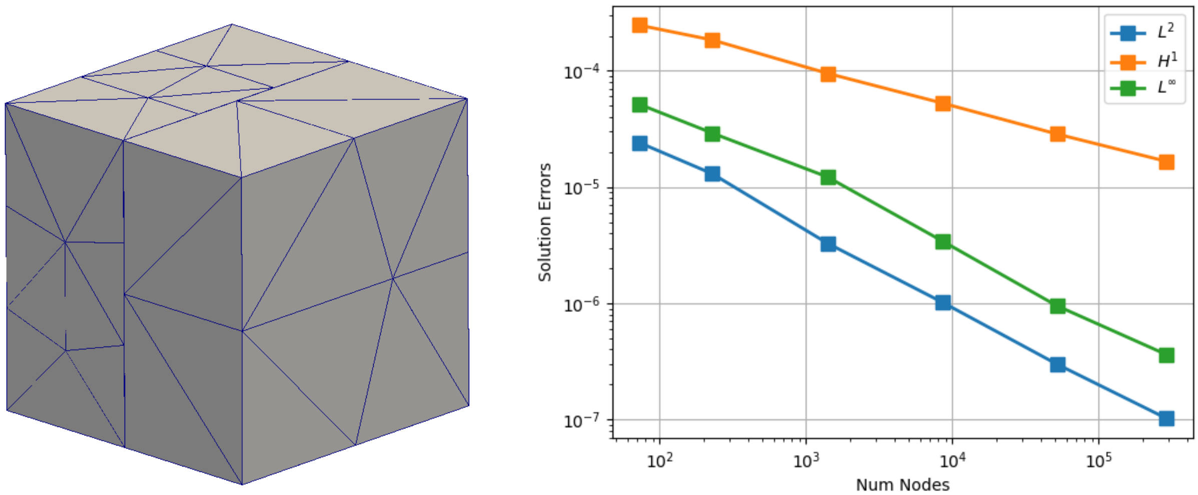
Same as in Section 2.1.

4.6.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.6.4. Verification of Solution

Same as in Section 2.1.



Coarse Mesh Error Norms

Figure 4.6-1.. Steady Tied Contact: Tet4 Meshes

Table 4.6-1.. Steady Tied Contact: Convergence Rates for Tet4 Meshes

Num Dofs	L^2	H^1	L^∞
229	1.61	0.76	1.49
1402	2.29	1.10	1.44
8535	1.93	0.98	2.11
51622	2.05	1.02	2.13
291153	1.88	0.94	1.68

For input decks see Appendix 12.3.6.

4.7. STEADY TET4TET10 CONTACT

This problem is identical to the one in Section 2.1 except that unstructured Tet4 meshes are used instead.

4.7.1. Features Tested

Basic heat conduction on Tet4 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.7.2. Boundary Conditions

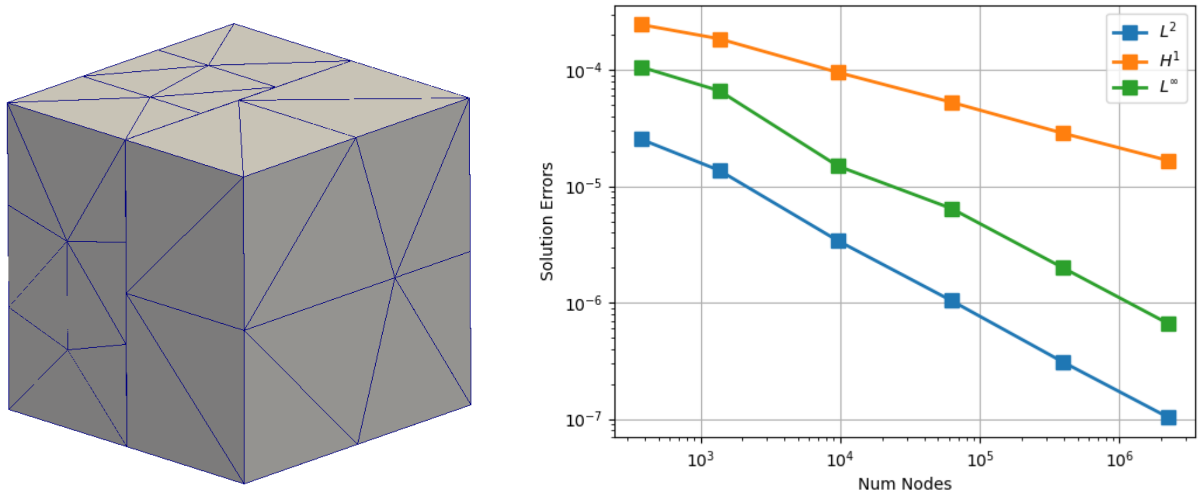
Same as in Section 2.1.

4.7.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.7.4. Verification of Solution

Same as in Section 2.1.



Coarse Mesh Error Norms

Figure 4.7-1.. Steady Tied Contact: Tet4 Meshes

For input decks see Appendix 12.3.7.

Table 4.7-1.. Steady Tied Contact: Convergence Rates for Tet4 Meshes

Num Dofs	L^2	H^1	L^∞
1364	1.43	0.65	1.10
9663	2.13	1.02	2.28
62724	1.90	0.95	1.35
392046	1.98	1.00	1.92
2249757	1.89	0.93	1.90

4.8. STEADY TET10 CONTACT

This problem is identical to the one in Section 2.1 except that unstructured Tet10 meshes are used instead.

4.8.1. Features Tested

Basic heat conduction on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.8.2. Boundary Conditions

Same as in Section 2.1.

4.8.3. Material Parameters

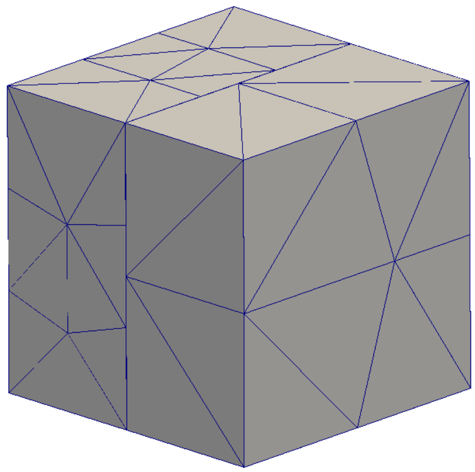
The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.8.4. Verification of Solution

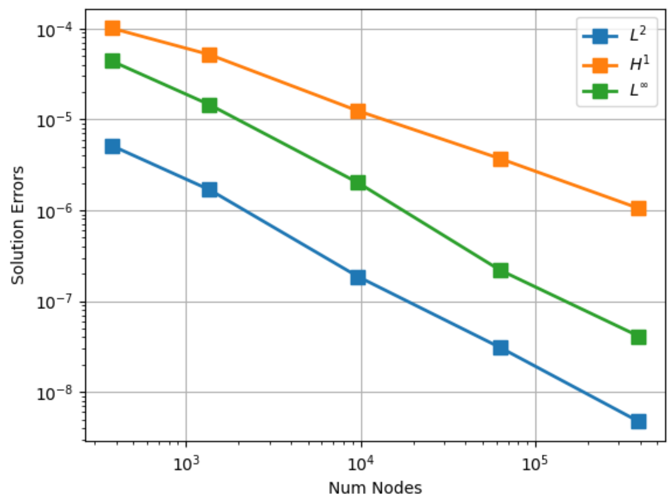
Same as in Section 2.1.

Table 4.8-1.. Steady Tied Contact: Convergence Rates for Tet10 Meshes

Num Dofs	L^2	H^1	L^∞
1364	2.60	1.55	2.57
9663	3.37	2.19	3.03
62724	2.87	1.93	3.56
392046	3.08	2.06	2.74



Coarse Mesh



Error Norms

Figure 4.8-1.. Steady Tied Contact: Tet10 Meshes

For input decks see Appendix 12.3.8.

4.9. STEADY TET10 DASH CONTACT

This problem is identical to the one in Section 2.1 except that unstructured Tet10 meshes are used instead.

4.9.1. Features Tested

Basic heat conduction on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.9.2. Boundary Conditions

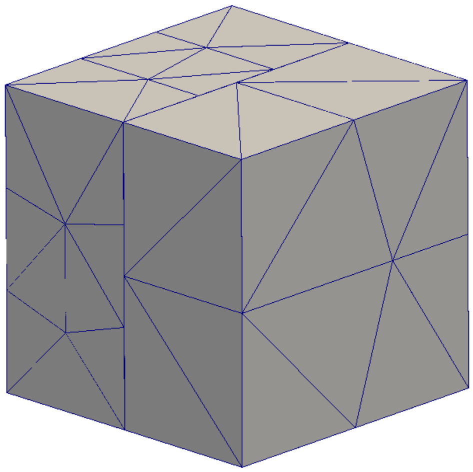
Same as in Section 2.1.

4.9.3. Material Parameters

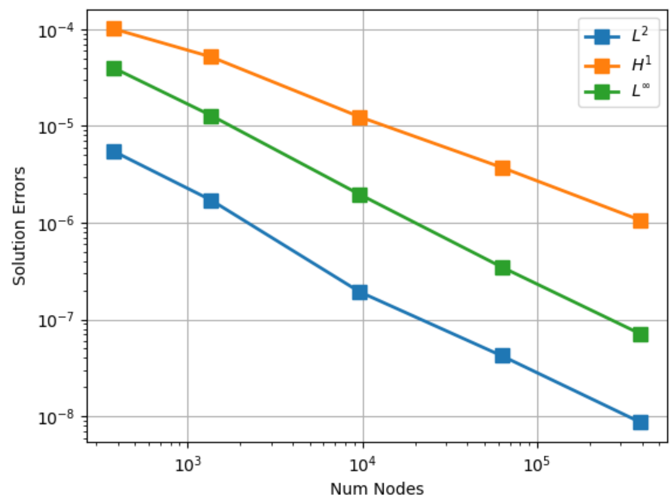
The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.9.4. Verification of Solution

Same as in Section 2.1.



Coarse Mesh



Error Norms

Figure 4.9-1.. Steady Tied Dash Contact: Tet10 Meshes

Table 4.9-1.. Steady Tied DASH Contact: Convergence Rates for Tet10 Meshes

Num Dofs	L^2	H^1	L^∞
1364	2.70	1.57	2.63
9663	3.35	2.19	2.88
62724	2.43	1.92	2.76
392046	2.60	2.06	2.61

For input decks see Appendix 12.3.9.

4.10. TRANSIENT TET4TET10 CONTACT

This problem tests basic transient heat conduction with contact in a 3D domain. The geometry consists of two halves of a unit cube meshed with Tet10 elements. The problem is solved using Tet4 interpolation and applying thermal contact at the common interface between the two domains.

4.10.1. Features Tested

Basic heat conduction on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.10.2. Boundary Conditions

Using a manufactured solution, Dirichlet boundary conditions are applied on all the non-contact exposed faces.

4.10.3. Material Parameters

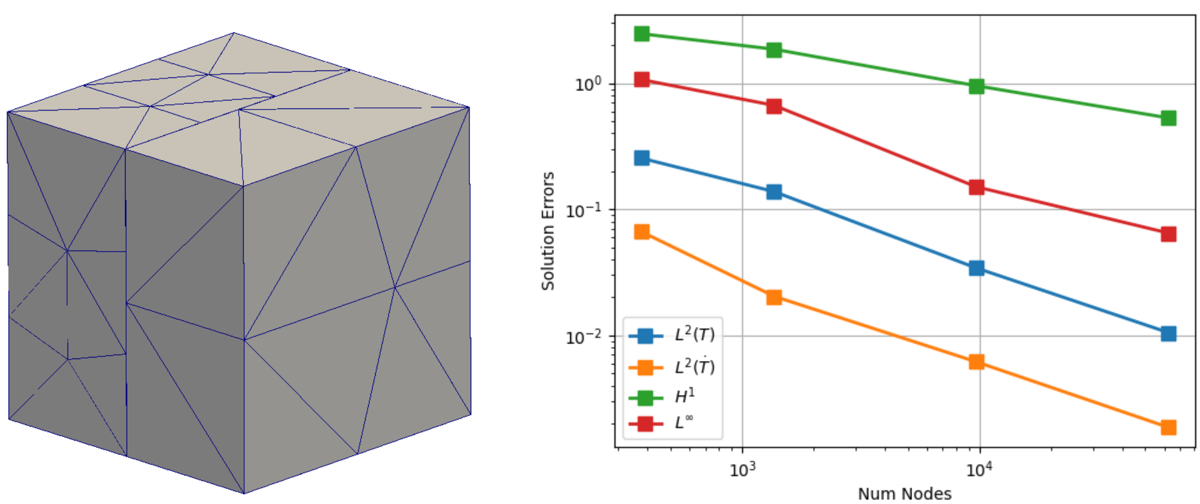
The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

4.10.4. Verification of Solution

A manufactured solution is chosen as

$$T(x,y,z,t) = (x-x^2)^2\,(y-y^2)^2\,(z-z^2)^2 m(t) + 1,$$
$$m(t) = 10^4\,\left[1. - \exp(-t) + t * \exp(-(t-1.0)*(t-1.0))\right];$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.



Coarse Mesh Error Norms

Figure 4.10-1.. Transient Tied Contact: Tet10 Meshes

For input decks see Appendix 12.3.10.

Table 4.10-1.. Transient Tied Contact: Convergence Rates for Tet10 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
1364	1.43	2.75	0.65	1.10
9663	2.13	1.83	1.02	2.28
62724	1.90	1.93	0.95	1.35

4.11. TRANSIENT TET10 CONTACT

This problem tests basic transient heat conduction with contact in a 3D domain. The geometry consists of two halves of a unit cube meshed with Tet10 elements. The problem is solved by applying thermal contact at the common interface between the two domains.

4.11.1. Features Tested

Basic heat conduction on Tet10 meshes; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.11.2. Boundary Conditions

Using a manufactured solution, Dirichlet boundary conditions are applied on all the non-contact exposed faces.

4.11.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

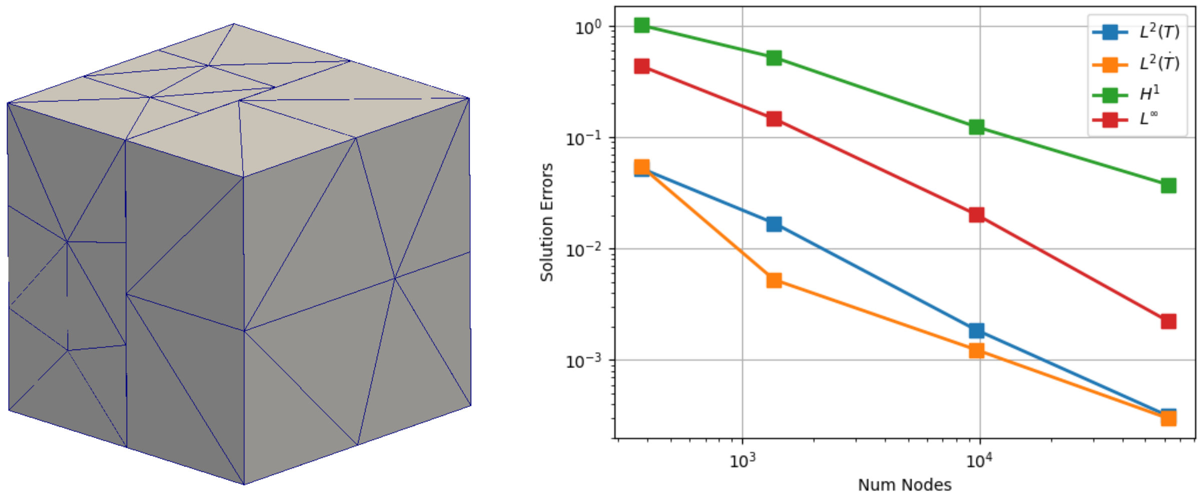
4.11.4. Verification of Solution

A manufactured solution is chosen as

$$T(x, y, z, t) = (x - x^2)^2\,(y - y^2)^2\,(z - z^2)^2 m(t) + 1,$$
$$m(t) = 10^4\,\left[1. - \exp(-t) + t * \exp(-(t - 1.0) * (t - 1.0))\right];$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For input decks see Appendix 12.3.11.



Coarse Mesh Error Norms

Figure 4.11-1.. Transient Tied Contact: Tet10 Meshes

Table 4.11-1.. Transient Tied Contact: Convergence Rates for Tet10 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
1364	2.63	5.44	1.55	2.53
9663	3.37	2.21	2.19	3.02
62724	2.86	2.28	1.93	3.55

4.12. TRANSIENT HEX8 TIED CONTACT

This problem tests transient heat conduction on a 3D domains with a nonconformal mesh between two blocks. Tied temperature (generalized contact) is used for matching the energy equation between nonconformal blocks. The geometry consists of a unit cube.

4.12.1. Features Tested

Transient heat conduction on Hex8 meshes; dirichlet, heat flux, and convective flux boundary conditions, Tied Contact, Nonconformal; constant source terms; heat flux and source term from Encore user subroutines.

4.12.2. Boundary Conditions

At surfaces 4 and 6, the temperature is prescribed as a constant value. On surfaces 3 and 5, a heat flux condition is prescribed using a sum of a constant heat flux and a heat flux from an Encore function (user subroutine). On surfaces 1 and 2, heat flux condition is prescribed using a sum of a convective flux boundary condition (with constant flux and convective coefficient) and a heat flux from an Encore function (user subroutine). Within the domain a source term is prescribed using a sum of a constant source and an Encore function (user subroutine). On the two interior surfaces connecting the nonconformal blocks (surfaces 7 and 8), a contact definition is defined as tied temperature.

4.12.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

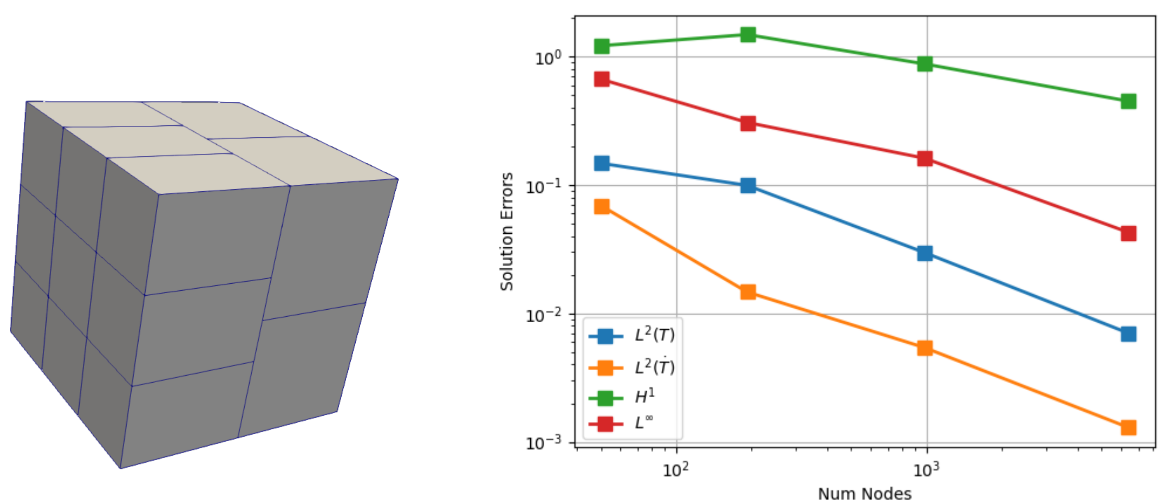
4.12.4. Verification of Solution

A manufactured solution is chosen as

$$T(x,y,z,t) = (x-x^2)^2\,(y-y^2)^2\,(z-z^2)^2 m(t) + 1,$$
$$m(t) = 10^4\,(1-\exp(-t) + t\,\exp(-(t-1)^2))$$

The source and heat flux user subroutines are chosen so that the solution satisfies the heat equation with the correct boundary conditions.

For each mesh, the errors in the temperature solution at final time are computed in the L^2 norm of T and $\dot{T}$, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, 2 and 1, respectively (within a tolerance).



Coarse Mesh Error Norms

Figure 4.12-1.. Tied Contact Transient Heat Conduction: Hex8 Meshes

Table 4.12-1.. Tied Contact Transient Heat Conduction: Convergence Rates for Hex8 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
192	0.88	3.46	-0.44	1.73
982	2.22	1.84	0.97	1.18
6419	2.31	2.29	1.07	2.13

4.13. TRANSIENT TET4 TIED CONTACT

This problem tests transient heat conduction and tied thermal contact in a 3D domain as in Section 2.7. The geometry consists of a unit cube that is split along the plane at $x = 0.5$.

4.13.1. Features Tested

Basic transient heat conduction on Tet4 meshes; non-conformal tied thermal contact; dirichlet, heat flux, and convective flux boundary conditions; constant source terms; heat flux and source term from Encore user subroutines.

4.13.2. Boundary Conditions

Identical to Section 2.7.

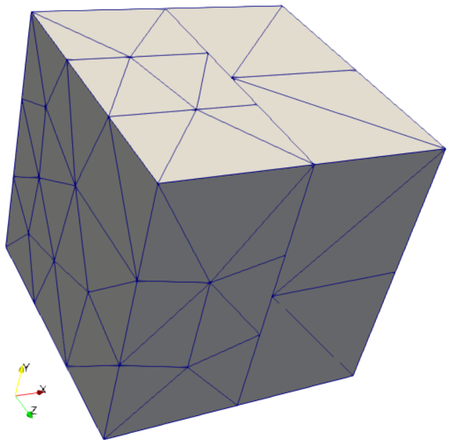
4.13.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

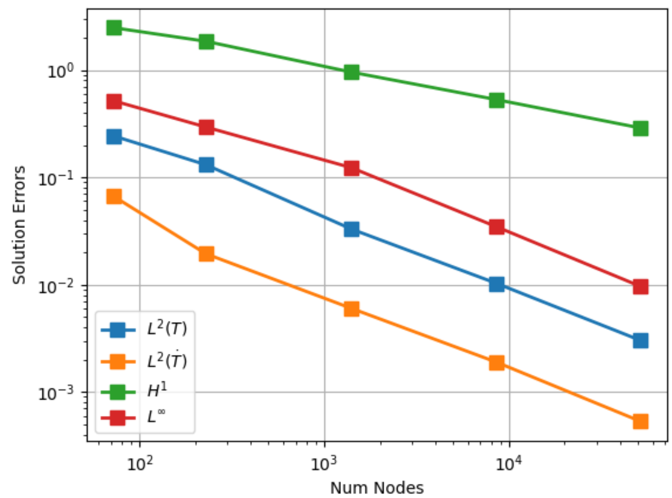
4.13.4. Verification of Solution

A manufactured solution is chosen as in Section 2.7.

For each mesh, the errors in the temperature solution at final time are computed in the L^2 norm of T and $\dot{T}$, L^∞ and H^1 norms. We see convergence rates for $\dot{T}$ that are slightly greater than two.



Coarse Mesh



Error Norms

Figure 4.13-1.. Transient Heat Conduction with Tied Contact: Tet4 Meshes

Table 4.13-1.. Transient Heat Conduction with Tied Contact: Convergence Rates for Tet4 Meshes

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
229	1.61	3.22	0.76	1.47
1402	2.29	1.94	1.10	1.44
8535	1.93	1.91	0.98	2.11
51622	2.05	2.12	1.02	2.14

This page intentionally left blank.

5. ELEMENT DEATH

5.1. CDFEM ELEMENT DEATH (HEAT FLUX)

This problem tests transient conduction and CDFEM element death using 2D and 3D domains. The geometry consists of a thick 1/4 cylindrical or 1/8 spherical shell.

5.1.1. Features Tested

Transient heat conduction, adaptive second order time integration (BDF2), CDFEM element death, temperature and heat flux boundary conditions, Tri3 and Tet4 meshes.

5.1.2. Boundary Conditions

On one surface, the exact solution is used to specify a time-varying temperature. At the other surface, an analytic heat flux is applied using the exact solution. The erosion of the volume from CDFEM element death causes the surface with the heat flux BC to gradually recede as the material is removed.

5.1.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

5.1.4. Verification of Solution

A manufactured solution T and exact source term S are chosen in 2D to be:

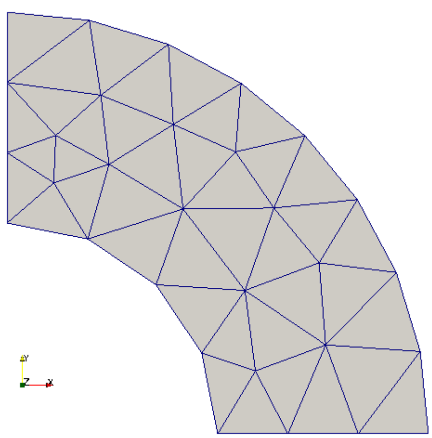
$$T(r,t) = \frac{\ln(r)}{\ln(2-t)}, \quad S(r,t) = \frac{\ln(r)}{(\ln(2-t))^2(2-t)}$$

and in 3D to be:

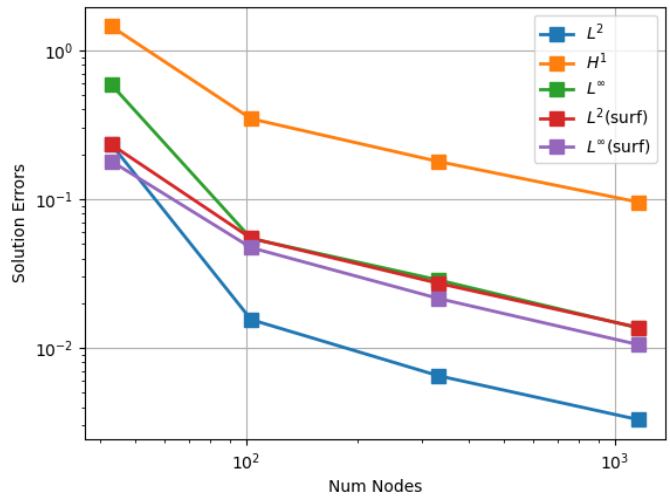
$$T(r,t) = (1+t)/r, \quad S(r,t) = 1/r.$$

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms over the volume, and in the L^2 and L^∞ norms over the outer surface. The test passes, only if the observed rates of convergence in these norms are one (within a tolerance). First order convergence is expected in this case, due to the nature of the coupling of the CDFEM mesh decomposition and the heat conduction solve.

5.1.5. Results: Tri3



Coarse Mesh



Error Norms

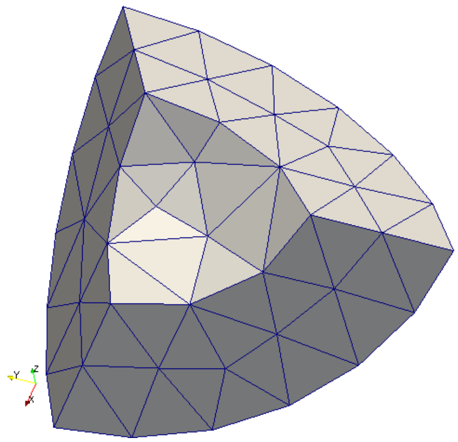
Figure 5.1-1.. CDFEM Element Death (Heat Flux): Tri3

Table 5.1-1.. CDFEM Element Death (Heat Flux): Convergence Rates for Tri3

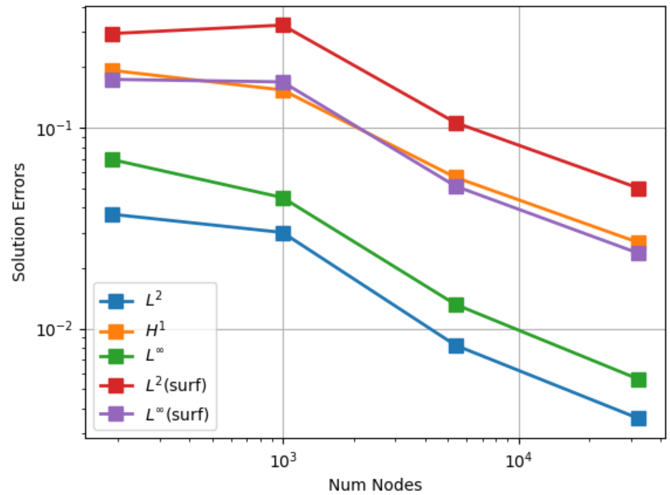
Num Dofs	L^2	H^1	L^∞	L^2(surf)	L^∞(surf)
103	3.92	2.06	3.44	2.10	1.93
332	1.25	0.96	0.93	1.01	1.13
1162	0.98	0.91	1.07	0.99	1.03

5.1.6. Results: Tet4

For input decks see Appendix 12.4.1.



Coarse Mesh



Error Norms

Figure 5.1-2.. CDFEM Element Death (Heat Flux): Tet4

Table 5.1-2.. CDFEM Element Death (Heat Flux): Convergence Rates for Tet4

Num Dofs	L^2	H^1	L^∞	L^2(surf)	L^∞(surf)
996	0.30	0.32	0.63	-0.14	0.04
5395	1.86	1.44	1.76	1.62	1.72
32358	1.21	1.08	1.24	1.08	1.11

5.2. 3D SPHERICAL SHELL ENCLOSURE

5.2.1. Problem Description

This problem tests transient conduction, enclosure radiation, and CDFEM element death. The initial geometry of this problem is a hollow sphere (block 2) inside and in contact with a second hollow sphere (block 1). The geometry is such that the solution maintains radial symmetry. The inner sphere decomposes at a specific failure temperature, resulting in a changing enclosure geometry.

5.2.2. Features Tested

Transient heat conduction, enclosure radiation, CDFEM element death, Tet4 meshes.

5.2.3. Boundary and Initial Conditions

The initial condition is a piecewise steady state temperature distribution defined below in (5.1). The boundary conditions specify the temperature T_4 at the outer surface (4) of the outer sphere and T_1 at the inner surface (1) of the inner sphere. The inner temperature T_1 will be gradually increased, while T_4 remains constant in time.

An enclosure is defined initially using the outer surface of the inner volume (surface 2 of block 2) and the inner surface of the outer volume (surface 3 of block 1). The erosion of the inner volume (block 2) from CDFEM element death causes surface 2 to gradually recede as the material within block 2 is removed.

Dimensions are defined in Table 5.2-1.

Table 5.2-1.. Dimensions of problem

radius of surface_1	r_1	0.01
radius of surface_2	r_2	0.02
radius of surface_3	r_3	0.03
radius of surface_4	r_4	0.04

5.2.4. Material Parameters

Material properties are shown in Table 5.2-2.

Table 5.2-2.. Material properties

Thermal conductivity	κ	1.0
Density	ρ	7682.0
Specific heat	C_p	10.0
emissivity (inner)	ϵ_2	0.6
emissivity (outer)	ϵ_3	0.7
Stefan-Boltzmann constant	σ	5.6704e-8
failure temperature (block 2)	T_c	867.011674920813

5.2.5. Verification of Solution

The solution after failure occurs is specified using inner and outer temperature solutions of the form:

$$T_i(r) \equiv T_1 + (T_c - T_1)\frac{1/r - 1/r_1}{1/r_2 - 1/r_1}, \quad r_1 \le r \le r_2, \tag{5.1}$$

$$T_o(r) \equiv C_o + (T_4 - C_o)\frac{1/r - 1/r_3}{1/r_4 - 1/r_3}, \quad r_3 \le r \le r_4 \tag{5.2}$$

Here all parameters are known except r_2 and C_o, which will vary with time. The initial value of r_2 is given in Table 5.2-1; the initial value of C_o is chosen to satisfy the enclosure radiation equilibrium equations below.

To complete the solution, we now derive a system of two nonlinear equations to solve for r_2 and C_o. These are the energy balances on the outer and inner enclosure surfaces, given by

$$R_2 \equiv q_2 - \sigma\epsilon_2 T_2^4 + \epsilon_2(F_{22}J_2 + F_{23}J_3) \tag{5.3}$$
$$R_3 \equiv -q_3 - \sigma\epsilon_3 T_3^4 + \epsilon_3(F_{32}J_2 + F_{33}J_3) \tag{5.4}$$

where the three terms in each equation represent fluxes from conduction, radiative emission, and radiative reflection. The conductive fluxes are defined by Fourier's law as

$$q_2 = -\kappa_2 \frac{\partial T_i}{\partial r}|_{r=r_2} = \kappa_2 \frac{T_c - T_1}{r_1^2(1/r_2 - 1/r_1)} \tag{5.5}$$
$$q_3 = -\kappa_3 \frac{\partial T_o}{\partial r}|_{r=r_3} = \kappa_3 \frac{T_4 - C_o}{r_3^2(1/r_4 - 1/r_3)} \tag{5.6}$$

The surface temperatures are

$$T_2 \equiv T_i|_{r=r_2} = T_c, \quad T_3 \equiv T_o|_{r=r_3} = C_o$$

The radiosities are obtained by solving the linear system for enclosure radiation

$$\begin{bmatrix} 1-(1-\epsilon_2)F_{22} & -(1-\epsilon_2)F_{23} \\ -(1-\epsilon_3)F_{32} & 1-(1-\epsilon_3)F_{33} \end{bmatrix} \begin{bmatrix} J_2 \\ J_3 \end{bmatrix} = \begin{bmatrix} \sigma\epsilon_2 T_2^4 \\ \sigma\epsilon_3 T_3^4 \end{bmatrix}$$

to obtain

$$\begin{bmatrix} J_2 \\ J_3 \end{bmatrix} = \frac{1}{\mathrm{a}} \begin{bmatrix} 1-(1-\epsilon_3)F_{33} & (1-\epsilon_2)F_{23} \\ (1-\epsilon_3)F_{32} & 1-(1-\epsilon_2)F_{22} \end{bmatrix} \begin{bmatrix} \sigma\epsilon_2 T_2^4 \\ \sigma\epsilon_3 T_3^4 \end{bmatrix}$$

where a is the determinant

$$a = (1-(1-\epsilon_2)F_{22})(1-(1-\epsilon_3)F_{33}) - (1-\epsilon_2)F_{23}(1-\epsilon_3)F_{32}$$

The viewfactor coefficients F_{ij} are given by

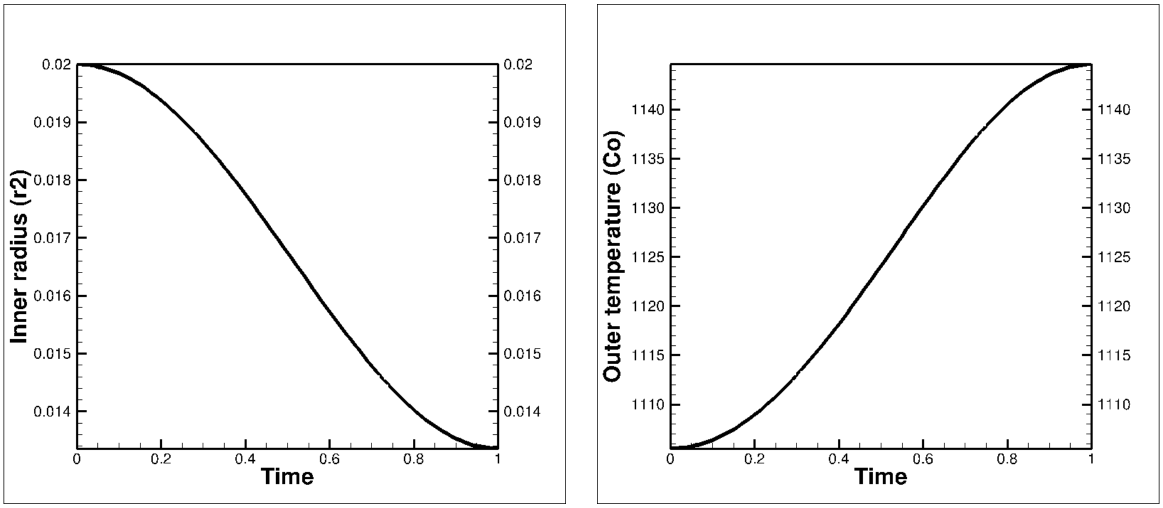
$$F_{22} = 0, \quad F_{23} = 1, \quad F_{32} = (r_2/r_3)^2, \quad F_{33} = 1 - F_{32}$$

The specific function we choose for $T_1(t)$ is

$$T_1(t) \equiv T_1 + 400(1 - \cos(\pi t))/2$$

The time histories of r_2 and C_o are shown in Figure 5.2-1.

In order to derive the source term, the time derivatives of r_2 and C_o are computed once the pair of nonlinear equations is solved using Newton's method. Since the spatial part of the piecewise solution is



$r_2(t)$ $C_o(t)$

Figure 5.2-1.. Evolution of parameters r_2 and C_o.

harmonic, the source terms become just $\rho c_p \partial_t T$, where

$$\partial_t T_i \equiv \dot{T}_1 + (T_c - \dot{T}_1)\frac{1/r - 1/r_1}{1/r_2 - 1/r_1} + (T_c - T_1)\frac{\dot{r}_2(1/r - 1/r_1)}{r_2^2(1/r_2 - 1/r_1)^2}, \quad r_1 \le r \le r_2, \tag{5.7}$$

$$\partial_t T_o \equiv \dot{C}_o(1 - \frac{1/r - 1/r_3}{1/r_4 - 1/r_3}), \quad r_3 \le r \le r_4 \tag{5.8}$$

5.2.6. Results

Results are presented running the problem on three meshes up to time $t = 0.9$.

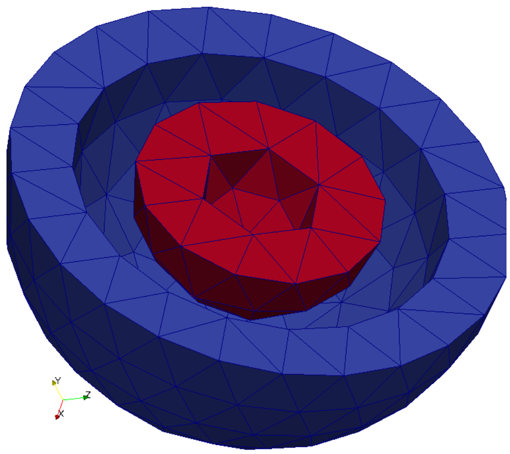
Table 5.2-3.. Convergence Rates at $t = 0.9$

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	L^∞	H^1
4289	2.09	1.93	0.96	1.06
25590	2.26	1.61	1.21	1.05

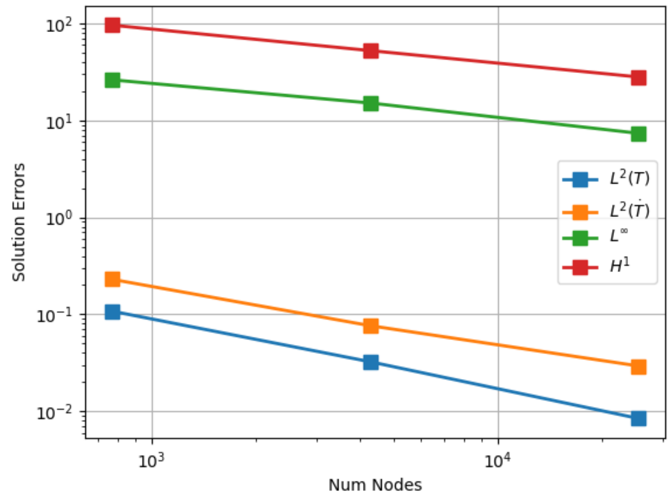
For input decks see Appendix 12.4.2.

5.3. STANDARD ELEMENT DEATH (HEAT FLUX)

This problem tests transient conduction with standard element death on a 2D square domain than is essentially a 1D problem.



Coarse Mesh



Error Norms ($t = 0.9$)

5.3.1. Features Tested

Transient heat conduction, adaptive second order time integration (BDF2), standard element death, temperature and heat flux boundary conditions, Tri3, Hex8 and Quad4 meshes.

5.3.2. Boundary Conditions

On one surface, the exact solution is used to specify a time-varying temperature. At the other surface, an analytic heat flux is applied using the exact solution. The erosion of the volume from element death causes the surface with the heat flux BC to recede element by element as the material is removed.

5.3.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

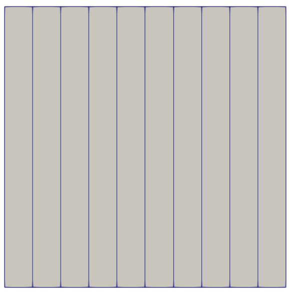
5.3.4. Verification of Solution

A manufactured solution T and exact source term S are chosen to be:

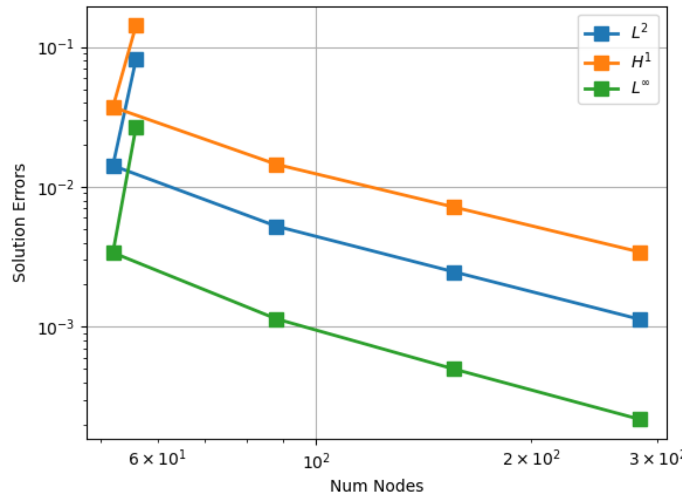
$$T(r,t) = \exp(t - x).$$

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms over the area. The test passes, only if the observed rates of convergence in these norms are one (within a tolerance). First order convergence is expected in this case.

5.3.5. Results: 1D Hex8



Coarse Mesh



Error Norms

Figure 5.3-1.. Element Death (Heat Flux): Hex8

Table 5.3-1.. Element Death (Heat Flux): Convergence Rates for Hex8

Num Dofs	Var1	Var2	Var3
52	nan	nan	nan
88	1.44	1.36	1.58
156	1.09	1.02	1.19
284	1.13	1.07	1.19

5.3.6. Results: 1D Quad4

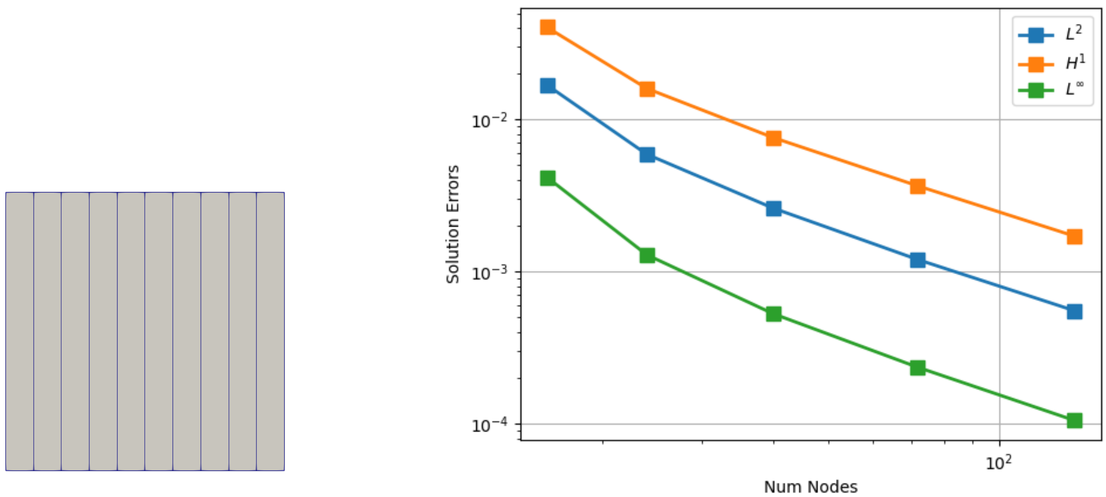
5.3.7. Results: 1D Tri3

5.3.8. Results: 2D Quad4

This problem tests transient conduction with standard element death on a 2D quarter slice of an annulus.

5.3.9. Features Tested

Transient heat conduction, fixed first order time integration, standard element death, Quad4 mesh.



Coarse Mesh   Error Norms

Figure 5.3-2.. Element Death (Heat Flux): Quad4

Table 5.3-2.. Element Death (Heat Flux): Convergence Rates for Quad4

Num Dofs	Var1	Var2	Var3
24	1.51	1.34	1.70
40	1.18	1.07	1.28
72	1.12	1.06	1.17
136	1.12	1.09	1.16

5.3.10. Boundary Conditions

On one surface, the exact solution is used to specify a time-varying temperature. On the other surfaces, the exact source solution is provided as the flux boundary condition. The erosion of the volume from element death is caused by having a minimum nodal value of temperature less than 1.

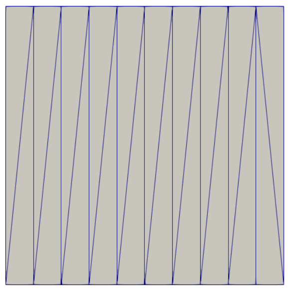
5.3.11. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

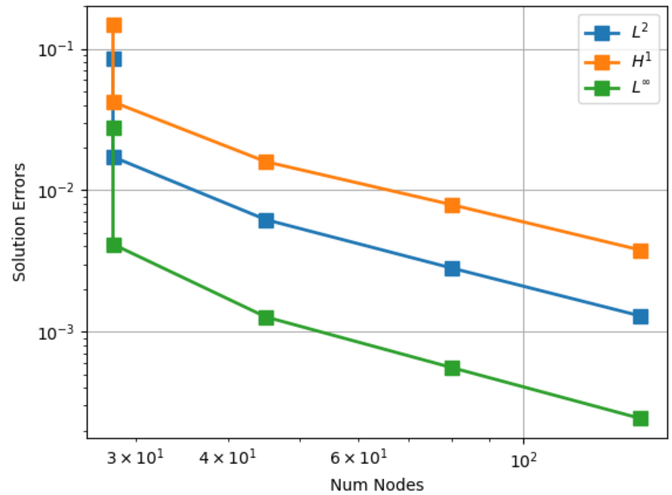
5.3.12. Verification of Solution

A manufactured solution T and exact source term S are chosen to be:

$$T(r,t) = \ln(\sqrt{x^2+y^2})(1/\ln(2-t)).$$



Coarse Mesh



Error Norms

Figure 5.3-3.. Element Death (Heat Flux): Tri3

Table 5.3-3.. Element Death (Heat Flux): Convergence Rates for Tri3

Num Dofs	Var1	Var2	Var3
28	nan	nan	nan
45	1.48	1.41	1.70
80	1.13	1.01	1.19
144	1.13	1.07	1.20

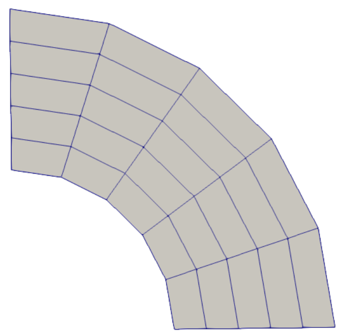
For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms over the area. The test passes, only if the observed rates of convergence in these norms are one (within a tolerance). First order convergence is expected in this case.

5.3.13. Results: 3D Hex8

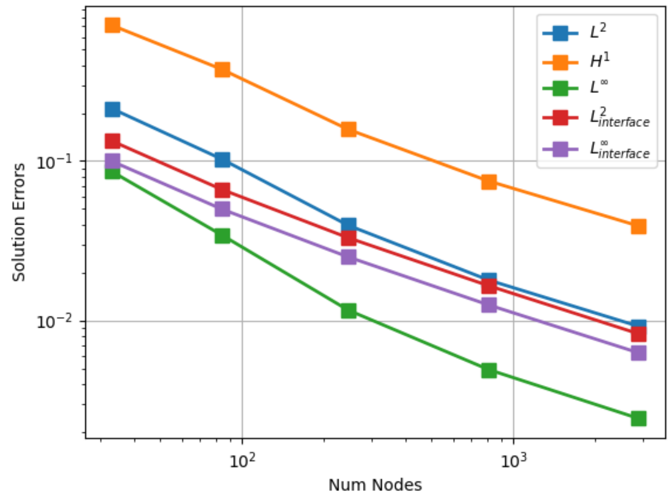
This problem evaluates transient conduction with standard element death on a 3D quarter of a hollow sphere geometry.

5.3.14. Features Tested

Transient heat conduction, fixed first order time integration, standard element death, Hex8 mesh.



Coarse Mesh



Error Norms

Figure 5.3-4.. Element Death (Heat Flux): Quad4

Table 5.3-4.. 2D Element Death (Heat Flux): Convergence Rates for Quad4

Num Dofs	Var1	Var2	Var3	Var4	Var5
84	1.56	1.37	1.97	1.50	1.48
246	1.79	1.62	2.02	1.30	1.29
810	1.33	1.25	1.45	1.17	1.16
2898	1.05	1.02	1.11	1.09	1.09

5.3.15. Boundary Conditions

On surface 2, the exact solution is used to specify a time-varying temperature. On all remaining surfaces, a heat flux boundary condition is imposed with a flux time function specified. The erosion of the volume from element death is caused by having a maximum nodal value of temperature greater than 1.

5.3.16. Material Parameters

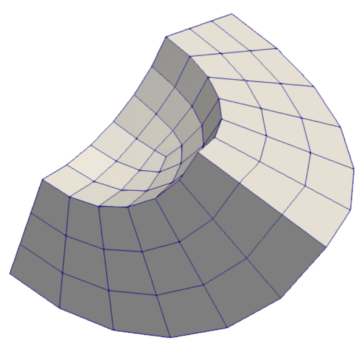
The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks.

5.3.17. Verification of Solution

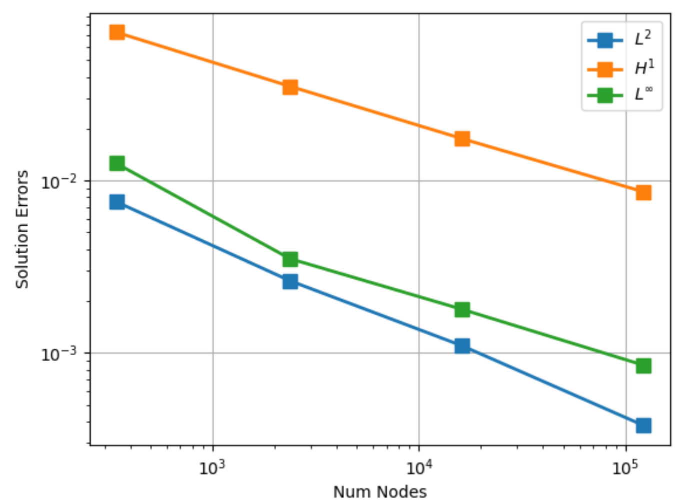
A manufactured solution T and exact source term S are chosen to be:

$$T(r,t) = \frac{1+t}{\sqrt{x^2+y^2+z^2}}.$$

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms over the area. The observed rates of convergence in these norms are one (within a tolerance). First order convergence is expected in this case.



Coarse Mesh



Error Norms

Figure 5.3-5.. Element Death (Heat Flux): Hex8

Table 5.3-5.. Element Death (Heat Flux): Convergence Rates for Hex8

Num Dofs	L^2	H^1	L^∞
2382	1.64	1.12	1.97
16214	1.36	1.10	1.06
122892	1.57	1.05	1.11

6. TIME INTEGRATION

6.1. ADAPTIVE TIME INTEGRATION

This problem tests the various implicit time integrators using both fixed and adaptive time stepping. The integrators are first order (Backward Euler), second order (Crank-Nicolson) and BDF2. The geometry is a 2D square.

6.1.1. Features Tested

Transient heat conduction, time integrators, adaptive time stepping, polynomial temperature dependence of density and thermal conductivity.

6.1.2. Boundary Conditions

The boundary conditions are prescribed at the nodes using the analytic solution. The initial condition is specified using an Encore function evaluated at the nodes.

6.1.3. Material Parameters

The specific heat is constant. The density and thermal conductivity are linear polynomials in the temperature.

6.1.4. Verification of Solution

A manufactured solution is chosen as

$$T(x, y, t) = \sin(C_1 t) + 2x\cos(C_2 t) + 3y\sin(C_3 t) + 4xy\cos(C_4 t) + 5x^2\sin(C_5 t) + 6y^2\cos(C_6 t)$$

which requires a source term. This solution is designed to have a non-trivial time-dependence using constants:

$$C_1 = \pi, \quad C_2 = 2\pi, \quad C_3 = 3\pi, \quad C_4 = \pi, \quad C_5 = 2.5\pi, \quad C_6 = 0.5\pi$$

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The L^2 error in the temperature time derivative is also computed. The test passes, only if the observed rates of convergence in these norms are 1 for H^1 and 2 for all other norms (within a tolerance).

Because the adaptive meshes use less time steps, we use time step size instead of mesh size for estimation of the convergence rates. We also include the L^2 error in the time derivative of the temperature.

For input decks see Appendix 12.5.1.

6.1.5. Results: First Order Fixed

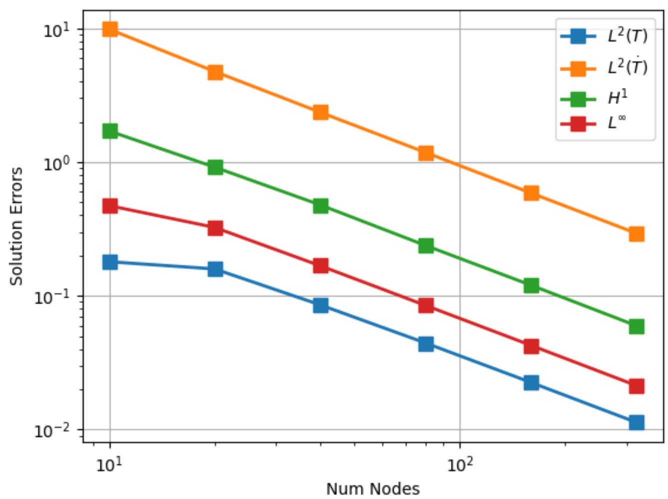


Figure 6.1-1.. Adaptive Time Integration: Errors for First Order Fixed

Table 6.1-1.. Adaptive Time Integration: Convergence Rates for First Order Fixed

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
20	0.18	1.05	0.89	0.54
40	0.89	1.01	0.94	0.95
80	0.95	1.01	1.01	0.99
160	0.98	1.00	0.99	0.99
320	0.99	1.00	1.00	1.00

6.1.6. Results: First Order Adaptive

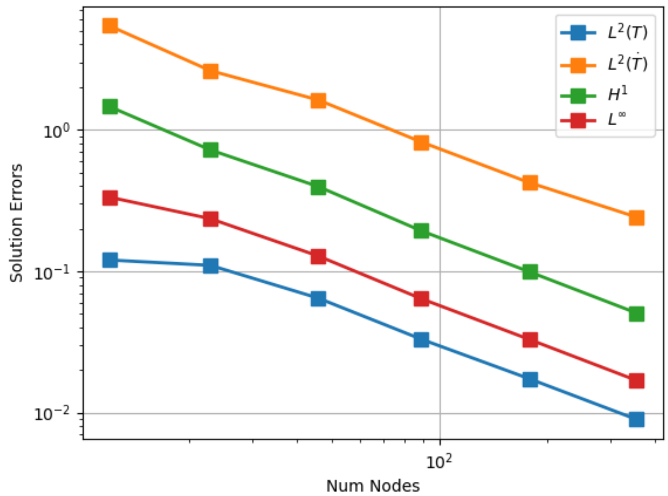


Figure 6.1-2.. Adaptive Time Integration: Errors for First Order Adaptive

Table 6.1-2.. Adaptive Time Integration: Convergence Rates for First Order Adaptive

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
23	0.13	1.13	1.09	0.54
46	0.77	0.68	0.86	0.87
89	1.01	1.04	1.09	1.05
178	0.93	0.95	0.96	0.95
355	0.96	0.81	0.98	0.97

6.1.7. Results: Second Order Fixed

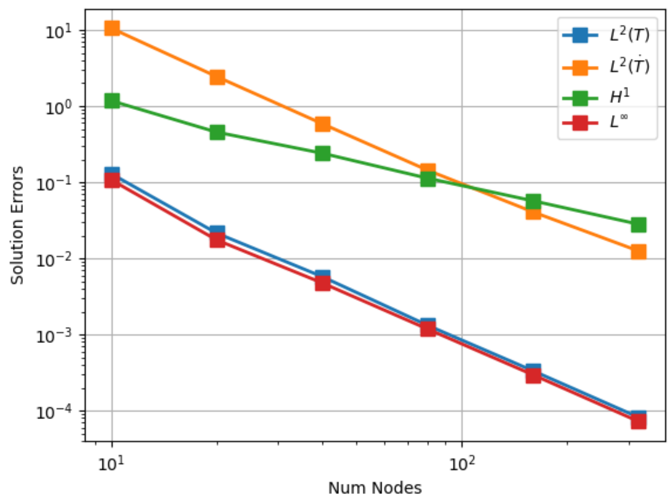


Figure 6.1-3.. Adaptive Time Integration: Errors for Second Order Fixed

Table 6.1-3.. Adaptive Time Integration: Convergence Rates for Second Order Fixed

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
20	2.59	2.11	1.37	2.62
40	1.90	2.06	0.92	1.88
80	2.13	2.03	1.10	2.02
160	1.98	1.82	0.99	2.00
320	2.03	1.71	1.02	2.04

6.1.8. Results: Second Order Adaptive

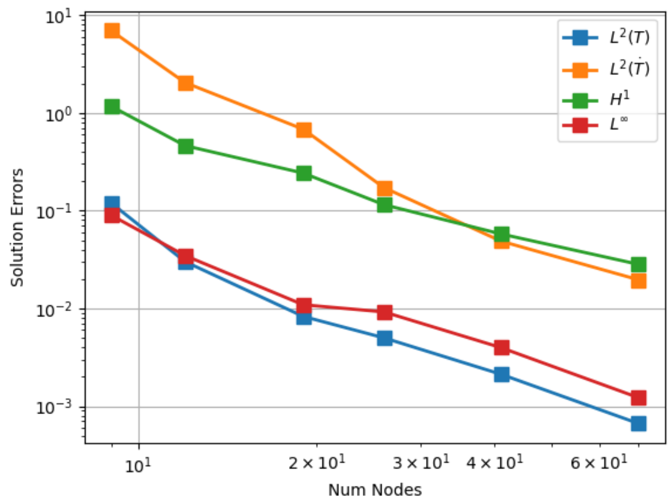


Figure 6.1-4.. Adaptive Time Integration: Errors for Second Order Adaptive

Table 6.1-4.. Adaptive Time Integration: Convergence Rates for Second Order Adaptive

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
12	4.81	4.26	3.24	3.31
19	2.81	2.39	1.41	2.52
26	1.61	4.38	2.38	0.54
41	1.88	2.77	1.51	1.85
70	2.17	1.70	1.34	2.21

6.1.9. Results: BDF2 Fixed

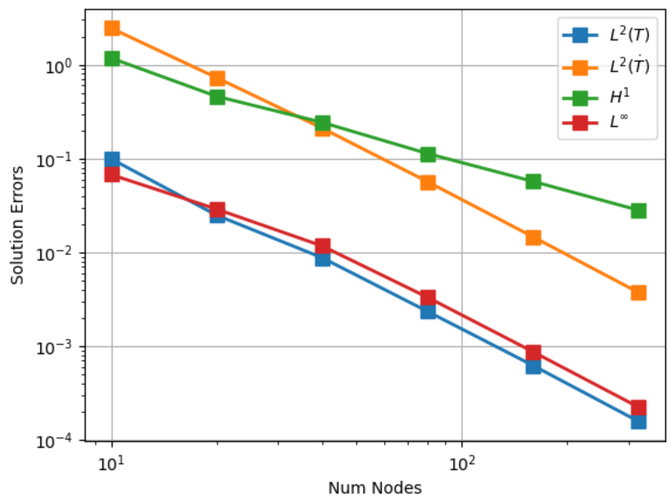


Figure 6.1-5.. Adaptive Time Integration: Errors for BDF2 Fixed

Table 6.1-5.. Adaptive Time Integration: Convergence Rates for BDF2 Fixed

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
20	2.00	1.75	1.36	1.22
40	1.51	1.79	0.92	1.30
80	1.90	1.90	1.10	1.82
160	1.92	1.95	0.99	1.93
320	1.98	1.98	1.02	1.96

6.1.10. Results: BDF2 Adaptive

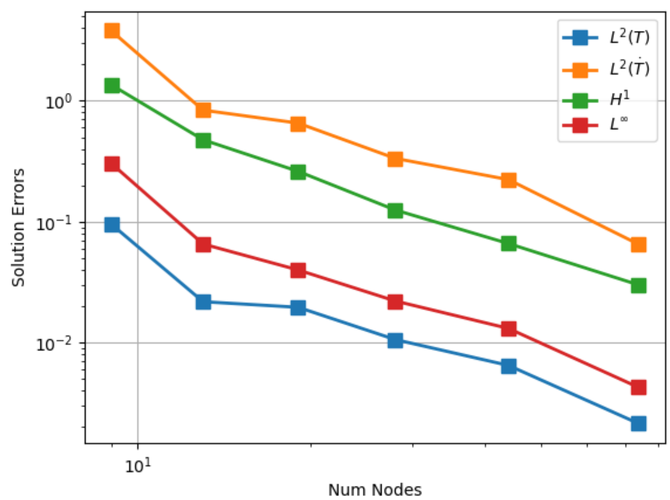


Figure 6.1-6.. Adaptive Time Integration: Errors for BDF2 Adaptive

Table 6.1-6.. Adaptive Time Integration: Convergence Rates for BDF2 Adaptive

Num Dofs	$L^2(T)$	$L^2(\dot{T})$	H^1	L^∞
13	4.03	4.12	2.86	4.19
19	0.27	0.65	1.58	1.30
28	1.59	1.73	1.90	1.52
44	1.09	0.90	1.42	1.16
74	2.13	2.37	1.51	2.16

This page intentionally left blank.

7. ENCLOSURE RADIATION

7.1. 2D CYLINDRICAL SHELL ENCLOSURE

7.1.1. Problem Description

This problem tests steady state coupled conduction and enclosure radiation. The geometry of this problem is a hollow cylinder (block 2) inside a second hollow cylinder (block 1), which is a radially symmetric problem.

7.1.2. Features Tested

Basic heat conduction, enclosure radiation, Quad4/Tri3 meshes.

7.1.3. Boundary Conditions

The boundary conditions specify the temperature of the outer surface of the outer sphere ($T(r_4) = T_4$) and the inner surface of the inner sphere ($T(r_1) = T_1$).

The problem is steady state but is initialized with a constant temperature of 300 in both blocks. The inner surface temperature T_1 is set to 300. The outer surface temperature T_4 is set to 1300.

Dimensions are defined in Table 7.1-1.

Table 7.1-1.. Dimensions of problem

radius of surface_1	r_1	0.01
radius of surface_2	r_2	0.02
radius of surface_3	r_3	0.03
radius of surface_4	r_4	0.04

7.1.4. Material Parameters

Material properties are shown in Table 7.1-2.

Table 7.1-2.. Material properties

Thermal conductivity (block_1)	κ_1	2.0
Thermal conductivity (block_2)	κ_2	0.35
Density	ρ	1.0
Specific heat	C_p	1.0
emissivity (surface_2)	ϵ_2	0.50
emissivity (surface_3)	ϵ_3	0.55
Stefan-Boltzmann constant	σ	5.6704e-8

7.1.5. Verification of Solution

In cylindrical coordinates, the temperature is independent of θ and z. Integrating this equation twice with respect to the radius r, we obtain the general solution in either hollow cylinder to be

$$T(r) = C_1 \log(r) + C_2,$$

for arbitrary constants C_1 and C_2. We will use $r_i, i = 1, \ldots, 4$ to denote the location of the four surfaces of constant r, numbered from inside to outside. Unless specified otherwise, we will use these subscripts for other quantities which are evaluated at one of the four surfaces.

Including the boundary conditions into the solution allows us to eliminate two constants and gives

$$T_{inner}(r) = T_1 + c_I \log(r/r_1) \text{ for } r_1 < r < r_2 \tag{7.1}$$

$$T_{outer}(r) = T_4 + c_O \log(r/r_4) \text{ for } r_3 < r < r_4 \tag{7.2}$$

To solve for c_I and c_O we compute the temperatures at the enclosure surfaces r_2 and r_3, defined as $T_2 = T_{inner}(r_2)$ and $T_3 = T_{outer}(r_3)$:

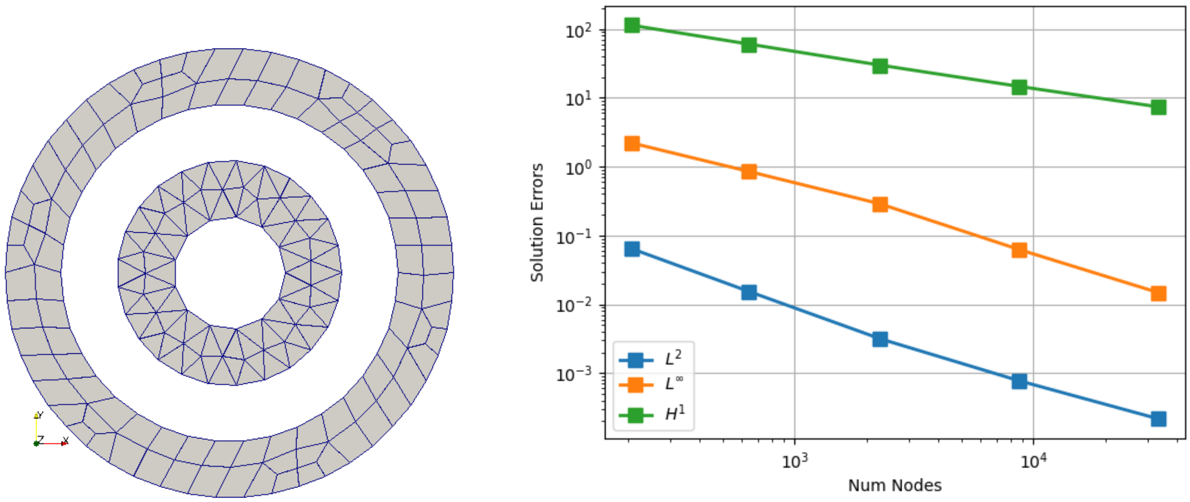
7.1.6. Results

The exact temperatures at the enclosure surfaces (to six digits precision) are $T_2 = 444.7977$ and $T_3 = 956.5915$. From these values we can compute the values of c_O and c_I and thus the exact solution.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

These optimal rates are observed in this test.

For input decks see Appendix 12.6.1.



Coarse Mesh   Error Norms

Figure 7.1-1.. Enclosure Radiation 2D

Table 7.1-3.. Enclosure Radiation 2D: Convergence Rates

Num Dofs	L^2	L^∞	H^1
640	2.54	1.68	1.11
2276	2.50	1.72	1.11
8673	2.10	2.28	1.07
33500	1.90	2.14	1.02

7.2. 2D ANNULAR ENCLOSURE

7.2.1. Problem Description

This problem tests steady state coupled conduction and enclosure radiation. The geometry is an annulus with a crack.

7.2.2. Features Tested

Basic heat conduction, enclosure radiation, Tri3 mesh.

7.2.3. Boundary Conditions

The outer and crack boundary conditions are prescribed at the nodes using the analytic solution. The inner boundary uses an enclosure boundary condition.

7.2.4. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant.

7.2.5. Verification of Solution

The manufactured solution is

$$
\begin{aligned}
J(\theta) &= k_1 + k_2\sqrt{\varepsilon}\cos\left(\frac{\sqrt{\varepsilon}\theta}{2}\right) / \sin\left(\sqrt{\frac{\varepsilon}{\pi}}2\right), \\
H(\theta) &= k_1 + k_2(\frac{\sqrt{\varepsilon}}{1-\varepsilon})\left(\cos\left(\frac{\sqrt{\varepsilon}\theta}{2}\right) / \sin\left(\frac{\sqrt{\varepsilon}\pi}{2}\right) - \sqrt{\varepsilon}\cos\left(\frac{\theta}{2}\right)\right), \\
q(\theta) &= J(\theta) - H(\theta), \\
\beta(\theta) &= \left(\frac{k_1 + k_2\cos\left(\frac{\theta}{2}\right)}{\sigma}\right)^{1/4}, \\
T(r,\theta) &= r\beta(\theta) + (r - r_{\text{cyl}})\left(\frac{q(\theta)}{\kappa} - \beta(\theta)\right),
\end{aligned}
$$

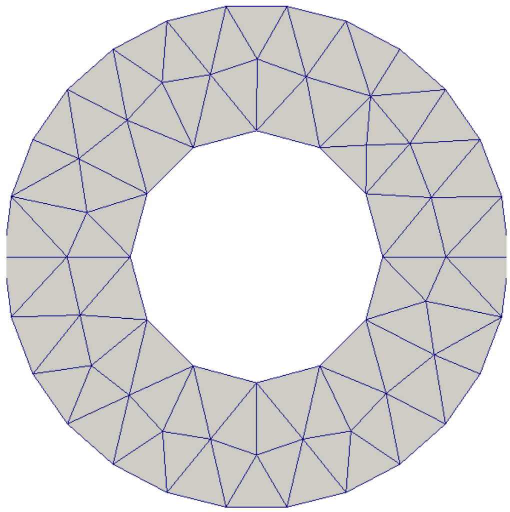
where J is the radiosity, H is the irradiance, q is the flux, and

$$
\begin{aligned}
\sigma &= 5.6704 \times 10^{-8}, \\
\kappa &= 1, \\
r_{\text{cyl}} &= 1, \\
\varepsilon &= 0.9, \\
k_1 &= 8000, \\
k_2 &= 400.
\end{aligned}
$$

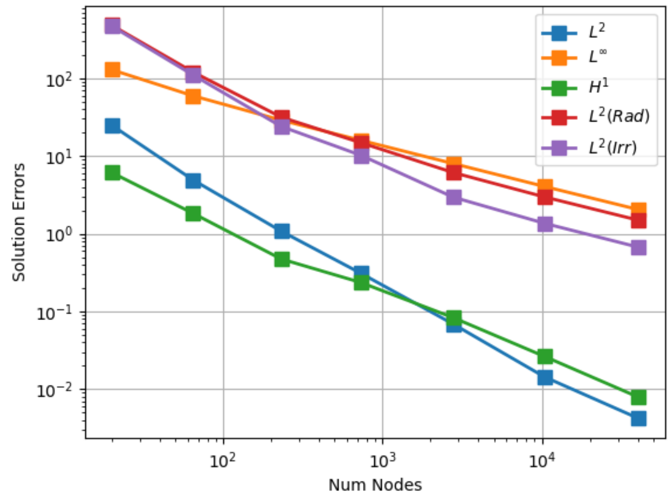
For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance). Additionally, the errors in the radiosity and irradiance are computed in the L^2 norms and be 1 (within a tolerance).

These optimal rates are observed in this test.

For input decks see Appendix 12.6.2.



Coarse Mesh



Error Norms

Figure 7.2-1.. 2D Full Enclosure Radiation

Table 7.2-1.. 2D Full Enclosure Radiation: Convergence Rates

Num Dofs	L^2	H^1	L^∞	$L^2(Rad)$	$L^2(Irr)$
65	2.76	1.31	2.05	2.36	2.46
232	2.38	1.13	2.13	2.10	2.40
734	2.18	1.04	1.21	1.32	1.49
2788	2.26	1.03	1.56	1.32	1.85
10415	2.37	1.03	1.74	1.10	1.18
40529	1.81	1.01	1.78	1.01	1.04

7.3. 3D SPHERICAL SHELL ENCLOSURE

7.3.1. Problem Description

This problem tests steady state coupled conduction and enclosure radiation. The geometry of this problem is a hollow sphere (block 2) inside a second hollow sphere (block 1), which is a radially symmetric problem.

7.3.2. Features Tested

Basic heat conduction, enclosure radiation, Hex8 meshes.

7.3.3. Boundary Conditions

The boundary conditions specify the temperature of the outer surface of the outer sphere ($T(r_4) = T_4$) and the inner surface of the inner sphere ($T(r_1) = T_1$).

The problem is steady state but is initialized with a constant temperature of 300 in both blocks. The inner surface temperature T_1 is set to 300. The outer surface temperature T_4 is set to 1300.

Dimensions are defined in Table 7.3-1.

Table 7.3-1.. Dimensions of problem

radius of surface_1	r_1	0.01
radius of surface_2	r_2	0.02
radius of surface_3	r_3	0.03
radius of surface_4	r_4	0.04

7.3.4. Material Parameters

Material properties are shown in Table 7.3-2.

Table 7.3-2.. Material properties

Thermal conductivity (block_1)	κ_1	2.0
Thermal conductivity (block_2)	κ_2	0.35
Density	ρ	1.0
Specific heat	C_p	1.0
emissivity (surface_2)	ϵ_2	0.50
emissivity (surface_3)	ϵ_3	0.55
Stefan-Boltzmann constant	σ	5.6704e-8

7.3.5. Verification of Solution

In spherical coordinates, the temperature is independent of θ and ϕ. Integrating this equation twice with respect to the radius r, we obtain the general solution in either hollow sphere to be

$$T(r) = C_1 r^{-1} + C_2,$$

for arbitrary constants C_1 and C_2. We will use $r_i, i = 1, \ldots, 4$ to denote the location of the four surfaces of constant r, numbered from inside to outside. Unless specified otherwise, we will use these subscripts for other quantities which are evaluated at one of the four surfaces.

Including the boundary conditions into the solution allows us to eliminate two constants and gives

$$T_{inner}(r) = T_1 + c_I\left(\frac{1}{r} - \frac{1}{r_4}\right) \text{ for } r_1 < r < r_2 \tag{7.3}$$

$$T_{outer}(r) = T_4 + c_O\left(\frac{1}{r} - \frac{1}{r_1}\right) \text{ for } r_3 < r < r_4 \tag{7.4}$$

To solve for c_I and c_O we compute the temperatures at the enclosure surfaces r_2 and r_3, defined as $T_2 = T_{inner}(r_2)$ and $T_3 = T_{outer}(r_3)$:

$$T_2 = T_1 + c_I\left(\frac{1}{r_2} - \frac{1}{r_4}\right) \tag{7.5}$$

$$T_3 = T_4 + c_O\left(\frac{1}{r_3} - \frac{1}{r_1}\right) \tag{7.6}$$

The fluxes at the surfaces between the two hollow spheres are

$$q_2 = \left(-\kappa \left.\frac{\partial T}{\partial r}\right|_{r=r_3}\right) \cdot \mathbf{n} = \frac{\kappa_1 c_I}{r_2^2}$$

$$q_3 = \left(-\kappa \left.\frac{\partial T}{\partial r}\right|_{r=r_2}\right) \cdot \mathbf{n} = \frac{\kappa_2 c_O}{r_3^2}$$

Here we have used κ_1 and κ_2 to denote the thermal conductivity of the inner and outer blocks, respectively.

These normal conductive fluxes are included in the total energy balance at the enclosure surfaces using the radiative transport equations (for grey diffuse surfaces):

$$q_2 = \sigma\epsilon_2 T_2^4 - \epsilon_2 \sum_j F_{2j} J_j$$

$$q_3 = \sigma\epsilon_3 T_3^4 - \epsilon_3 \sum_j F_{3j} J_j$$

where σ is the Stefan Boltzmann constant, ϵ is the emissivity, F_{ij} is the geometric viewfactor of surface i with respect to surface j and J_j is the radiosity for surface j.

The viewfactor coefficient F_{ij} is the fraction of energy that leaves surface i and arrives at surface j. For this geometric setup, no point on the inner surface at r_2 can "see" itself (no straight line can be drawn from a point on its surface onto itself) and so $F_{22} = 0$. By viewfactor reciprocity

$$\sum_j F_{ij} = 1$$

we must have $F_{23} = 1$. The outer-to-inner view factor F_{32} can be computed analytically to be

$$F_{32} = \frac{r_2^2}{r_3^2}$$

and again by viewfactor reciprocity

$$F_{33} = 1 - F_{32} = 1 - \frac{r_2^2}{r_3^2}$$

The system of equations that must be solved for the radiosities at the inner and outer surfaces is given by

$$J_2 = \epsilon_2 \sigma T_2^4 + (1 - \epsilon_2)[F_{22} J_2 + F_{23} J_3]$$
$$J_3 = \epsilon_3 \sigma T_3^4 + (1 - \epsilon_3)[F_{32} J_2 + F_{33} J_3]$$

Solving this system of equations, we can write J_2 and J_3 in terms of temperature, and plug this back into the equation for the surface flux. We then get a system of two nonlinear equations to solve for T_2 and T_3, the temperatures of the adjacent surfaces without Dirichlet boundary conditions. For our given set of parameters, these equations are solved iteratively in Matlab using the fsolve function.

7.3.6. Results

The exact temperatures at the enclosure surfaces (to six digits precision) are $T_2 = 564.783$ and $T_3 = 1047.825$. From these values we can compute the values of c_O and c_I and thus the exact solution.

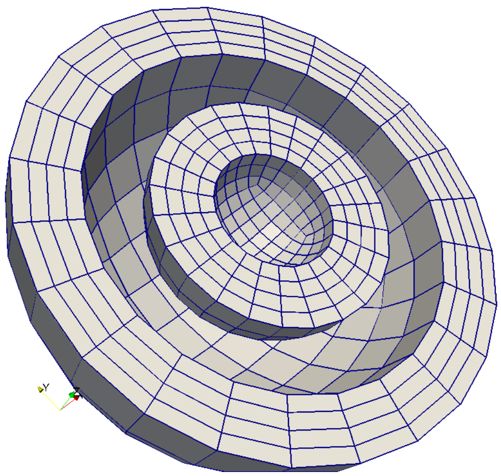
For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

These optimal rates are observed in this test.

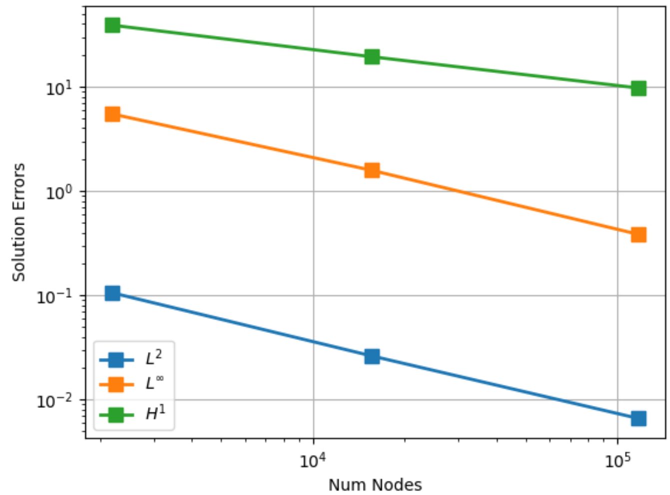
Table 7.3-3.. Enclosure Radiation: Convergence Rates

Num Dofs	L^2	L^∞	H^1
15588	2.14	1.90	1.06
117572	2.05	2.11	1.03

For input decks see Appendix 12.6.3.



Coarse Mesh



Error Norms

Figure 7.3-1.. Enclosure Radiation

7.4. 3D SPHERICAL SHELL PARTIAL ENCLOSURE

7.4.1. Problem Description

This problem tests coupled conduction and enclosure radiation with a partial enclosure. The geometry consists of two thick spherical shells separated by a gap. The outer shell has a section removed so that the enclosure is only partial.

7.4.2. Features Tested

Basic heat conduction, enclosure radiation with partial enclosure, Hex8 meshes.

7.4.3. Boundary Conditions

The outer and inner boundary conditions are prescribed at the nodes using the analytic solution. The analytic solution is used to set the boundary conditions on the cutaway face near the opening in the outer shell.

7.4.4. Material Parameters

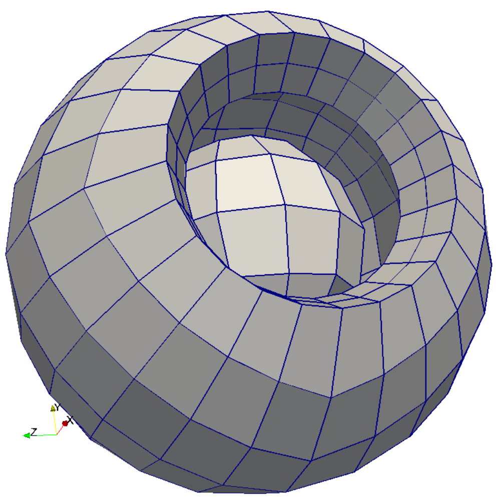
The values of density, thermal conductivity, and specific heat are all constant within each element block; however, the values differ between blocks.

7.4.5. Verification of Solution

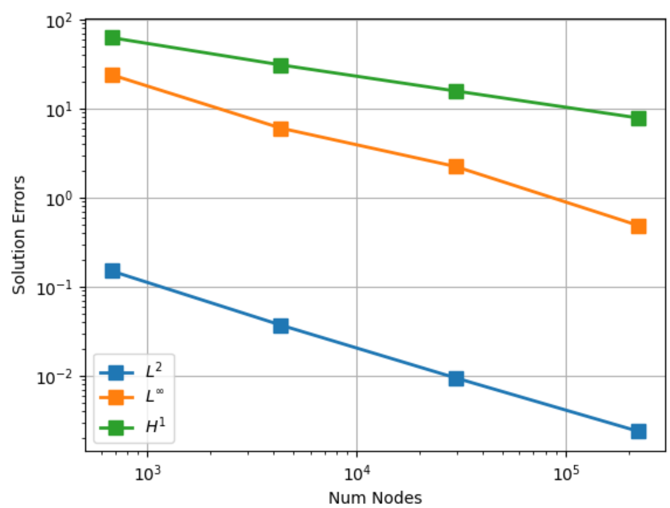
The analytic solution is identical to Section 7.3. The area for the partial enclosure is computed analytically.

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 2, 2, and 1, respectively (within a tolerance).

These optimal rates are observed in this test.



Coarse Mesh



Error Norms

Figure 7.4-1.. Partial Enclosure Radiation

Table 7.4-1.. Partial Enclosure Radiation: Convergence Rates

Num Dofs	L^2	L^∞	H^1
4338	2.26	2.22	1.13
29694	2.13	1.54	1.06
223162	2.06	2.27	1.03

For input decks see Appendix 12.6.4.

8. CHEMISTRY

8.1. FIRST ORDER REACTION (SPATIALLY VARYING TEMPERATURE)

This problem tests the interface to the CHEMEQ solver under the assumption that the temperature remains is variable in space but remains constant in time. The geometry consists of a unit cube meshed with Hex8 elements refined only in one direction (x).

8.1.1. Features Tested

CHEMEQ solver; source term from chemistry; nonlinear solver; second order time integrator with fixed time steps; constant initial temperature; constant temperature boundary condition.

8.1.2. Boundary Conditions

A constant temperature is applied on surface 1. The initial temperature is provided by an Encore user subroutine and the initial species values are $A = 1$ and $B = 0$.

8.1.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks. The CHEMEQ parameters are chosen to model a single first order reaction $A \rightarrow B$ with constant values of pre-exponential factor and activation energy.

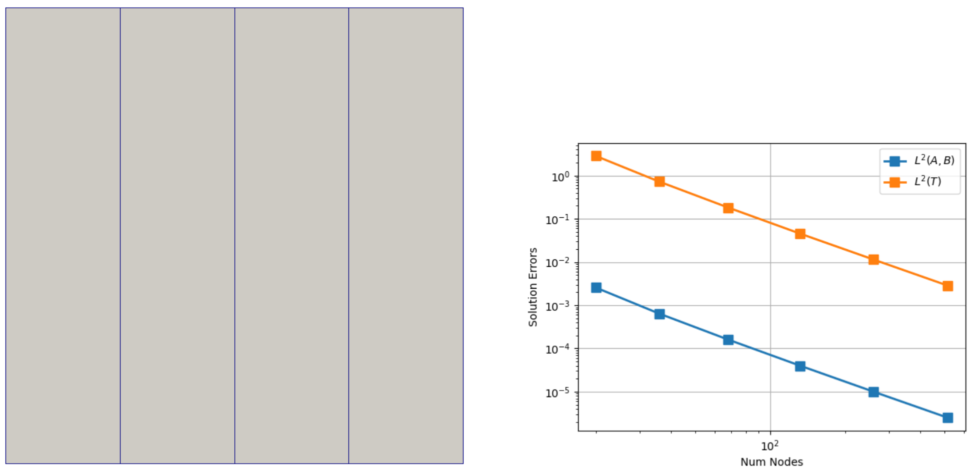
8.1.4. Verification of Solution

A manufactured solution is chosen as

$$
\begin{aligned}
T(x) &= 400\,(1 + 0.2\cos(\pi\,x)), \\
A(x,t) &= \exp\left\{ -\exp(5)\ \exp(-\frac{1000}{R\,T(x)})\,t \right\}, \\
B(x,t) &= 1 - A(x,t)
\end{aligned}
$$

where $R = 1.9872$ is the ideal gas constant. A source term is used to insure that the temperature does not vary in time.

For each mesh, the errors in the temperature and species A and B are computed in the L^2 norm. The test passes, only if the observed rates of convergence in these norms are 2 (within a tolerance).



Coarse Mesh　　Error Norms

Figure 8.1-1.. First Order Reaction (Spatially Varying Temperature)

Table 8.1-1.. First Order Reaction (Spatially Varying Temperature): Convergence Rates for Hex8 Meshes

Num Dofs	$L^2(A,B)$	$L^2(T)$
36	2.37	2.34
68	2.18	2.18
132	2.09	2.09
260	2.05	2.04
516	2.02	2.02

For input decks see Appendix 12.7.2.

8.2. FIRST ORDER REACTION

This problem tests the interface to the CHEMEQ solver under a temperature field that is variable in space and time. The geometry consists of a unit cube meshed with Hex8 elements refined only in one direction.

8.2.1. Features Tested

CHEMEQ solver; source term from chemistry; nonlinear solver; second order time integrator with fixed time steps; initial temperature from user sub; constant temperature boundary condition.

8.2.2. Boundary Conditions

The initial temperature and the temperature boundary condition on surface 1 are provided by an Encore user subroutine and the initial species values are $A = 1$ and $B = 0$.

8.2.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant, equal to one in both blocks. The CHEMEQ parameters are chosen to model a single first order reaction $A \to B$ with constant values of pre-exponential factor and activation energy.

8.2.4. Verification of Solution

A manufactured solution is chosen as

$$\begin{aligned}
\phi(x,t) &= \exp(a - E/(RT_0))(1 + 0.1\sin(x))\exp(t), \\
\Phi(x,t) &= \exp(a - E/(RT_0))(1 + 0.1\sin(x))(\exp(t) - 1), \\
T(x,t) &= (E/R)/(a - \ln(\phi(x,t))), \\
A(x,t) &= \exp(-\Phi(x,t)), \\
B(x,t) &= 1 - A(x,t)
\end{aligned}$$

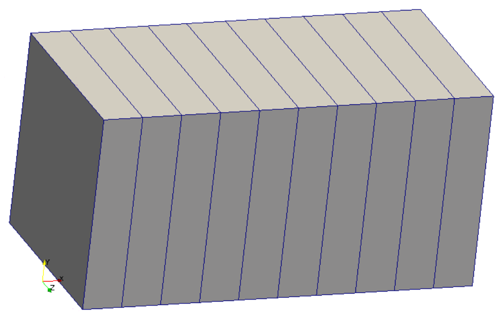
where a is the log pre-exponential factor, R is the ideal gas constant, E is the activation energy, and T_0 is a reference temperature value. The form of the solution is contrived so that

$$\begin{aligned}
\partial_t A(x,t) = -\partial_t B(x,t) &= -\phi(x,t)\, A(x,t) \\
\phi(x,t) &= \exp(a)\exp\left(-\frac{E}{RT(x,t)}\right)
\end{aligned}$$

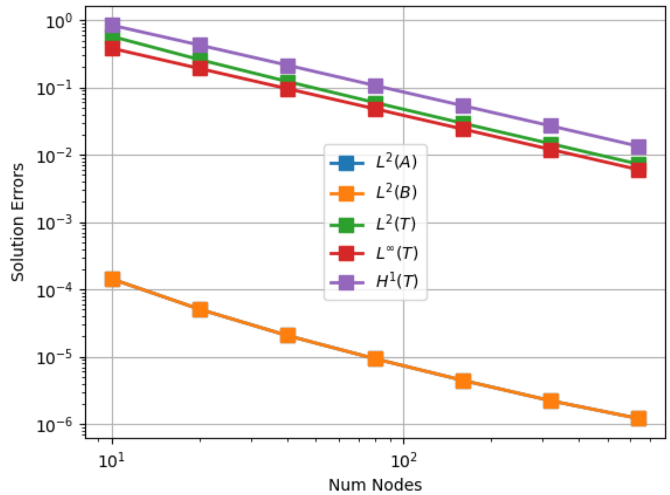
This allows the chemistry ODEs to be satisfied exactly, but a source term is needed in the energy equation.

For each mesh, the errors in the temperature and species A and B are computed in the L^2 norm. The test passes, only if the observed rates of convergence in these norms are 1 (within a tolerance). Currently it is not clear why the convergence rates are only first order.

For input decks see Appendix 12.7.3.



Coarse Mesh



Error Norms

Figure 8.2-1.. First Order Reaction

Table 8.2-1.. First Order Reaction: Convergence Rates for Hex8 Meshes

Num Dofs	$L^2(A)$	$L^2(B)$	$L^2(T)$	$L^\infty(T)$	$H^1(T)$
20	1.50	1.50	1.15	1.00	0.98
40	1.30	1.30	1.08	1.00	1.00
80	1.15	1.15	1.04	1.00	1.00
160	1.07	1.07	1.02	1.00	1.00
320	1.00	1.00	1.01	1.00	1.00
640	0.88	0.88	1.01	1.00	1.00

8.3. DAE AND PRESSURE TEST

This test runs CHEMEQ with a kinetics model that includes both pressure dependence and distributed activation energy for a single element mesh with uniform temperature and pressure.

8.3.1. Features Tested

Basic heat conduction on a Hex8 mesh; CHEMEQ solver with pressure dependence and distributed activation energy.

8.3.2. Boundary Conditions

No boundary conditions are prescribed, resulting in an adiabatic flux BC.

8.3.3. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant.

8.3.4. Verification of Solution

The analytic solution for the concentration of species A as a function of time for the constant values used in this test case is

$$A(t) = \frac{1}{2}\text{erfc}(\frac{1}{6}(\sqrt(2) - 6\text{erf}^{-1}(1 - 45t/(2\exp(\frac{61}{18})))))$$

The test compares the temperature errors against a gold file of the error at each time step. The exact solution for the concentration of A is also output to the exodus file and a comparison plotting that and the solved for concentration as a function of time has them lying on top of one another.

For input decks see Appendix 12.7.4.

This page intentionally left blank.

9. MISCELLANEOUS

9.1. THERMAL POSTPROCESSING

9.1.1. Problem Description

This problem tests basic thermal postprocessors in Aria.

9.1.2. Features Tested

Basic heat conduction, thermal postprocessors, Hex8 meshes.

9.1.3. Boundary Conditions

Dirichlet BCs are specified using the exact solution on surface 1. On surface 2, a natural convection BC is specified, using the exact solution as the reference temperature and a constant heat transfer coefficient. Similarly, a radiative flux BC is applied on surface 3, with constant values of emissivity and radiation form factor. A source term is applied within all blocks based on substituting the exact solution into the heat conduction operator.

9.1.4. Material Parameters

The values of density, thermal conductivity, and specific heat are all constant with the same value for both blocks.

9.1.5. Verification of Solution

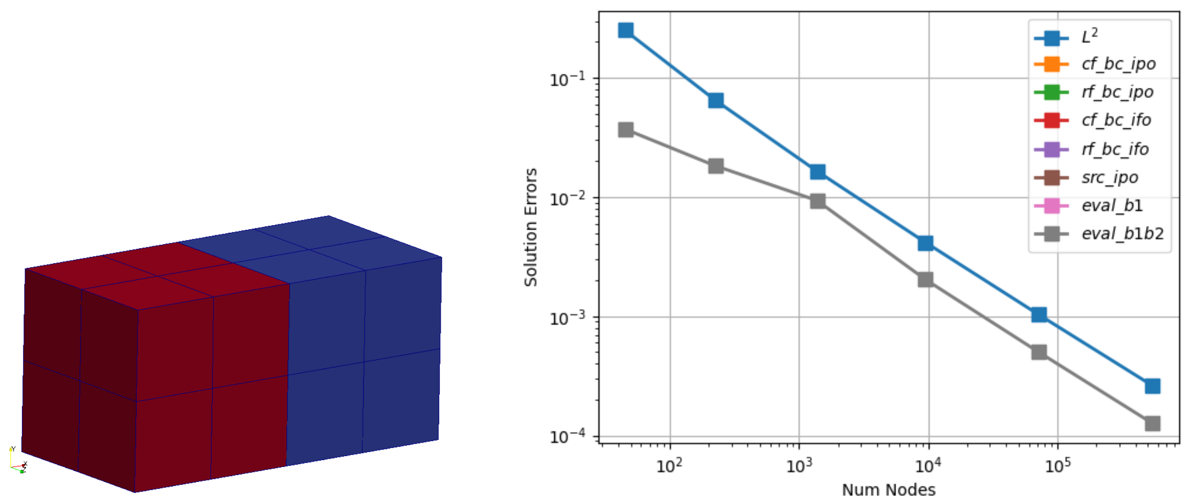
The manufactured solution is

$$T_0 + \exp(C_0(x^2 - 1) + C_1(y^2 - 0.25) + C_2(z^2 - 0.25) + C_3 t).$$

Postprocessors are computed for the integrated power output for convective and radiative BCs (cf_bc_ipo, rf_bc_ipo), the integrated flux output for convective and radiative BCs (cf_bc_ifo, rf_bc_ifo), the integrated power output for volume source terms (src_ipo), and several point evaluations (eval_b1, eval_b1b2, eval_s2).

For each mesh, the errors in the temperature solution are computed in the L^2 norm and for various postprocessors. The test passes, only if the observed rates of convergence are 2 (except for the integrated power output for source terms, which convergences with order 4).

These optimal rates are observed in this test clearly in most cases. However, for the point evaluation cases, a large amount of variability exists in the convergence rates.



Coarse Mesh   Error Norms

Figure 9.1-1.. Thermal Postprocess

Table 9.1-1.. Thermal Postprocess: Convergence Rates

Num Dofs	L^2	cf_bc_ipo	rf_bc_ipo	cf_bc_ifo	rf_bc_ifo	$eval\_b1$	$eval\_b1b2$	$eval\_s2$
225	2.52	2.87	2.82	2.87	2.82	1.16	-2.02	1.33
1377	2.27	2.52	2.48	2.52	2.48	1.77	1.66	1.11
9537	2.14	2.27	2.25	2.27	2.25	2.11	2.15	2.37
70785	2.07	2.13	2.12	2.13	2.12	2.06	2.07	2.09
545025	2.04	2.06	2.05	2.06	2.05	2.03	2.04	2.04

For input decks see Appendix 12.8.1.

9.2. LOCAL COORDINATES: CARTESIAN

This problem tests the use of a local Cartesian coordinate system in a material model. The geometry is a 3D cube that has been rotated.

9.2.1. Features Tested

Steady heat conduction, time integrators, tensor thermal conductivity, local Cartesian coordinates in a material model.

9.2.2. Boundary Conditions

The boundary conditions are prescribed at the nodes using the analytic solution. The initial condition is specified using an Encore function evaluated at the nodes.

9.2.3. Material Parameters

The specific heat is constant. The density and thermal conductivity are constant, with a diagonal (tensor) thermal conductivity in the local coordinate space of the material.

9.2.4. Verification of Solution

A manufactured solution is chosen as

$$T(X,Y,Z) = T_0 + T_1 \cos(x_k X)\cos(y_k Y)\cos(z_k Z)$$

where (X, Y, Z) are the local material coordinates, which are related to the Cartesian coordinates (x, y, z) by a rotation matrix consisting of a product of rotations (22.5 deg around the z-axis and 45 deg around the x-axis).

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 1 for H^1 and 2 for all other norms (within a tolerance).

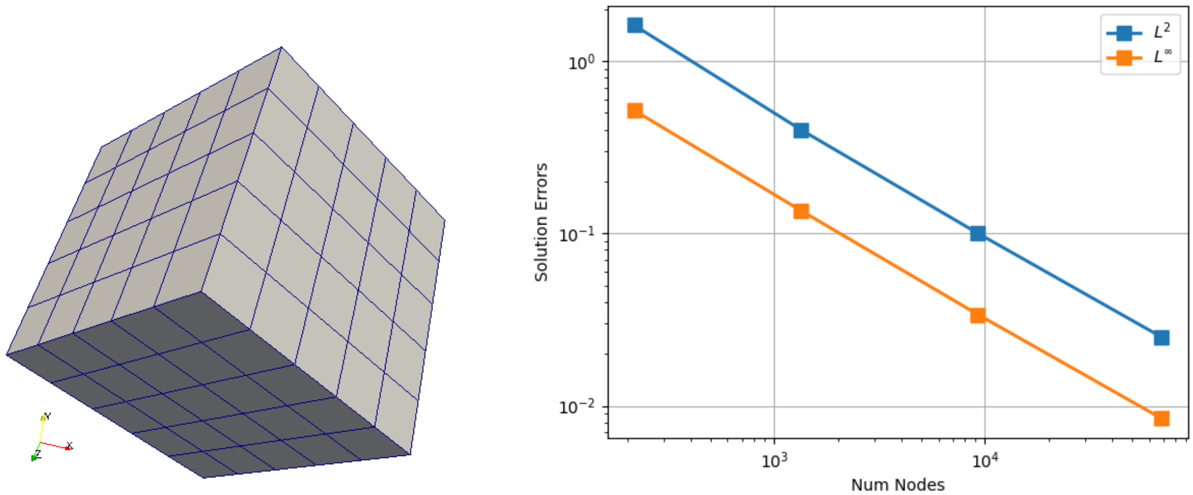
Table 9.2-1.. Local Cartesian Coordinate System: Convergence Rates

Num Dofs	L^2	L^∞
1331	2.29	2.20
9261	2.15	2.15
68921	2.07	2.07

For input decks see Appendix 12.8.3.

9.3. LOCAL COORDINATES: CYLINDRICAL

This problem tests the use of a local cylindrical coordinate system in a material model. The geometry is a 3D cube that has been rotated.



Coarse Mesh Error Norms

Figure 9.2-1.. Local Cartesian Coordinate System

9.3.1. Features Tested

Steady heat conduction, time integrators, tensor thermal conductivity, local coordinates in a material model.

9.3.2. Boundary Conditions

The boundary conditions are prescribed at the nodes using the analytic solution. The initial condition is specified using an Encore function evaluated at the nodes.

9.3.3. Material Parameters

The specific heat and density are constant. The diagonal (tensor) components of the thermal conductivity are specified using constant values in the local coordinate space of the material.

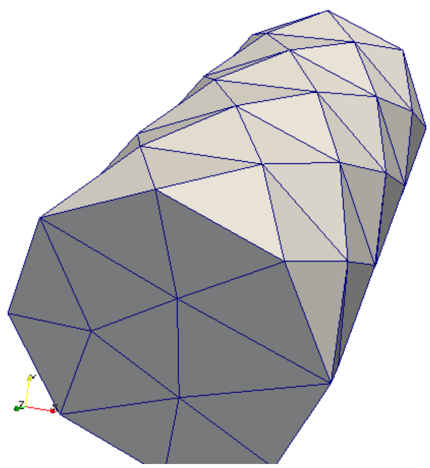
9.3.4. Verification of Solution

A manufactured solution is chosen as

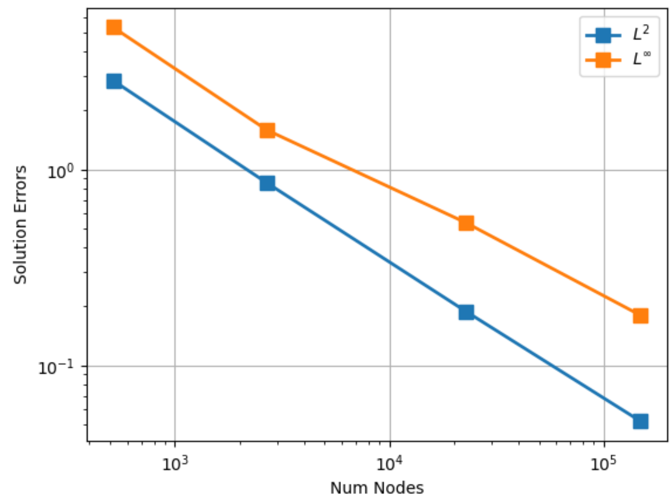
$$T(X, Y, Z) = T_0 + T_1(2R)^3 \cos(\theta) \cos(z_k Z)$$

where (R, Θ, Z) are the local cylindrical material coordinates, which are related to the standard cylindrical coordinates (x, y, z) by a rotation matrix consisting of a product of rotations (22.5 deg around the z-axis and 45 deg around the x-axis).

For each mesh, the errors in the temperature solution are computed in the L^2, L^∞ and H^1 norms. The test passes, only if the observed rates of convergence in these norms are 1 for H^1 and 2 for all other norms (within a tolerance).



Coarse Mesh



Error Norms

Figure 9.3-1.. Local Cylindrical Coordinate System

Table 9.3-1.. Local Cylindrical Coordinate System: Convergence Rates

Num Dofs	L^2	L^∞
2692	2.19	2.19
22723	2.12	1.53
148839	2.07	1.74

For input decks see Appendix 12.8.4.

This page intentionally left blank.

10. LOW-MACH FLUID FLOW

Documentation for the following tests is in progress:

```
_rtest/aria/cvfemConvTaylorVortex/cvfemConvTaylorVortex.test|np4
_rtest/aria/gfemConvTaylorVortex/gfemConvTaylorVortex.test|np4
_rtest/aria/hfemConvTaylorVortex/hfemConvTaylorVortex.test|np4
mConvTaylorVortex/cvfemConvTaylorVortex.test|np8
mSteadyTaylorVortex/cvfemSteadyTaylorVortex.test|np8
mSteadyTaylorVortexKeps/cvfemSteadyTaylorVortexKeps.test|np8
m_couette_flow/cdfem_couette_flow.test|cdfem_couette_flow_tri3
m_couette_flow/cdfem_couette_flow.test|cdfem_couette_flow_tri6
ConvTaylorVortex/gfemConvTaylorVortex.test|np8
SteadyTaylorVortex/gfemSteadyTaylorVortex.test|np8
```

This page intentionally left blank.

11. HOW TO BUILD THIS DOCUMENT

You need to have Sierra developer access (through WebCars). Then you should clone the Sierra Git repository containing the tests to a location with adequate memory (currently more than 80GB), using a command like this:

```
git clone sierra-git:/git/tests
```

Then you need to assign the verification tests, running the following command from your local tests repository:

```
assign --path aria_rtest/verification
```

This will produce a text file called assigned.tests containing the list of all tests to run. You should edit the second line of this file to indicate the remote location (accessible from the HPC machine where you will run the tests). For example, I might have something like this:

```
# Created by assign at Fri Sep 19 09:52:09 2014
#@ /gscratch1/bcarnes/TESTS
aria_rtest/verification/1dnonlin_verify1/1dnonlin_verify1.test|np8
aria_rtest/verification/cyl_shell_2d/cyl_shell_2d.test|np8
aria_rtest/verification/cyl_shell_3d/cyl_shell_3d.test|np8
...
```

Next you need to copy the test files and the assigned.test file to the remote location (here it is "/gscratch1/bcarnes/TESTS/"):

```
rsync -azv aria_rtest/verification redsky:/gscratch1/bcarnes/TESTS/aria_rtest
scp assigned.tests redsky:/gscratch1/bcarnes/TESTS/
```

Here I am only copying the verification test sub-directory, since I do not want to run any other tests.

On the HPC machine, you will need to load a pre-built version of the code such as the nightly master build:

```
module load sierra/master
```

To see where the executables are located, you can run something like:

```
[bcarnes@redsky-login9 ~]$ which aria
/projects/sierra/redsky/install/master/bin/aria
```

Finally, to run the tests, you use the testrun script, with a few additional arguments. The first locates the source code needed to compile the various user subroutines (which we just found from running "which aria"), the second enables tests to run as long as needed, the third uses the queue, and the fourth saves the results so you can use them in the manual.

```
testrun --user sourcedir=/projects/sierra/tlcc2/install/master/ \
        --allow-multipliers=time \
        --queued \
        --save-all-results
```

It may take 1-2 hours to run all the tests. Note that if the tests start to fail with an error associated with the ACCOUNT not being set, you may need to set it using your WCID:

```
export ACCOUNT=fyXXXXX
```

To view your available WCIDs, run the following command:

```
mywcid
```

To build this manual, you should clone the Sierra Git repository containing the documentation files using a command like this:

```
git clone sierra-git:/git/docs
```

Then go to the directory within your local repository containing the Aria Verification Manual files:

```
cd aria/doc/verification_manual
```

Once the tests have all ran successfully, you should sync the results from the remote location back to this directory:

```
rsync -azv redsky:/gscratch1/bcarnes/TESTS/results .
```

Then run the a script to execute any local postprocessing needed to create the plots for the tests:

```
python ariaPostprocess.py
```

Finally you can create the manual using pdflatex:

```
pdflatex Aria_Verification_Manual.tex
```

which should create a new PDF output file.

12. INPUT DECKS FOR VERIFICATION PROBLEMS

12.1. BASIC THERMAL TESTS

12.1.1. Steady Heat Conduction: Hex8 Meshes

12.1.2. Steady Heat Conduction: Hex20 Meshes

12.1.3. Steady Heat Conduction: Hex27 Meshes

12.1.4. Steady Heat Conduction: Tet4 Meshes

12.1.5. Steady Heat Conduction: Tet4Tet10 Meshes

```
BEGIN SIERRA myJob

  Begin universal aria expressions
    User Expression = Scalar_String_Function \$
      f = "(x-x^2)^2*(y-y^2)^2*(z-z^2)^2 + 1" \$
      user_tag = T_mms

    User Expression = Vector_String_Function \$
      f_x = "2*(x-x^2)*(1-2*x)*(y-y^2)^2*(z-z^2)^2" \$
      f_y = "2*(y-y^2)*(1-2*y)*(x-x^2)^2*(z-z^2)^2" \$
      f_z = "2*(z-z^2)*(1-2*z)*(x-x^2)^2*(y-y^2)^2" \$
      user_tag = gradT

    User Expression = Scalar_String_Function \$
      f = "2*((1-2*x)^2 - 2*(x-x^2))*(y-y^2)^2*(z-z^2)^2  \$
         + 2*((1-2*y)^2 - 2*(y-y^2))*(x-x^2)^2*(z-z^2)^2  \$
         + 2*((1-2*z)^2 - 2*(z-z^2))*(x-x^2)^2*(y-y^2)^2" \$
      user_tag = laplaceT

    User Expression = Scalar_String_Function \$
      f = "-laplaceT - 1.0" \$
      user_tag = mms_src
  End

   BEGIN ARIA MATERIAL Kryptonite
     Density              = Constant rho=1
     Thermal Conductivity = constant k=1
     Specific Heat        = Constant cp=1
     heat conduction      = basic
   END ARIA MATERIAL Kryptonite
```

```
BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN CG SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = NONE
    CONVERGENCE TOLERANCE = 1.000000e-14
  END
END TPETRA EQUATION SOLVER


 BEGIN FINITE ELEMENT MODEL cube
    database name = cube_h{N}_tet10.e
    coordinate system is cartesian
    decomposition method = rcb

    BEGIN PARAMETERS FOR BLOCK block_1
       material Kryptonite
    END PARAMETERS FOR BLOCK block_1

 END FINITE ELEMENT MODEL cube

 BEGIN PROCEDURE myAriaProcedure

    Begin Solution Control Description
       Use System Main
       Begin System Main
          Begin Sequential The_Time_Block
             Advance myRegion
          End
            Simulation Start Time = 0
          Simulation Termination Time = 1
       End
    End

    BEGIN ARIA REGION myRegion

       Nonlinear Solution Strategy = Newton
       Maximum Nonlinear Iterations = 10
       Nonlinear Residual Tolerance = 1.0e-12
       Nonlinear Correction Tolerance = 1.0e-12
       Nonlinear Relaxation Factor  = 1.0
       use dof averaged nonlinear residual

       use finite element model cube
       Use Linear Solver Iterative_Solver #Direct_Solver

       EQ ENERGY for TEMPERATURE on block_1 using Q1 with DIFF SRC

       # surface_4: x=0
       # surface_6: x=1

       # surface_3: y=0
       # surface_5: y=1

       # surface_1: z=1
       # surface_2: z=0

       # const Temp BC (x)
       BC const dirichlet at surface_4 Temperature = 1.0
       BC const dirichlet at surface_6 Temperature = 1.0

       # const flux BC (y)
       BC FLUX for Energy on surface_3  = constant flux = 3
       BC FLUX for Energy on surface_5  = constant flux = 5

       BC Flux for Energy on surface_3 = Scalar_String_Function f = "-gradT_y - 3.0"
       BC Flux for Energy on surface_5 = Scalar_String_Function f = "gradT_y - 5.0"
```

```
         # convective flux BC with const Temp and H (z)
         BC Flux for Energy on surface_1 = Nat_Conv T_Ref=1 H=1
         BC Flux for Energy on surface_2 = Nat_Conv T_Ref=2 H=2

         BC Flux for Energy on surface_1 = Scalar_String_Function f = "gradT_z - (T_mms-1.0)"
         BC Flux for Energy on surface_2 = Scalar_String_Function f = "-gradT_z - 2.0*(T_mms-2.0)"

         #  const source term
         Source For ENERGY on block_1 = Constant value=1
         Source for ENERGY on block_1 = Scalar_String_Function f = "mms_src"

        postprocess L_2_norm of function "temperature-T_mms" on all_blocks as l2
        postprocess L_inf_norm of function "temperature-T_mms" on all_blocks as linf
        postprocess L_2_norm of function "sqrt((grad_temperature_x-gradT_x)^2+(grad_temperature_y-gradT_y)^2+(grad_temperature_z

        Begin Heartbeat hb
          At Step 0 Interval is 1
          Precision = 6
          Stream Name = errors_h{N}.dat
          legend = off
          labels = off
          Timestamp Format    = ""
          variable = global time
          variable = global number_of_nodes
          variable = global l2
          variable = global h1
          variable = global linf
        End

         BEGIN RESULTS OUTPUT LABEL diffusion output
            database Name = thermal_steady_tet4_h{N}.e
            at step 1, increment = 1
            title Aria cube test
            nodal variables = nonlinear_solution->TEMPERATURE as T
         END RESULTS OUTPUT LABEL diffusion output

      END ARIA REGION myRegion

   END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.1.6. Steady Heat Conduction: Tet10 Meshes

12.1.7. Transient Heat Conduction: Hex8 Meshes

```
# T_ref_s1 = { T_ref_s1 = 1 }
# h_s1 = { h_s1 = 1 }
# T_ref_s2 = { T_ref_s2 = 2 }
# h_s2 = { h_s2 = 2 }
# C_s = { C_s = 10000.0 }

BEGIN SIERRA myJob
  #{include(exact_transient.inc)}


  BEGIN ARIA MATERIAL Kryptonite
    Density              = Constant rho={rho}
    Thermal Conductivity = constant k={k}
    Specific Heat        = Constant cp={Cp}
    heat conduction      = basic
    latent heat = constant value={L}
```

```
  user expression = scalar_string_function user_tag = "m_s" \$
    f = "{C_s}*(exp(-t) - exp(-(t - 1)*(t - 1))*(t*(2*t - 2) - 1))"

  user expression = scalar_string_function user_tag = "m_s_dot" \$
    f = "-{C_s}*(exp(-t) + exp(-(t - 1)*(t - 1))*(4*t - 2) - exp(-(t - 1)*(t - 1))*(2*t - 2)*(t*(2*t - 2) - 1))"
END ARIA MATERIAL Kryptonite

BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
  BEGIN SUPERLU SOLVER
  END
END TPETRA EQUATION SOLVER


BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN CG SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = NONE
    CONVERGENCE TOLERANCE = 1.000000e-12
  END
END TPETRA EQUATION SOLVER


BEGIN FINITE ELEMENT MODEL cube
  database name = cube_h{N}_hex8.e
  coordinate system is cartesian
  decomposition method = rcb

  BEGIN PARAMETERS FOR BLOCK block_1
    material Kryptonite
  END PARAMETERS FOR BLOCK block_1

END FINITE ELEMENT MODEL cube

BEGIN PROCEDURE myAriaProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Begin Transient The_Time_Block
        Advance myRegion
      End
      Simulation Start Time = 0
      Simulation Termination Time = 3
    End
    Begin Parameters For Transient The_Time_Block
      Begin Parameters For Aria Region myRegion
        Initial Time Step Size = {0.5/2**N}
        Time Integration Method = Second_Order
        Time Step Variation = fixed
      End
    End
  End

  BEGIN ARIA REGION myRegion

    Nonlinear Solution Strategy = Newton
    Maximum Nonlinear Iterations = 10
    Nonlinear Residual Tolerance = 1.0e-12
    Nonlinear Correction Tolerance = 1.0e-12
    Nonlinear Relaxation Factor  = 1.0
    use dof averaged nonlinear residual

    use finite element model cube
    Use Linear Solver Iterative_Solver #Direct_Solver

    EQ ENERGY for TEMPERATURE on block_1 using Q1 with MASS DIFF SRC
```

```
IC for temperature on block_1 = scalar_string_function f="exact_sln"

# surface_4: x=0
# surface_6: x=1

# surface_3: y=0
# surface_5: y=1

# surface_1: z=1
# surface_2: z=0

# const Temp BC (x)
BC const dirichlet at surface_4 Temperature = 1.0
BC const dirichlet at surface_6 Temperature = 1.0

# const flux BC (y)
BC FLUX for Energy on surface_3  = constant flux = 3
BC FLUX for Energy on surface_3  = scalar_string_function f="exact_flux_y - 3"

Begin Heat Flux Boundary Condition hfbc2
  Add Surface surface_5
  Flux = -5
End
BC FLUX for Energy on surface_5  = scalar_string_function f="-exact_flux_y - 5"

# convective flux BC with const Temp and H (z)
BC Flux for Energy on surface_1 = Nat_Conv T_Ref={T_ref_s1} H={h_s1}
BC FLUX for Energy on surface_1  = scalar_string_Function \$
  f="-exact_flux_z - {h_s1}*(exact_sln - {T_ref_s1})"

Begin Convective Flux Boundary Condition cfbc2
  Add Surface surface_2
  Convective Coefficient = {h_s2}
  Reference Temperature = {T_ref_s2}
End
BC FLUX for Energy on surface_2  = scalar_string_Function \$
  f="exact_flux_z - {h_s2}*(exact_sln - {T_ref_s2})"

#  const source term
Begin Volume Heating vh1
  Add Volume block_1
  Value = 1
End
Source for Energy on block_1 = melting Ts={Ts} Tl={Tl}
Source For ENERGY on block_1 = scalar_string_Function f="exact_src - melting_src - 1"

postprocess max of function "abs(exact_sln - temperature)" on all_blocks as linf_err
postprocess l_2_norm of function "exact_sln - temperature" on all_blocks as l2_err
postprocess l_2_norm of function "exact_sln_dot - dt_temperature" on all_blocks as l2_dot_err
postprocess integral of function "\$
  (gradExact_sln_x - grad_temperature_x)^2 + \$
  (gradExact_sln_y - grad_temperature_y)^2 + \$
  (gradExact_sln_z - grad_temperature_z)^2" on all_blocks as H1_err_sq
postprocess global_function "sqrt(H1_err_sq)" as H1_err

Begin heartbeat testctrl
  Stream name = errors{N}.dat
  Timestamp Format    = ""
  Precision           = 8
  Format = csv
  Labels = Off
  Legend = On
  At step 0, Increment is 1
  variable = global time
  variable = global number_of_nodes
  variable = global l2_err
  variable = global l2_dot_err
```

```
        variable = global H1_err
        variable = global linf_err
      end

      BEGIN RESULTS OUTPUT LABEL diffusion output
        database Name = output{N}.e
        at step 0, increment = {2**N}
        #at step 0, increment = 1
        title Aria cube test
        nodal variables = solution->TEMPERATURE as T
        nodal variables = time_derivative_at_time->TEMPERATURE as TDOT
      END RESULTS OUTPUT LABEL diffusion output

    END ARIA REGION myRegion

  END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.1.8. Transient Heat Conduction: Tet4 Meshes

```
# T_ref_s1 = { T_ref_s1 = 1 }
# h_s1 = { h_s1 = 1 }
# h_s2 = { h_s2 = 2 }
# T0 = { T0 = 2 }
# C_s = { C_s = 10000.0 }
# omega = { omega = PI }
# bn_vol = { bn_vol = 0.5 }
# L = { L = 10 }
# rho = { rho = 1 }
# Cp = { Cp = 1 }
# k = { k = 1 }
# Ts = { Ts = 0.5 }
# Tl = { Tl = 1.5 }
# Tm = { Tm = 0.5 * (Ts + Tl) }
# sigma = { sigma = 0.429858 * (Tm - Ts) }
# invsigma = { invsigma = 1. / (sigma * sqrt(2.)) }

BEGIN SIERRA myJob
  begin universal aria expressions
    user expression = scalar_string_function user_tag = "m_s" \$
      f = "{C_s}*( 1. - exp(-t) + t*exp( -(t-1.0)*(t-1.0) ) )"

    user expression = scalar_string_function user_tag = "m_s_dot" \$
      f = "{C_s}*( exp(-t) + (1.0 + t*( -2.0*(t-1.0)) )*exp( -(t-1.0)*(t-1.0)) )"

    user expression = scalar_string_function user_tag = "bulk_node_exact_solution" \$
      f = "{T0} * (sin({omega} * t) + 1)"

    user expression = scalar_string_function user_tag = "bulk_node_source" \$
      f = "{rho * Cp * omega * T0} * cos({omega} * t) - ({h_s2} * (1 - bulk_node_exact_solution))/{bn_vol}"

    user expression = scalar_string_function user_tag = "exact_spatial_var" \$
      f = "(x-x*x)^2*(y-y*y)^2*(z-z*z)^2"

    user expression = scalar_string_function user_tag = "exact_sln" \$
      f = "exact_spatial_var*m_s + 1.0"

    user expression = scalar_string_function user_tag = "exact_sln_dot" \$
      f = "exact_spatial_var*m_s_dot"

    user expression = vector_string_function user_tag = "gradExact_sln" \$
      f_x = "2.0*m_s*(x-x*x)*(1-2*x)*(y-y*y)^2*(z-z*z)^2" \$
      f_y = "2.0*m_s*(y-y*y)*(1-2*y)*(x-x*x)^2*(z-z*z)^2" \$
      f_z = "2.0*m_s*(z-z*z)*(1-2*z)*(x-x*x)^2*(y-y*y)^2"
```

```
  user expression = vector_string_function user_tag = "exact_flux" \$
    f_x = "{-k}*gradExact_sln_x" \$
    f_y = "{-k}*gradExact_sln_y" \$
    f_z = "{-k}*gradExact_sln_z"

  user expression = scalar_string_function user_tag = "exact_sln_laplace" \$
    f = "2.0*m_s*( ( (1-2*x)*(1-2*x) - 2*(x-x*x) )*(y-y*y)^2*(z-z*z)^2 \$
                 + ( (1-2*y)*(1-2*y) - 2*(y-y*y) )*(x-x*x)^2*(z-z*z)^2 \$
                 + ( (1-2*z)*(1-2*z) - 2*(z-z*z) )*(x-x*x)^2*(y-y*y)^2 )"

  user expression = scalar_string_function user_tag = "exact_src" \$
    f = "{rho}*{Cp}*exact_sln_dot - {k}*exact_sln_laplace"
end universal aria expressions

BEGIN ARIA MATERIAL Kryptonite
  Density              = Constant rho={rho}
  Thermal Conductivity = constant k={k}
  Specific Heat        = Constant cp={Cp}
  heat conduction      = basic
END ARIA MATERIAL Kryptonite

BEGIN ARIA MATERIAL Air
  Density              = Constant rho={rho}
  Thermal Conductivity = constant k={k}
  Specific Heat        = Constant cp={Cp}
  heat conduction      = basic
END ARIA MATERIAL

BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
  BEGIN SUPERLU SOLVER
  END
END TPETRA EQUATION SOLVER


BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN CG SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = NONE
    CONVERGENCE TOLERANCE = 1.000000e-12
  END
END TPETRA EQUATION SOLVER


BEGIN FINITE ELEMENT MODEL cube
  database name = cube_h{N}_tet4.e
  coordinate system is cartesian
  decomposition method = rcb

  BEGIN PARAMETERS FOR BLOCK block_1
    material Kryptonite
  END PARAMETERS FOR BLOCK block_1

END FINITE ELEMENT MODEL cube

BEGIN PROCEDURE myAriaProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Begin Transient The_Time_Block
        Advance myRegion
      End
      Simulation Start Time = 0
      Simulation Termination Time = 3
    End
    Begin Parameters For Transient The_Time_Block
```

```
    Begin Parameters For Aria Region myRegion
      Initial Time Step Size = {0.5/2**N}
      Time Integration Method = Second_Order
      Time Step Variation = fixed
    End
  End
End

BEGIN ARIA REGION myRegion

  Nonlinear Solution Strategy = Newton
  Minimum Nonlinear Solves = 1
  Maximum Nonlinear Iterations = 10
  Nonlinear Residual Tolerance = 1.0e-12
  nonlinear residual minimum convergence rate = 0.999 number of steps = 3
  Nonlinear Correction Tolerance = 1.0e-12
  Nonlinear Relaxation Factor  = 1.0
  use dof averaged nonlinear residual

  use finite element model cube
  Use Linear Solver Iterative_Solver #Direct_Solver

  EQ ENERGY for TEMPERATURE on block_1 using Q1 with MASS DIFF SRC

  IC const on block_1 temperature = 1.0

  # surface_4: x=0
  # surface_6: x=1

  # surface_3: y=0
  # surface_5: y=1

  # surface_1: z=1
  # surface_2: z=0

  # const Temp BC (x)
  BC const dirichlet at surface_4 Temperature = 1.0
  BC const dirichlet at surface_6 Temperature = 1.0

  # const flux BC (y)
  BC FLUX for Energy on surface_3  = constant flux = 3
  BC FLUX for Energy on surface_3  = scalar_string_function f="exact_flux_y - 3"

  BC FLUX for Energy on surface_5  = constant flux = 5
  BC FLUX for Energy on surface_5  = scalar_string_function f="-exact_flux_y - 5"

  # convective flux BC with const Temp and H (z)
  BC Flux for Energy on surface_1 = Nat_Conv T_Ref={T_ref_s1} H={h_s1}
  BC FLUX for Energy on surface_1  = scalar_string_Function \$
    f="-exact_flux_z - {h_s1}*(exact_sln - {T_ref_s1})"

  Begin Bulk Fluid Element aBulkNode
    material = Air
    bulk element volume = constant v = {bn_vol}
    initial temperature = {T0}
    bulk eq energy for temperature using p0 with mass src
    bulk source for energy = scalar_string_function f="bulk_node_source"
  End
  Begin Convective Flux Boundary Condition bulk_flux
    add surface surface_2
    use bulk element aBulkNode
    convective coefficient = {h_s2}
  End
  BC FLUX for Energy on surface_2  = scalar_string_Function \$
    f="exact_flux_z - {h_s2}*(exact_sln - bulk_node_exact_solution)"

  #  const source term
  Source For ENERGY on block_1 = Constant value=1
```

```
      Source For ENERGY on block_1 = scalar_string_Function f="exact_src-1"

      postprocess l_inf_norm of function "exact_sln - temperature" on block_1 as linf_err
      postprocess l_inf_norm of function "bulk_node_exact_solution - temperature" on block_for_abulknode as linf_bulk_node
      postprocess l_2_norm of function "exact_sln - temperature" on block_1 as l2_err
      postprocess l_2_norm of function "exact_sln_dot - dt_temperature" on block_1 as l2_dot_err
      postprocess integral of function "\$
        (gradExact_sln_x - grad_temperature_x)^2 + \$
        (gradExact_sln_y - grad_temperature_y)^2 + \$
        (gradExact_sln_z - grad_temperature_z)^2" on block_1 as H1_err_sq
      postprocess global_function "sqrt(H1_err_sq)" as H1_err

      Begin heartbeat testctrl
        Stream name = errors_thermal_transient_hex8_tied_contact_h{N}.dat
        Timestamp Format    = ""
        Precision           = 8
        Format = csv
        Labels = Off
        Legend = On
        At step 0, Increment is 1
        variable = global time
        variable = global number_of_nodes
        variable = global l2_err
        variable = global l2_dot_err
        variable = global H1_err
        variable = global linf_err
        variable = global linf_bulk_node
      end

      BEGIN RESULTS OUTPUT LABEL diffusion output
        database Name = errors{N}.e
        at step 0, increment = {2**N}
        title Aria cube test
        nodal variables = solution->TEMPERATURE as T
        nodal variables = time_derivative_at_time->TEMPERATURE as TDOT
        global variables = abulknode_T


        nodal variables = err_exact_soln
        nodal variables = err_exact_soln_dot
        nodal variables = err_exact_src
        nodal variables = err_exact_flux
      END RESULTS OUTPUT LABEL diffusion output

    END ARIA REGION myRegion

  END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.1.9. Transient Heat Conduction: Tet4Tet10 Meshes

```
# T_ref_s1 = { T_ref_s1 = 1 }
# h_s1 = { h_s1 = 1 }
# T_ref_s2 = { T_ref_s2 = 2 }
# h_s2 = { h_s2 = 2 }
# C_s = { C_s = 10000.0 }

BEGIN SIERRA myJob
  #{include(exact_transient.inc)}


  # load user plugin file ./exact_transient.so

  BEGIN ARIA MATERIAL Kryptonite
    Density              = Constant rho={rho}
```

```
  Thermal Conductivity = constant k={k}
  Specific Heat        = Constant cp={Cp}
  heat conduction      = basic

  user expression = scalar_string_function user_tag = "m_s" \$
    f = "{C_s}*( 1. - exp(-t) + t*exp( -(t-1.0)*(t-1.0) ) )"

  user expression = scalar_string_function user_tag = "m_s_dot" \$
    f = "{C_s}*( exp(-t) + (1.0 + t*( -2.0*(t-1.0)) )*exp( -(t-1.0)*(t-1.0)) )"
END ARIA MATERIAL Kryptonite

BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
  BEGIN SUPERLU SOLVER
  END
END TPETRA EQUATION SOLVER


BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN CG SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = NONE
    CONVERGENCE TOLERANCE = 1.000000e-12
  END
END TPETRA EQUATION SOLVER


BEGIN FINITE ELEMENT MODEL cube
  database name = cube_h{N}_tet10.e
  coordinate system is cartesian
  decomposition method = rcb

  BEGIN PARAMETERS FOR BLOCK block_1
    material Kryptonite
  END PARAMETERS FOR BLOCK block_1

END FINITE ELEMENT MODEL cube

BEGIN PROCEDURE myAriaProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Begin Transient The_Time_Block
        Advance myRegion
      End
      Simulation Start Time = 0
      Simulation Termination Time = 3
    End
    Begin Parameters For Transient The_Time_Block
      Begin Parameters For Aria Region myRegion
        Initial Time Step Size = {0.5/2**N}
        Time Integration Method = Second_Order
        Time Step Variation = fixed
      End
    End
  End

  BEGIN ARIA REGION myRegion

    Nonlinear Solution Strategy = Newton
    Maximum Nonlinear Iterations = 10
    Nonlinear Residual Tolerance = 1.0e-12
    Nonlinear Correction Tolerance = 1.0e-12
    Nonlinear Relaxation Factor  = 1.0
    use dof averaged nonlinear residual
```

```
use finite element model cube
Use Linear Solver Iterative_Solver #Direct_Solver

EQ ENERGY for TEMPERATURE on block_1 using Q1 with MASS DIFF SRC

IC const on block_1 temperature = 1.0

# surface_4: x=0
# surface_6: x=1

# surface_3: y=0
# surface_5: y=1

# surface_1: z=1
# surface_2: z=0

# const Temp BC (x)
BC const dirichlet at surface_4 Temperature = 1.0
BC const dirichlet at surface_6 Temperature = 1.0

# const flux BC (y)
BC FLUX for Energy on surface_3  = constant flux = 3
BC FLUX for Energy on surface_3  = scalar_string_function f="exact_flux_y - 3"

BC FLUX for Energy on surface_5  = constant flux = 5
BC FLUX for Energy on surface_5  = scalar_string_function f="-exact_flux_y - 5"

# convective flux BC with const Temp and H (z)
BC Flux for Energy on surface_1 = Nat_Conv T_Ref={T_ref_s1} H={h_s1}
BC FLUX for Energy on surface_1  = scalar_string_Function \$
  f="-exact_flux_z - {h_s1}*(exact_sln - {T_ref_s1})"

BC Flux for Energy on surface_2 = Nat_Conv T_Ref={T_ref_s2} H={h_s2}
BC FLUX for Energy on surface_2  = scalar_string_Function \$
  f="exact_flux_z - {h_s2}*(exact_sln - {T_ref_s2})"

#  const source term
Source For ENERGY on block_1 = Constant value=1
Source For ENERGY on block_1 = scalar_string_Function f="exact_src-1"

postprocess max of function "abs(exact_sln - temperature)" on all_blocks as linf_err
postprocess l_2_norm of function "exact_sln - temperature" on all_blocks as l2_err
postprocess l_2_norm of function "exact_sln_dot - dt_temperature" on all_blocks as l2_dot_err
postprocess integral of function "\$
  (gradExact_sln_x - grad_temperature_x)^2 + \$
  (gradExact_sln_y - grad_temperature_y)^2 + \$
  (gradExact_sln_z - grad_temperature_z)^2" on all_blocks as H1_err_sq
postprocess global_function "sqrt(H1_err_sq)" as H1_err

Begin heartbeat testctrl
  Stream name = errors{N}.dat
  Timestamp Format    = ""
  Precision           = 8
  Format = csv
  Labels = Off
  Legend = On
  At step 0, Increment is 1
  variable = global time
  variable = global number_of_nodes
  variable = global l2_err
  variable = global l2_dot_err
  variable = global H1_err
  variable = global linf_err
end

BEGIN RESULTS OUTPUT LABEL diffusion output
  database Name = output{N}.e
  at step 0, increment = {2**N}
```

```
      title Aria cube test
      nodal variables = solution->TEMPERATURE as T
      nodal variables = time_derivative_at_time->TEMPERATURE as TDOT
    END RESULTS OUTPUT LABEL diffusion output

  END ARIA REGION myRegion

 END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.1.10. Transient Heat Conduction: Tet10 Meshes

```
# T_ref_s1 = { T_ref_s1 = 1 }
# h_s1 = { h_s1 = 1 }
# T_ref_s2 = { T_ref_s2 = 2 }
# h_s2 = { h_s2 = 2 }
# C_s = { C_s = 10000.0 }

BEGIN SIERRA myJob
  #{include(exact_transient.inc)}


  BEGIN ARIA MATERIAL Kryptonite
    Density              = Constant rho=1
    Thermal Conductivity = constant k=1
    Specific Heat        = Constant cp=1
    heat conduction      = basic

    user expression = scalar_string_function user_tag = "m_s" \$
      f = "{C_s}*( 1. - exp(-t) + t*exp( -(t-1.0)*(t-1.0) ) )"

    user expression = scalar_string_function user_tag = "m_s_dot" \$
      f = "{C_s}*( exp(-t) + (1.0 + t*( -2.0*(t-1.0)) )*exp( -(t-1.0)*(t-1.0)) )"
  END ARIA MATERIAL Kryptonite

  BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
    BEGIN SUPERLU SOLVER
    END
  END TPETRA EQUATION SOLVER


  BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
    BEGIN CG SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 1000
      RESIDUAL SCALING = NONE
      CONVERGENCE TOLERANCE = 1.000000e-12
    END
  END TPETRA EQUATION SOLVER


  BEGIN FINITE ELEMENT MODEL cube
    database name = cube_h{N}_tet10.e
    coordinate system is cartesian
    decomposition method = rcb

    BEGIN PARAMETERS FOR BLOCK block_1
      material Kryptonite
    END PARAMETERS FOR BLOCK block_1

  END FINITE ELEMENT MODEL cube

  BEGIN PROCEDURE myAriaProcedure
```

```
Begin Solution Control Description
  Use System Main
  Begin System Main
    Begin Transient The_Time_Block
      Advance myRegion
    End
    Simulation Start Time = 0
    Simulation Termination Time = 3
  End
  Begin Parameters For Transient The_Time_Block
    Begin Parameters For Aria Region myRegion
      Initial Time Step Size = {0.5/2**N}
      Time Integration Method = Second_Order
      Time Step Variation = fixed
    End
  End
End

BEGIN ARIA REGION myRegion

  Nonlinear Solution Strategy = Newton
  Maximum Nonlinear Iterations = 10
  Nonlinear Residual Tolerance = 1.0e-12
  Nonlinear Correction Tolerance = 1.0e-12
  Nonlinear Relaxation Factor  = 1.0
  use dof averaged nonlinear residual

  use finite element model cube
  Use Linear Solver Iterative_Solver #Direct_Solver

  EQ ENERGY for TEMPERATURE on block_1 using Q2 with MASS DIFF SRC

  IC const on block_1 temperature = 1.0

  # surface_4: x=0
  # surface_6: x=1

  # surface_3: y=0
  # surface_5: y=1

  # surface_1: z=1
  # surface_2: z=0

  # const Temp BC (x)
  BC const dirichlet at surface_4 Temperature = 1.0
  BC const dirichlet at surface_6 Temperature = 1.0

  # const flux BC (y)
  BC FLUX for Energy on surface_3  = constant flux = 3
  BC FLUX for Energy on surface_3  = scalar_string_function f="exact_flux_y - 3"

  BC FLUX for Energy on surface_5  = constant flux = 5
  BC FLUX for Energy on surface_5  = scalar_string_function f="-exact_flux_y - 5"

  # convective flux BC with const Temp and H (z)
  BC Flux for Energy on surface_1 = Nat_Conv T_Ref={T_ref_s1} H={h_s1}
  BC FLUX for Energy on surface_1  = scalar_string_Function \$
    f="-exact_flux_z - {h_s1}*(exact_sln - {T_ref_s1})"

  BC Flux for Energy on surface_2 = Nat_Conv T_Ref={T_ref_s2} H={h_s2}
  BC FLUX for Energy on surface_2  = scalar_string_Function \$
    f="exact_flux_z - {h_s2}*(exact_sln - {T_ref_s2})"

  #  const source term
  Source For ENERGY on block_1 = Constant value=1
  Source For ENERGY on block_1 = scalar_string_Function f="exact_src-1"

  postprocess max of function "abs(exact_sln - temperature)" on all_blocks as linf_err
```

```
    postprocess l_2_norm of function "exact_sln - temperature" on all_blocks as l2_err
    postprocess l_2_norm of function "exact_sln_dot - dt_temperature" on all_blocks as l2_dot_err
    postprocess integral of function "\$
      (gradExact_sln_x - grad_temperature_x)^2 + \$
      (gradExact_sln_y - grad_temperature_y)^2 + \$
      (gradExact_sln_z - grad_temperature_z)^2" on all_blocks as H1_err_sq
    postprocess global_function "sqrt(H1_err_sq)" as H1_err

    Begin heartbeat testctrl
      Stream name = errors{N}.dat
      Timestamp Format    = ""
      Precision           = 8
      Format = csv
      Labels = Off
      Legend = On
      At step 0, Increment is 1
      variable = global time
      variable = global number_of_nodes
      variable = global l2_err
      variable = global l2_dot_err
      variable = global H1_err
      variable = global linf_err
    end

    BEGIN RESULTS OUTPUT LABEL diffusion output
      database Name = output{N}.e
      at step 0, increment = {2**N}
      title Aria cube test
      nodal variables = solution->TEMPERATURE as T
      nodal variables = time_derivative_at_time->TEMPERATURE as TDOT
    END RESULTS OUTPUT LABEL diffusion output

  END ARIA REGION myRegion

 END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.2. THERMAL BOUNDARY CONDITIONS

12.2.1. Radiative Heat Flux 3.1

12.2.2. Radiative Heat Flux From Fortran User Subroutine

12.2.3. Convective Heat Flux 3.3

12.3. THERMAL CONTACT

12.3.1. 1D Flat Contact 4.1

12.3.1.1. Hex8 Tied

```
#{R=0.0}
begin sierra Aria

    title Adaptive Square

    Begin Universal Aria Expressions
      # gamma = { gamma = (2. - R) / (2. + R) }
      user expression = scalar_string_function user_tag="exactsoln" f="(x < 0.) * (0.5 * (1. + x) * ({gamma} + x)) + (x>=0.) * (
      user expression = scalar_string_function user_tag="exactsolndx" f="(x < 0.) * 0.5 *(({gamma} + x) + 1. + x) + (x>=0.) * 0.
    End

BEGIN ARIA MATERIAL M1
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    BEGIN ARIA MATERIAL M2
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    Begin Finite Element Model bar
        Database Name = 2blocks_contact_unaligned_hex8_h{N}.g
        Begin parameters for block block_1
            material M1
        End
        Begin parameters for block block_2
            material M2
        End
    End Finite Element Model bar

  BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
    BEGIN UMFPACK SOLVER
    END
  END TPETRA EQUATION SOLVER


  BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
    BEGIN GMRES SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 20000
      RESIDUAL SCALING = R0
      CONVERGENCE TOLERANCE = automatic
    END
  END TPETRA EQUATION SOLVER

    begin procedure myProcedure

        Begin Solution Control Description
            Use System Main
            Begin System Main
                Begin Sequential The_Time_Block
                    Advance myRegion
                End
                Simulation Start Time = 0
```

```
        Simulation Termination Time = 1
    End
End

begin Aria region myRegion

    Nonlinear Solution Strategy = Newton
    Maximum Nonlinear Iterations = 10
    Nonlinear Residual Tolerance = 1.0e-12
    Nonlinear Correction Tolerance = 1.0e-12
    Nonlinear Relaxation Factor  = 1.0
    use dof averaged nonlinear residual

    use finite element model bar
    use linear solver Iterative_Solver #Direct_Solver #

    EQ ENERGY for TEMPERATURE on all_blocks using Q1 with DIFF SRC

    Begin  TEMPERATURE BOUNDARY CONDITION xm1
        add surface surface_1
        TEMPERATURE = 0.
    End

    Begin  TEMPERATURE BOUNDARY CONDITION xp1
        add surface surface_2
        TEMPERATURE = 1.
    End

    Begin VOLUME HEATING sm
        add volume block_1
        VALUE = -1.
    End

    Begin VOLUME HEATING sp
        add volume block_2
        VALUE = 1.
    End

    begin contact definition res1
        contact surface surf_1 contains surface_3
        contact surface surf_2 contains surface_4

        begin interaction inter_1
            surfaces = surf_1 surf_2
        end interaction inter_1

        begin enforcement enf_1
            Enforcement for Energy = Tied_Temperature
        end enforcement

    end contact definition res1

    postprocess l_2_norm of function "temperature - exactsoln" on all_blocks as l2_err
    postprocess l_inf_norm of function "temperature - exactsoln" on all_blocks as linf_err
    postprocess l_2_norm of scalar_string_function f = "sqrt((exactsolndx - (grad_temperature_x))^2+(0 - (grad_temperatu

    Begin Heartbeat pp_out
      At Step 1 Interval is 1
      Legend = on
      Labels = off
      Format = csv
      Precision = 5
      Stream Name = errors_h{N}.dat
      Global Time as Time
      Global Number_of_Nodes as Num_Nodes
      Global l2_err
      Global linf_err
      Global h1_err
```

```
            End

Begin Results Output Label diffusion output
                database Name = 2blocks_tied_h{N}.e
                At Step 1, Increment = 1
                Title Calore Two Blocks
                Nodal Variables = solution->temperature as T
                Element Variables = l2_err linf_err h1_err
            End

        end

    end

end
```

12.3.1.2. Hex8 Resistance

```
#{R=4.0}
begin sierra Aria

    title Adaptive Square

    Begin Universal Aria Expressions
      # gamma = { gamma = (2. - R) / (2. + R) }
      user expression = scalar_string_function user_tag="exactsoln" f="(x < 0.) * (0.5 * (1. + x) * ({gamma} + x)) + (x>=0.) * (
      user expression = scalar_string_function user_tag="exactsolndx" f="(x < 0.) * 0.5 *(({gamma} + x) + 1. + x) + (x>=0.) * 0.
    End

    BEGIN ARIA MATERIAL M1
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    BEGIN ARIA MATERIAL M2
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    Begin Finite Element Model bar
        Database Name = 2blocks_contact_unaligned_hex8_h{N}.g
        Begin parameters for block block_1
            material M1
        End
        Begin parameters for block block_2
            material M2
        End
    End Finite Element Model bar

  BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
    BEGIN UMFPACK SOLVER
    END
  END TPETRA EQUATION SOLVER


  BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
    BEGIN GMRES SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 20000
      RESIDUAL SCALING = R0
      CONVERGENCE TOLERANCE = automatic
```

```
  END
END TPETRA EQUATION SOLVER

  begin procedure myProcedure

      Begin Solution Control Description
          Use System Main
          Begin System Main
              Begin Sequential The_Time_Block
                  Advance myRegion
              End
              Simulation Start Time = 0
              Simulation Termination Time = 1
          End
      End

      begin Aria region myRegion

          Nonlinear Solution Strategy = Newton
          Maximum Nonlinear Iterations = 10
          Nonlinear Residual Tolerance = 1.0e-12
          Nonlinear Correction Tolerance = 1.0e-12
          Nonlinear Relaxation Factor  = 1.0
          use dof averaged nonlinear residual

          use finite element model bar
          use linear solver Iterative_Solver #Direct_Solver #

          EQ ENERGY for TEMPERATURE on all_blocks using Q1 with DIFF SRC

          Begin  TEMPERATURE BOUNDARY CONDITION xm1
              add surface surface_1
              TEMPERATURE = 0.
          End

          Begin  TEMPERATURE BOUNDARY CONDITION xp1
              add surface surface_2
              TEMPERATURE = 1.
          End

          Begin VOLUME HEATING sm
              add volume block_1
              VALUE = -1.
          End

          Begin VOLUME HEATING sp
              add volume block_2
              VALUE = 1.
          End

          begin contact definition res1
              contact surface surf_1 contains surface_3
              contact surface surf_2 contains surface_4

              begin interaction inter_1
                  surfaces = surf_1 surf_2
              end interaction inter_1

              begin enforcement enf_1
                  gap conductance coefficient = constant value= {1.0/R}
                  Enforcement for Energy = gap_conductance
              end enforcement

          end contact definition res1

          postprocess value of expression exactsoln on all_blocks as exact_soln
          postprocess value of function "temperature - exactsoln" on all_blocks as nodal_l2_err
          postprocess l_2_norm of function "temperature - exactsoln" on all_blocks as l2_err
```

```
            # The linf on the hex8 mesh is just always O(1e-11)
            postprocess l_1_norm of function "temperature - exactsoln" on block_2 as l1_err
            postprocess l_2_norm of scalar_string_function f = "sqrt((exactsolndx - (grad_temperature_x))^2+(0 - (grad_temperatu

            Begin Heartbeat pp_out
              At Step 1 Interval is 1
              Legend = on
              Labels = off
              Format = csv
              Precision = 5
              Stream Name = errors_h{N}.dat
              Global Time as Time
              Global Number_of_Nodes as Num_Nodes
              Global l2_err
              Global l1_err
              Global h1_err
            End

            Begin Results Output Label diffusion output
                database Name = 2blocks_res_h{N}.e
                At Step 1, Increment = 1
                Title Calore Two Blocks
                Nodal Variables = solution->temperature as T
                Nodal Variables = nodal_l2_err
                Nodal Variables = exact_soln
                Element Variables = l2_err linf_err h1_err
            End

        end

    end

end
```

12.3.1.3. Tet4 Tied

```
#{R=0.0}
begin sierra Aria

    title Adaptive Square

    Begin Universal Aria Expressions
      # gamma = { gamma = (2. - R) / (2. + R) }
      user expression = scalar_string_function user_tag="exactsoln" f="(x < 0.) * (0.5 * (1. + x) * ({gamma} + x)) + (x>=0.) * (
      user expression = scalar_string_function user_tag="exactsolndx" f="(x < 0.) * 0.5 *(({gamma} + x) + 1. + x) + (x>=0.) * 0.
    End

    BEGIN ARIA MATERIAL M1
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    BEGIN ARIA MATERIAL M2
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    Begin Finite Element Model bar
        Database Name = 2blocks_contact_unaligned_tet4_h{N}.g
        Begin parameters for block block_1
            material M1
        End
```

```
      Begin parameters for block block_2
          material M2
      End
  End Finite Element Model bar

BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
  BEGIN UMFPACK SOLVER
  END
END TPETRA EQUATION SOLVER


BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN GMRES SOLVER
    BEGIN DD-ILU PRECONDITIONER
      subdomain overlap level = 1
    END
    MAXIMUM ITERATIONS = 200
    RESIDUAL SCALING = R0
    CONVERGENCE TOLERANCE = automatic
  END
END TPETRA EQUATION SOLVER

  begin procedure myProcedure

      Begin Solution Control Description
          Use System Main
          Begin System Main
              Begin Sequential The_Time_Block
                  Advance myRegion
              End
              Simulation Start Time = 0
              Simulation Termination Time = 1
          End
      End

      begin Aria region myRegion

          Nonlinear Solution Strategy = Newton
          Maximum Nonlinear Iterations = 10
          Nonlinear Residual Tolerance = 1.0e-12
          Nonlinear Correction Tolerance = 1.0e-12
          Nonlinear Relaxation Factor  = 1.0
          use dof averaged nonlinear residual

          use finite element model bar
          use linear solver Iterative_Solver #Direct_Solver #

          EQ ENERGY for TEMPERATURE on all_blocks using Q1 with DIFF SRC

          Begin  TEMPERATURE BOUNDARY CONDITION xm1
              add surface surface_1
              TEMPERATURE = 0.
          End

          Begin  TEMPERATURE BOUNDARY CONDITION xp1
              add surface surface_2
              TEMPERATURE = 1.
          End

          Begin VOLUME HEATING sm
              add volume block_1
              VALUE = -1.
          End

          Begin VOLUME HEATING sp
              add volume block_2
              VALUE = 1.
          End
```

```
        begin contact definition res1
            contact surface surf_1 contains surface_3
            contact surface surf_2 contains surface_4

            begin interaction inter_1
                surfaces = surf_1 surf_2
            end interaction inter_1

            begin enforcement enf_1
                Enforcement for Energy = Tied_Temperature
            end enforcement

        end contact definition res1

        postprocess l_2_norm of function "temperature - exactsoln" on all_blocks as l2_err
        postprocess l_inf_norm of function "temperature - exactsoln" on all_blocks as linf_err
        postprocess l_2_norm of scalar_string_function f = "sqrt((exactsolndx - (grad_temperature_x))^2+(0 - (grad_temperatu

        Begin Heartbeat pp_out
          At Step 1 Interval is 1
          Legend = on
          Labels = off
          Format = csv
          Precision = 5
          Stream Name = errors_h{N}.dat
          Global Time as Time
          Global Number_of_Nodes as Num_Nodes
          Global l2_err
          Global linf_err
          Global h1_err
        End

        Begin Results Output Label diffusion output
            database Name = 2blocks_tied_h{N}.e
            At Step 1, Increment = 1
            Title Calore Two Blocks
            Nodal Variables = solution->temperature as T
            Element Variables = l2_err linf_err h1_err
        End

    end

  end

end
```

12.3.1.4. Tet4 Resistance

```
#{R=4.0}
begin sierra Aria

    title Adaptive Square

    Begin Universal Aria Expressions
      # gamma = { gamma = (2. - R) / (2. + R) }
      user expression = scalar_string_function user_tag="exactsoln" f="(x < 0.) * (0.5 * (1. + x) * ({gamma} + x)) + (x>=0.) * (
      user expression = scalar_string_function user_tag="exactsolndx" f="(x < 0.) * 0.5 *(({gamma} + x) + 1. + x) + (x>=0.) * 0.
    End

    BEGIN ARIA MATERIAL M1
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END
```

```
  BEGIN ARIA MATERIAL M2
      Density              = Constant rho=1
      Thermal Conductivity = constant k=1
      Specific Heat        = Constant cp=1
      heat conduction      = basic
  END

  Begin Finite Element Model bar
      Database Name = 2blocks_contact_unaligned_tet4_h{N}.g
      Begin parameters for block block_1
          material M1
      End
      Begin parameters for block block_2
          material M2
      End
  End Finite Element Model bar

BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
  BEGIN UMFPACK SOLVER
  END
END TPETRA EQUATION SOLVER


BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN GMRES SOLVER
    BEGIN DD-ILU PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 200
    RESIDUAL SCALING = R0
    CONVERGENCE TOLERANCE = automatic
  END
END TPETRA EQUATION SOLVER

  begin procedure myProcedure

      Begin Solution Control Description
          Use System Main
          Begin System Main
              Begin Sequential The_Time_Block
                  Advance myRegion
              End
              Simulation Start Time = 0
              Simulation Termination Time = 1
          End
      End

      begin Aria region myRegion

          Nonlinear Solution Strategy = Newton
          Maximum Nonlinear Iterations = 10
          Nonlinear Residual Tolerance = 1.0e-12
          Nonlinear Correction Tolerance = 1.0e-12
          Nonlinear Relaxation Factor  = 1.0
          use dof averaged nonlinear residual

          use finite element model bar
          use linear solver Iterative_Solver #Direct_Solver #

          EQ ENERGY for TEMPERATURE on all_blocks using Q1 with DIFF SRC

          Begin  TEMPERATURE BOUNDARY CONDITION xm1
              add surface surface_1
              TEMPERATURE = 0.
          End

          Begin  TEMPERATURE BOUNDARY CONDITION xp1
              add surface surface_2
```

```
            TEMPERATURE = 1.
        End

        Begin VOLUME HEATING sm
            add volume block_1
            VALUE = -1.
        End

        Begin VOLUME HEATING sp
            add volume block_2
            VALUE = 1.
        End

        begin contact definition res1
            contact surface surf_1 contains surface_3
            contact surface surf_2 contains surface_4

            begin interaction inter_1
                surfaces = surf_1 surf_2
            end interaction inter_1

            begin enforcement enf_1
                gap conductance coefficient = constant value= {1.0/R}
                Enforcement for Energy = gap_conductance
            end enforcement

        end contact definition res1

        postprocess l_2_norm of function "temperature - exactsoln" on all_blocks as l2_err
        postprocess l_inf_norm of function "temperature - exactsoln" on all_blocks as linf_err
        postprocess l_2_norm of scalar_string_function f = "sqrt((exactsolndx - (grad_temperature_x))^2+(0 - (grad_temperatu

        Begin Heartbeat pp_out
          At Step 1 Interval is 1
          Legend = on
          Labels = off
          Format = csv
          Precision = 5
          Stream Name = errors_h{N}.dat
          Global Time as Time
          Global Number_of_Nodes as Num_Nodes
          Global l2_err
          Global linf_err
          Global h1_err
        End

        Begin Results Output Label diffusion output
            database Name = 2blocks_res_h{N}.e
            At Step 1, Increment = 1
            Title Calore Two Blocks
            Nodal Variables = solution->temperature as T
            Element Variables = l2_err linf_err h1_err
        End

    end

  end

end
```

12.3.1.5. Hex8-Tet4 Tied

```
#{R=0.0}
begin sierra Aria

  title Adaptive Square
```

```
  Begin Universal Aria Expressions
    # gamma = { gamma = (2. - R) / (2. + R) }
    user expression = scalar_string_function user_tag="exactsoln" f="(x < 0.) * (0.5 * (1. + x) * ({gamma} + x)) + (x>=0.) * (
    user expression = scalar_string_function user_tag="exactsolndx" f="(x < 0.) * 0.5 *(({gamma} + x) + 1. + x) + (x>=0.) * 0.
  End

  BEGIN ARIA MATERIAL M1
      Density              = Constant rho=1
      Thermal Conductivity = constant k=1
      Specific Heat        = Constant cp=1
      heat conduction      = basic
  END

  BEGIN ARIA MATERIAL M2
      Density              = Constant rho=1
      Thermal Conductivity = constant k=1
      Specific Heat        = Constant cp=1
      heat conduction      = basic
  END

  Begin Finite Element Model bar
      Database Name = 2blocks_contact_unaligned_hex8_tet4_h{N}.g
      Begin parameters for block block_1
          material M1
      End
      Begin parameters for block block_2
          material M2
      End
  End Finite Element Model bar

BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
  BEGIN UMFPACK SOLVER
  END
END TPETRA EQUATION SOLVER


BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN GMRES SOLVER
    BEGIN DD-ILU PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 200
    RESIDUAL SCALING = R0
    CONVERGENCE TOLERANCE = automatic
  END
END TPETRA EQUATION SOLVER

  begin procedure myProcedure

      Begin Solution Control Description
          Use System Main
          Begin System Main
              Begin Sequential The_Time_Block
                  Advance myRegion
              End
              Simulation Start Time = 0
              Simulation Termination Time = 1
          End
      End

      begin Aria region myRegion

          Nonlinear Solution Strategy = Newton
          Maximum Nonlinear Iterations = 10
          Nonlinear Residual Tolerance = 1.0e-12
          Nonlinear Correction Tolerance = 1.0e-12
          Nonlinear Relaxation Factor  = 1.0
          use dof averaged nonlinear residual
```

```
            use finite element model bar
            use linear solver Iterative_Solver #Direct_Solver #

            EQ ENERGY for TEMPERATURE on all_blocks using Q1 with DIFF SRC

            Begin  TEMPERATURE BOUNDARY CONDITION xm1
                add surface surface_1
                TEMPERATURE = 0.
            End

            Begin  TEMPERATURE BOUNDARY CONDITION xp1
                add surface surface_2
                TEMPERATURE = 1.
            End

            Begin VOLUME HEATING sm
                add volume block_1
                VALUE = -1.
            End

            Begin VOLUME HEATING sp
                add volume block_2
                VALUE = 1.
            End

            begin contact definition res1
                contact surface surf_1 contains surface_3
                contact surface surf_2 contains surface_4

                begin interaction inter_1
                    surfaces = surf_1 surf_2
                end interaction inter_1

                begin enforcement enf_1
                    Enforcement for Energy = Tied_Temperature
                end enforcement

            end contact definition res1

            postprocess l_2_norm of function "temperature - exactsoln" on all_blocks as l2_err
            postprocess l_inf_norm of function "temperature - exactsoln" on all_blocks as linf_err
            postprocess l_2_norm of scalar_string_function f = "sqrt((exactsolndx - (grad_temperature_x))^2+(0 - (grad_temperatu

            Begin Heartbeat pp_out
              At Step 1 Interval is 1
              Legend = on
              Labels = off
              Format = csv
              Precision = 5
              Stream Name = errors_h{N}.dat
              Global Time as Time
              Global Number_of_Nodes as Num_Nodes
              Global l2_err
              Global linf_err
              Global h1_err
            End

Begin Results Output Label diffusion output
                database Name = 2blocks_tied_h{N}.e
                At Step 1, Increment = 1
                Title Calore Two Blocks
                Nodal Variables = solution->temperature as T
                Element Variables = l2_err linf_err h1_err
            End

        end
```

```
    end

end
```

12.3.1.6. Hex8-Tet4 Resistance

```
#{R=4.0}
begin sierra Aria

    title Adaptive Square

    Begin Universal Aria Expressions
      # gamma = { gamma = (2. - R) / (2. + R) }
      user expression = scalar_string_function user_tag="exactsoln" f="(x < 0.) * (0.5 * (1 + x) * ({gamma} + x)) + (x>=0.) * (1
      user expression = scalar_string_function user_tag="exactsolndx" f="(x < 0.) * 0.5 *(({gamma} + x) + 1. + x) + (x>=0.) * 0.
    End


    BEGIN ARIA MATERIAL M1
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    BEGIN ARIA MATERIAL M2
        Density              = Constant rho=1
        Thermal Conductivity = constant k=1
        Specific Heat        = Constant cp=1
        heat conduction      = basic
    END

    Begin Finite Element Model bar
        Database Name = 2blocks_contact_unaligned_hex8_tet4_h{N}.g
        Begin parameters for block block_1
            material M1
        End
        Begin parameters for block block_2
            material M2
        End
    End Finite Element Model bar

  BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
    BEGIN UMFPACK SOLVER
    END
  END TPETRA EQUATION SOLVER


  BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
    BEGIN GMRES SOLVER
      BEGIN DD-ILU PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 200
      RESIDUAL SCALING = R0
      CONVERGENCE TOLERANCE = automatic
    END
  END TPETRA EQUATION SOLVER

    begin procedure myProcedure

        Begin Solution Control Description
            Use System Main
            Begin System Main
                Begin Sequential The_Time_Block
                    Advance myRegion
                End
```

```
        Simulation Start Time = 0
        Simulation Termination Time = 1
    End
End

begin Aria region myRegion

    Nonlinear Solution Strategy = Newton
    Maximum Nonlinear Iterations = 10
    Nonlinear Residual Tolerance = 1.0e-12
    Nonlinear Correction Tolerance = 1.0e-12
    Nonlinear Relaxation Factor  = 1.0
    use dof averaged nonlinear residual

    use finite element model bar
    use linear solver Iterative_Solver #Direct_Solver #

    EQ ENERGY for TEMPERATURE on all_blocks using Q1 with DIFF SRC

    Begin  TEMPERATURE BOUNDARY CONDITION xm1
        add surface surface_1
        TEMPERATURE = 0.
    End

    Begin  TEMPERATURE BOUNDARY CONDITION xp1
        add surface surface_2
        TEMPERATURE = 1.
    End

    Begin VOLUME HEATING sm
        add volume block_1
        VALUE = -1.
    End

    Begin VOLUME HEATING sp
        add volume block_2
        VALUE = 1.
    End

    begin contact definition res1
        contact surface surf_1 contains surface_3
        contact surface surf_2 contains surface_4

        begin interaction inter_1
            surfaces = surf_1 surf_2
        end interaction inter_1

        begin enforcement enf_1
            gap conductance coefficient = constant value= {1.0/R}
            Enforcement for Energy = gap_conductance
        end enforcement

    end contact definition res1

    postprocess l_2_norm of function "temperature - exactsoln" on all_blocks as l2_err
    postprocess l_inf_norm of function "temperature - exactsoln" on all_blocks as linf_err
    postprocess l_2_norm of scalar_string_function f = "sqrt((exactsolndx - (grad_temperature_x))^2+(0 - (grad_temperatu

    Begin Heartbeat pp_out
      At Step 1 Interval is 1
      Legend = on
      Labels = off
      Format = csv
      Precision = 5
      Stream Name = errors_h{N}.dat
      Global Time as Time
      Global Number_of_Nodes as Num_Nodes
      Global l2_err
```

```
      Global linf_err
      Global h1_err
    End

    Begin Results Output Label diffusion output
      database Name = 2blocks_res_h{N}.e
      At Step 1, Increment = 1
      Title Calore Two Blocks
      Nodal Variables = solution->temperature as T
      Element Variables = l2_err linf_err h1_err
    End

  end

 end

end
```

12.3.2. 3D Curved Contact 4.2

12.3.2.1. Hex8-Hex8 Case

12.3.2.2. Tet4-Tet4 Case

12.3.2.3. Hex8-Tet4 Case

12.3.3. Steady Hex8 Contact

12.3.4. Steady Hex20 Contact

12.3.5. Steady Hex27 Contact

12.3.6. Steady Tet4 Contact

```
#N={N=5}
BEGIN SIERRA myJob

  Begin universal aria expressions
    User Expression = Scalar_String_Function \$
      f = "(x-x^2)^2*(y-y^2)^2*(z-z^2)^2 + 1" \$
      user_tag = T_mms

    User Expression = Vector_String_Function \$
      f_x = "2*(x-x^2)*(1-2*x)*(y-y^2)^2*(z-z^2)^2" \$
      f_y = "2*(y-y^2)*(1-2*y)*(x-x^2)^2*(z-z^2)^2" \$
      f_z = "2*(z-z^2)*(1-2*z)*(x-x^2)^2*(y-y^2)^2" \$
      user_tag = gradT

    User Expression = Scalar_String_Function \$
      f = "2*((1-2*x)^2 - 2*(x-x^2))*(y-y^2)^2*(z-z^2)^2  \$
         + 2*((1-2*y)^2 - 2*(y-y^2))*(x-x^2)^2*(z-z^2)^2  \$
         + 2*((1-2*z)^2 - 2*(z-z^2))*(x-x^2)^2*(y-y^2)^2" \$
      user_tag = laplaceT

    User Expression = Scalar_String_Function \$
      f = "-laplaceT - 1.0" \$
      user_tag = mms_src
  End
```

```
 BEGIN ARIA MATERIAL Kryptonite
   Density              = Constant rho=1
   Thermal Conductivity = constant k=1
   Specific Heat        = Constant cp=1
   heat conduction      = basic
 END ARIA MATERIAL Kryptonite

BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN GMRES SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = NONE
    CONVERGENCE TOLERANCE = 1.000000e-15
  END
END TPETRA EQUATION SOLVER


 BEGIN FINITE ELEMENT MODEL cube
    database name = cube_contact_h{N}_tet4.e
    coordinate system is cartesian
    decomposition method = rcb

    BEGIN PARAMETERS FOR BLOCK block_1
       material Kryptonite
    END PARAMETERS FOR BLOCK block_1

    BEGIN PARAMETERS FOR BLOCK block_2
       material Kryptonite
    END PARAMETERS FOR BLOCK block_2

 END FINITE ELEMENT MODEL cube

 BEGIN PROCEDURE myAriaProcedure

    Begin Solution Control Description
       Use System Main
       Begin System Main
          Begin Sequential The_Time_Block
             Advance myRegion
          End
            Simulation Start Time = 0
          Simulation Termination Time = 1
       End
    End

    BEGIN ARIA REGION myRegion

       Nonlinear Solution Strategy = Newton
       Maximum Nonlinear Iterations = 10
       Nonlinear Residual Tolerance = 1.0e-12
       Nonlinear Correction Tolerance = 1.0e-12
       Nonlinear Relaxation Factor  = 1.0
       use dof averaged nonlinear residual

       use finite element model cube
       Use Linear Solver Iterative_Solver #Direct_Solver

       EQ ENERGY for TEMPERATURE on block_1 using Q1 with DIFF SRC
       EQ ENERGY for TEMPERATURE on block_2 using Q1 with DIFF SRC

       # surface_4: x=0
       # surface_6: x=1

       # surface_3: y=0
       # surface_5: y=1

       # surface_1: z=1
```

```
      # surface_2: z=0

      # const Temp BC (x)
      BC const dirichlet at surface_4 Temperature = 1.0
      BC const dirichlet at surface_6 Temperature = 1.0

      # const flux BC (y)
      BC FLUX for Energy on surface_3  = constant flux = 3
      BC FLUX for Energy on surface_5  = constant flux = 5

      BC Flux for Energy on surface_3 = Scalar_String_Function f = "-gradT_y - 3.0"
      BC Flux for Energy on surface_5 = Scalar_String_Function f = "gradT_y - 5.0"

      # convective flux BC with const Temp and H (z)
      BC Flux for Energy on surface_1 = Nat_Conv T_Ref=1 H=1
      BC Flux for Energy on surface_2 = Nat_Conv T_Ref=2 H=2

      BC Flux for Energy on surface_1 = Scalar_String_Function f = "gradT_z - (T_mms-1.0)"
      BC Flux for Energy on surface_2 = Scalar_String_Function f = "-gradT_z - 2.0*(T_mms-2.0)"


      #  const source term
      Source For ENERGY on block_1 = Constant value=1
      Source For ENERGY on block_2 = Constant value=1

      Source for ENERGY on block_1 = Scalar_String_Function f = "mms_src"
      Source for ENERGY on block_2 = Scalar_String_Function f = "mms_src"

      begin contact definition mpc1
        contact surface surf_1 contains surface_7
        contact surface surf_2 contains surface_8

        begin interaction inter_1
          surfaces = surf_1 surf_2
        end
        begin enforcement enf_1
          Enforcement for Energy = Tied_Temperature
        end
      end

     postprocess L_2_norm of function "temperature-T_mms" on all_blocks as l2
     postprocess L_inf_norm of function "temperature-T_mms" on all_blocks as linf
     postprocess L_2_norm of function "sqrt((grad_temperature_x-gradT_x)^2+(grad_temperature_y-gradT_y)^2+(grad_temperature_z

     Begin Heartbeat hb
       At Step 0 Interval is 1
       Precision = 6
       Stream Name = errors_h{N}.dat
       legend = off
       labels = off
       Timestamp Format    = ""
       variable = global time
       variable = global number_of_nodes
       variable = global l2
       variable = global h1
       variable = global linf
     End

      BEGIN RESULTS OUTPUT LABEL diffusion output
         database Name = thermal_steady_tet4_tied_contact_h{N}.e
         at step 1, increment = 1
         title Aria cube test
         nodal variables = nonlinear_solution->TEMPERATURE as T
      END RESULTS OUTPUT LABEL diffusion output

   END ARIA REGION myRegion

END PROCEDURE myAriaProcedure
```

```
END SIERRA myJob
```

12.3.7. Steady Tet4Tet10 Contact

```
#N={N=5}
BEGIN SIERRA myJob

  Begin universal aria expressions
    User Expression = Scalar_String_Function \$
      f = "(x-x^2)^2*(y-y^2)^2*(z-z^2)^2 + 1" \$
      user_tag = T_mms

    User Expression = Vector_String_Function \$
      f_x = "2*(x-x^2)*(1-2*x)*(y-y^2)^2*(z-z^2)^2" \$
      f_y = "2*(y-y^2)*(1-2*y)*(x-x^2)^2*(z-z^2)^2" \$
      f_z = "2*(z-z^2)*(1-2*z)*(x-x^2)^2*(y-y^2)^2" \$
      user_tag = gradT

    User Expression = Scalar_String_Function \$
      f = "2*((1-2*x)^2 - 2*(x-x^2))*(y-y^2)^2*(z-z^2)^2  \$
         + 2*((1-2*y)^2 - 2*(y-y^2))*(x-x^2)^2*(z-z^2)^2  \$
         + 2*((1-2*z)^2 - 2*(z-z^2))*(x-x^2)^2*(y-y^2)^2" \$
      user_tag = laplaceT

    User Expression = Scalar_String_Function \$
      f = "-laplaceT - 1.0" \$
      user_tag = mms_src
  End

   BEGIN ARIA MATERIAL Kryptonite
     Density              = Constant rho=1
     Thermal Conductivity = constant k=1
     Specific Heat        = Constant cp=1
     heat conduction      = basic
   END ARIA MATERIAL Kryptonite

  BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
    BEGIN GMRES SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 1000
      RESIDUAL SCALING = NONE
      CONVERGENCE TOLERANCE = 1.000000e-15
    END
  END TPETRA EQUATION SOLVER


   BEGIN FINITE ELEMENT MODEL cube
      database name = cube_contact_h{N}_tet10.e
      coordinate system is cartesian
      decomposition method = rcb

      BEGIN PARAMETERS FOR BLOCK block_1
         material Kryptonite
      END PARAMETERS FOR BLOCK block_1

      BEGIN PARAMETERS FOR BLOCK block_2
         material Kryptonite
      END PARAMETERS FOR BLOCK block_2

   END FINITE ELEMENT MODEL cube


   BEGIN PROCEDURE myAriaProcedure
```

```
Begin Solution Control Description
   Use System Main
   Begin System Main
      Begin Sequential The_Time_Block
         Advance myRegion
      End
        Simulation Start Time = 0
      Simulation Termination Time = 1
   End
End

BEGIN ARIA REGION myRegion

   Nonlinear Solution Strategy = Newton
   Maximum Nonlinear Iterations = 10
   Nonlinear Residual Tolerance = 1.0e-12
   Nonlinear Correction Tolerance = 1.0e-12
   Nonlinear Relaxation Factor  = 1.0
   use dof averaged nonlinear residual

   use finite element model cube
   Use Linear Solver Iterative_Solver #Direct_Solver

   EQ ENERGY for TEMPERATURE on block_1 using Q1 with DIFF SRC
   EQ ENERGY for TEMPERATURE on block_2 using Q1 with DIFF SRC

   # surface_4: x=0
   # surface_6: x=1

   # surface_3: y=0
   # surface_5: y=1

   # surface_1: z=1
   # surface_2: z=0

   # const Temp BC (x)
   BC const dirichlet at surface_4 Temperature = 1.0
   BC const dirichlet at surface_6 Temperature = 1.0

   # const flux BC (y)
   BC FLUX for Energy on surface_3  = constant flux = 3
   BC FLUX for Energy on surface_5  = constant flux = 5

   BC Flux for Energy on surface_3 = Scalar_String_Function f = "-gradT_y - 3.0"
   BC Flux for Energy on surface_5 = Scalar_String_Function f = "gradT_y - 5.0"

   # convective flux BC with const Temp and H (z)
   BC Flux for Energy on surface_1 = Nat_Conv T_Ref=1 H=1
   BC Flux for Energy on surface_2 = Nat_Conv T_Ref=2 H=2

   BC Flux for Energy on surface_1 = Scalar_String_Function f = "gradT_z - (T_mms-1.0)"
   BC Flux for Energy on surface_2 = Scalar_String_Function f = "-gradT_z - 2.0*(T_mms-2.0)"

   #  const source term
   Source For ENERGY on block_1 = Constant value=1
   Source For ENERGY on block_2 = Constant value=1

   Source for ENERGY on block_1 = Scalar_String_Function f = "mms_src"
   Source for ENERGY on block_2 = Scalar_String_Function f = "mms_src"

   begin contact definition mpc1
     contact surface surf_1 contains surface_7
     contact surface surf_2 contains surface_8

     begin interaction inter_1
       surfaces = surf_1 surf_2
     end
     begin enforcement enf_1
```

```
          Enforcement for Energy = Tied_Temperature
        end
      end

     postprocess L_2_norm of function "temperature-T_mms" on all_blocks as l2
     postprocess L_inf_norm of function "temperature-T_mms" on all_blocks as linf
     postprocess L_2_norm of function "sqrt((grad_temperature_x-gradT_x)^2+(grad_temperature_y-gradT_y)^2+(grad_temperature_z

     Begin Heartbeat hb
       At Step 0 Interval is 1
       Precision = 6
       Stream Name = errors_h{N}.dat
       legend = off
       labels = off
       Timestamp Format    = ""
       variable = global time
       variable = global number_of_nodes
       variable = global l2
       variable = global h1
       variable = global linf
     End

      BEGIN RESULTS OUTPUT LABEL diffusion output
         database Name = thermal_steady_tet4_tied_contact_h{N}.e
         at step 1, increment = 1
         # time interval is 1.0
         title Aria cube test
         nodal variables = nonlinear_solution->TEMPERATURE as T
         element variables = l2_error h1_error linf_error
      END RESULTS OUTPUT LABEL diffusion output

   END ARIA REGION myRegion

 END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.3.8. Steady Tet10 Contact

12.3.9. Steady Tet10 Dash Contact

12.3.10. Transient Tet4Tet10 Contact

```
#N={N=5}
# T_ref_s1 = { T_ref_s1 = 1 }
# h_s1 = { h_s1 = 1 }
# T_ref_s2 = { T_ref_s2 = 2 }
# h_s2 = { h_s2 = 2 }
# C_s = { C_s = 10000.0 }

BEGIN SIERRA myJob
  #{include(exact_transient.inc)}


  BEGIN ARIA MATERIAL Kryptonite
    Density              = Constant rho={rho}
    Thermal Conductivity = constant k={k}
    Specific Heat        = Constant cp={Cp}
    heat conduction      = basic

    user expression = scalar_string_function user_tag = "m_s" \$
      f = "{C_s}*( 1. - exp(-t) + t*exp( -(t-1.0)*(t-1.0) ) )"
```

```
  user expression = scalar_string_function user_tag = "m_s_dot" \$
    f = "{C_s}*( exp(-t) + (1.0 + t*( -2.0*(t-1.0)) )*exp( -(t-1.0)*(t-1.0)) )"
END ARIA MATERIAL Kryptonite

BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
  BEGIN SUPERLU SOLVER
  END
END TPETRA EQUATION SOLVER


BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
  BEGIN GMRES SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = NONE
    CONVERGENCE TOLERANCE = 1.000000e-15
  END
END TPETRA EQUATION SOLVER


BEGIN FINITE ELEMENT MODEL cube
  database name = cube_contact_h{N}_tet10.e
  coordinate system is cartesian
  decomposition method = rcb

  BEGIN PARAMETERS FOR BLOCK block_1
    material Kryptonite
  END PARAMETERS FOR BLOCK block_1

  BEGIN PARAMETERS FOR BLOCK block_2
    material Kryptonite
  END PARAMETERS FOR BLOCK block_2

END FINITE ELEMENT MODEL cube

BEGIN PROCEDURE myAriaProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Begin Transient The_Time_Block
        Advance myRegion
      End
      Simulation Start Time = 0
      Simulation Termination Time = 3
    End
    Begin Parameters For Transient The_Time_Block
      Begin Parameters For Aria Region myRegion
        Initial Time Step Size = {0.5/2**N}
        Time Integration Method = Second_Order
        Time Step Variation = fixed
      End
    End
  End

  BEGIN ARIA REGION myRegion

    Nonlinear Solution Strategy = Newton
    Maximum Nonlinear Iterations = 10
    Nonlinear Residual Tolerance = 1.0e-12
    Nonlinear Correction Tolerance = 1.0e-12
    Nonlinear Relaxation Factor  = 1.0
    use dof averaged nonlinear residual

    use finite element model cube
    Use Linear Solver Iterative_Solver #Direct_Solver
```

```
EQ ENERGY for TEMPERATURE on block_1 using Q1 with MASS DIFF SRC
EQ ENERGY for TEMPERATURE on block_2 using Q1 with MASS DIFF SRC

IC const on block_1 temperature = 1.0
IC const on block_2 temperature = 1.0

# surface_4: x=0
# surface_6: x=1

# surface_3: y=0
# surface_5: y=1

# surface_1: z=1
# surface_2: z=0

# const Temp BC (x)
BC const dirichlet at surface_4 Temperature = 1.0
BC const dirichlet at surface_6 Temperature = 1.0

# const flux BC (y)
BC FLUX for Energy on surface_3  = constant flux = 3
BC FLUX for Energy on surface_3  = scalar_string_function f="exact_flux_y - 3"

BC FLUX for Energy on surface_5  = constant flux = 5
BC FLUX for Energy on surface_5  = scalar_string_function f="-exact_flux_y - 5"

# convective flux BC with const Temp and H (z)
BC Flux for Energy on surface_1 = Nat_Conv T_Ref={T_ref_s1} H={h_s1}
BC FLUX for Energy on surface_1  = scalar_string_Function \$
  f="-exact_flux_z - {h_s1}*(exact_sln - {T_ref_s1})"

BC Flux for Energy on surface_2 = Nat_Conv T_Ref={T_ref_s2} H={h_s2}
BC FLUX for Energy on surface_2  = scalar_string_Function \$
  f="exact_flux_z - {h_s2}*(exact_sln - {T_ref_s2})"

#  const source term
Source For ENERGY on block_1 = Constant value=1
Source For ENERGY on block_1 = scalar_string_Function f="exact_src-1"

Source For ENERGY on block_2 = Constant value=1
Source For ENERGY on block_2 = scalar_string_Function f="exact_src-1"

begin contact definition mpc1
  contact surface surf_1 contains surface_7
  contact surface surf_2 contains surface_8

  begin interaction inter_1
    surfaces = surf_1 surf_2
  end
  begin enforcement enf_1
    Enforcement for Energy = Tied_Temperature
  end
end

postprocess max of function "abs(exact_sln - temperature)" on all_blocks as linf_err
postprocess l_2_norm of function "exact_sln - temperature" on all_blocks as l2_err
postprocess l_2_norm of function "exact_sln_dot - dt_temperature" on all_blocks as l2_dot_err
postprocess integral of function "\$
  (gradExact_sln_x - grad_temperature_x)^2 + \$
  (gradExact_sln_y - grad_temperature_y)^2 + \$
  (gradExact_sln_z - grad_temperature_z)^2" on all_blocks as H1_err_sq
postprocess global_function "sqrt(H1_err_sq)" as H1_err

Begin heartbeat testctrl
  Stream name = errors_h{N}.dat
  Timestamp Format    = ""
  Precision           = 8
  Format = csv
```

```
        Labels = Off
        Legend = On
        At step 0, Increment is 1
        variable = global time
        variable = global number_of_nodes
        variable = global l2_err
        variable = global l2_dot_err
        variable = global H1_err
        variable = global linf_err
      end

      BEGIN RESULTS OUTPUT LABEL diffusion output
        database Name = thermal_transient_tet10_tied_contact_h{N}.e
        at step 0, increment = {2**N}
        title Aria cube test
        nodal variables = solution->TEMPERATURE as T
        nodal variables = time_derivative_at_time->TEMPERATURE as TDOT
      END RESULTS OUTPUT LABEL diffusion output

    END ARIA REGION myRegion

  END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.3.11. Transient Tet10 Contact

```
# T_ref_s1 = { T_ref_s1 = 1 }
# h_s1 = { h_s1 = 1 }
# T_ref_s2 = { T_ref_s2 = 2 }
# h_s2 = { h_s2 = 2 }
# C_s = { C_s = 10000.0 }

BEGIN SIERRA myJob
  #{include(exact_transient.inc)}


  BEGIN ARIA MATERIAL Kryptonite
    Density              = Constant rho=1
    Thermal Conductivity = constant k=1
    Specific Heat        = Constant cp=1
    heat conduction      = basic

    user expression = scalar_string_function user_tag = "m_s" \$
      f = "{C_s}*( 1. - exp(-t) + t*exp( -(t-1.0)*(t-1.0) ) )"

    user expression = scalar_string_function user_tag = "m_s_dot" \$
      f = "{C_s}*( exp(-t) + (1.0 + t*( -2.0*(t-1.0)) )*exp( -(t-1.0)*(t-1.0)) )"
  END ARIA MATERIAL Kryptonite

  BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
    BEGIN SUPERLU SOLVER
    END
  END TPETRA EQUATION SOLVER


  BEGIN TPETRA EQUATION SOLVER  ITERATIVE_SOLVER
    BEGIN GMRES SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 1000
      RESIDUAL SCALING = NONE
      CONVERGENCE TOLERANCE = 1.000000e-15
    END
  END TPETRA EQUATION SOLVER
```

```
BEGIN FINITE ELEMENT MODEL cube
  database name = cube_contact_h{N}_tet10.e
  coordinate system is cartesian
  decomposition method = rcb

  BEGIN PARAMETERS FOR BLOCK block_1
    material Kryptonite
  END PARAMETERS FOR BLOCK block_1

  BEGIN PARAMETERS FOR BLOCK block_2
    material Kryptonite
  END PARAMETERS FOR BLOCK block_2

END FINITE ELEMENT MODEL cube

BEGIN PROCEDURE myAriaProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Begin Transient The_Time_Block
        Advance myRegion
      End
      Simulation Start Time = 0
      Simulation Termination Time = 3
    End
    Begin Parameters For Transient The_Time_Block
      Begin Parameters For Aria Region myRegion
        Initial Time Step Size = {0.5/2**N}
        Time Integration Method = Second_Order
        Time Step Variation = fixed
      End
    End
  End

  BEGIN ARIA REGION myRegion

    Nonlinear Solution Strategy = Newton
    Maximum Nonlinear Iterations = 10
    Nonlinear Residual Tolerance = 1.0e-12
    Nonlinear Correction Tolerance = 1.0e-12
    Nonlinear Relaxation Factor  = 1.0
    use dof averaged nonlinear residual

    use finite element model cube
    Use Linear Solver Iterative_Solver #Direct_Solver

    EQ ENERGY for TEMPERATURE on block_1 using Q2 with MASS DIFF SRC
    EQ ENERGY for TEMPERATURE on block_2 using Q2 with MASS DIFF SRC

    IC const on block_1 temperature = 1.0
    IC const on block_2 temperature = 1.0

    # surface_4: x=0
    # surface_6: x=1

    # surface_3: y=0
    # surface_5: y=1

    # surface_1: z=1
    # surface_2: z=0

    # const Temp BC (x)
    BC const dirichlet at surface_4 Temperature = 1.0
    BC const dirichlet at surface_6 Temperature = 1.0

    # const flux BC (y)
```

```
  BC FLUX for Energy on surface_3  = constant flux = 3
  BC FLUX for Energy on surface_3  = scalar_string_function f="exact_flux_y - 3"

  BC FLUX for Energy on surface_5  = constant flux = 5
  BC FLUX for Energy on surface_5  = scalar_string_function f="-exact_flux_y - 5"

  # convective flux BC with const Temp and H (z)
  BC Flux for Energy on surface_1 = Nat_Conv T_Ref={T_ref_s1} H={h_s1}
  BC FLUX for Energy on surface_1  = scalar_string_Function \$
    f="-exact_flux_z - {h_s1}*(exact_sln - {T_ref_s1})"

  BC Flux for Energy on surface_2 = Nat_Conv T_Ref={T_ref_s2} H={h_s2}
  BC FLUX for Energy on surface_2  = scalar_string_Function \$
    f="exact_flux_z - {h_s2}*(exact_sln - {T_ref_s2})"

  #  const source term
  Source For ENERGY on block_1 = Constant value=1
  Source For ENERGY on block_1 = scalar_string_Function f="exact_src-1"

  Source For ENERGY on block_2 = Constant value=1
  Source For ENERGY on block_2 = scalar_string_Function f="exact_src-1"

  begin contact definition mpc1
    contact surface surf_1 contains surface_7
    contact surface surf_2 contains surface_8

    begin interaction inter_1
      surfaces = surf_1 surf_2
    end
    begin enforcement enf_1
      Enforcement for Energy = Tied_Temperature
    end
  end

  postprocess max of function "abs(exact_sln - temperature)" on all_blocks as linf_err
  postprocess l_2_norm of function "exact_sln - temperature" on all_blocks as l2_err
  postprocess l_2_norm of function "exact_sln_dot - dt_temperature" on all_blocks as l2_dot_err
  postprocess integral of function "\$
    (gradExact_sln_x - grad_temperature_x)^2 + \$
    (gradExact_sln_y - grad_temperature_y)^2 + \$
    (gradExact_sln_z - grad_temperature_z)^2" on all_blocks as H1_err_sq
  postprocess global_function "sqrt(H1_err_sq)" as H1_err

  Begin heartbeat testctrl
    Stream name = errors_h{N}.dat
    Timestamp Format    = ""
    Precision           = 8
    Format = csv
    Labels = Off
    Legend = On
    At step 0, Increment is 1
    variable = global time
    variable = global number_of_nodes
    variable = global l2_err
    variable = global l2_dot_err
    variable = global H1_err
    variable = global linf_err
  end

  BEGIN RESULTS OUTPUT LABEL diffusion output
    database Name = thermal_transient_tet10_tied_contact_h{N}.e
    at step 0, increment = {2**N}
    title Aria cube test
    nodal variables = solution->TEMPERATURE as T
    nodal variables = time_derivative_at_time->TEMPERATURE as TDOT
  END RESULTS OUTPUT LABEL diffusion output

END ARIA REGION myRegion
```

```
  END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.4. ELEMENT DEATH

12.4.1. CDFEM Element Death (Heat Flux)

12.4.1.1. Tri3

```
BEGIN SIERRA Aria

Title \$
1-d standard conduction problem, Carslaw and Jaeger P. 292\$

  BEGIN TPETRA EQUATION SOLVER  SOLVE_TEMPERATURE
    BEGIN CG SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 10000
      RESIDUAL SCALING = R0
      CONVERGENCE TOLERANCE = 1.000000e-14
    END
  END TPETRA EQUATION SOLVER


Begin Global Constants
     Stefan Boltzmann Constant = 5.67e-8  # W/m^2-K^4
     Ideal Gas Constant = 8.314           # J/mol-K
End

BEGIN ARIA MATERIAL solid
  DENSITY = constant rho = 1.
  Thermal Conductivity = constant k = 1.
  Specific Heat = Constant cp = 1.0
  Heat Conduction             = basic
END ARIA MATERIAL solid

BEGIN FINITE ELEMENT MODEL VERIFY_DEATH
  DATABASE NAME = input{N}_tri3.e
  decomposition method = rcb
  COORDINATE SYSTEM = CARTESIAN
  DATABASE TYPE = EXODUSII
  USE MATERIAL solid FOR block_1
  USE MATERIAL solid FOR block_1_dead
END FINITE ELEMENT MODEL VERIFY_DEATH

Begin Universal Aria Expressions
   User Expression = Scalar_String_Function F = "ln(sqrt(x*x+y*y))*(1/ln(2-t))" User_Tag = exact_soln
   User Expression = Scalar_String_Function F = "sqrt(x*x+y*y)" User_Tag = radius
   User Expression = Scalar_String_Function F = "2-t" User_Tag = exact_interface
   User Expression = Scalar_String_Function F = "ln(sqrt(x*x+y*y))*(-1/(ln(2-t)*ln(2-t)))*(1/(2-t))*(-1)" User_Tag = exact_src
   User Expression = Vector_String_Function F_X = "(x/(x*x+y*y))*(1/ln(2-t))" F_Y = "(y/(x*x+y*y))*(1/ln(2-t))" User_Tag = exact
   User Expression = Scalar_String_Function F = "-1/((2-t)*ln(2-t))" User_Tag = exact_flux
End
Begin Procedure myProcedure

  Begin Solution Control Description
     Use System Main
     Begin System Main
       Simulation Start Time            = 0.0
       Simulation Termination Time      = 0.9
       Begin transient MySolveBlock
           Advance myRegion
        End
     End
      begin parameters for transient MySolveBlock
        start time = 0.0
        begin parameters for aria region myRegion
          initial time step size = {0.1*(0.5**(N-1))}
```

```
        Predictor-Corrector Tolerance = {0.05*(0.5**(N-1))}
        Maximum Time Step Size = {0.2*(0.5**(N-1))}
        Time Integration Method = bdf2
        time step variation = adaptive
      end
  end
End

Begin Aria Region myRegion

  Begin Heartbeat pp_out
    Legend = on
    Labels = off
    Format = csv
    At Step 1 Interval is 1
    Precision = 6
    Stream Name = errors{N}.dat
    variable is Global time
    variable is Global number_of_nodes as num_nodes
    Global l2 as l2
    Global h1 as h1
    Global linf as linf
    Global l2_interface as l2_interface
    Global linf_interface as linf_interface
  End
  Use Finite Element Model VERIFY_DEATH

  Begin CDFEM Death death_by_temp
    add volume block_1
    Criterion is solution->Temperature > 1.0
  End

  Use Linear Solver solve_temperature

  nonlinear solution strategy = newton
  maximum nonlinear iterations = 2
  nonlinear correction tolerance = 1.0e-12
  nonlinear residual tolerance = 1.0e-12
  nonlinear relaxation factor  = 1.0

  use dof averaged nonlinear residual

  eq energy for temperature on all_blocks using q1 with lumped_mass diff src

  IC for temperature on block_1 = Scalar_String_Function F = "exact_soln"

  BC Dirichlet for Temperature on surface_2 = Scalar_String_Function F = "exact_soln"

  SOURCE for Energy on block_1 = Scalar_String_Function F = "exact_src"

  BC Flux for Energy on surface_1 = Scalar_String_Function F = "exact_flux"
  BC Flux for Energy on surface_block_1_death_by_temp = Scalar_String_Function F = "exact_flux"

  postprocess l_2_norm of scalar_string_function f = "exact_soln - (temperature)" on block_1 as l2
  postprocess l_2_norm of scalar_string_function f = "sqrt((exact_soln_grad_x - (grad_temperature_x))^2+(exact_soln_grad_y - (
  postprocess l_inf_norm of scalar_string_function f = "exact_soln - (temperature)" on block_1 as linf

  Begin Postprocess l_2_norm
    Location = surface_1
    Location = surface_block_1_death_by_temp
    Output Name = l2_interface
    Source Type = Scalar_String_Function F = "exact_interface - (radius)"
  End
  Begin Postprocess l_inf_norm
    Location = surface_1
    Location = surface_block_1_death_by_temp
    Output Name = linf_interface
    Source Type = Scalar_String_Function F = "exact_interface - (radius)"
```

```
    End

    Begin Results Output Label diffusion output
      database name = output{N}.e
      at Step 0, Interval = {2**N}
      Nodal Variables = solution->temperature as TEMP
      Global Variables = linf_interface l2_err h1_err linf_err
    End Results Output Label diffusion output

  End Aria Region myRegion

End Procedure myProcedure

END SIERRA ARIA
```

12.4.1.2. Tet4

```
BEGIN SIERRA Aria

Title \$
1-d standard conduction problem, Carslaw and Jaeger P. 292\$

  BEGIN TPETRA EQUATION SOLVER  SOLVE_TEMPERATURE
    BEGIN CG SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 10000
      RESIDUAL SCALING = R0
      CONVERGENCE TOLERANCE = 1.000000e-14
    END
  END TPETRA EQUATION SOLVER


Begin Global Constants
     Stefan Boltzmann Constant = 5.67e-8  # W/m^2-K^4
     Ideal Gas Constant = 8.314           # J/mol-K
End

BEGIN ARIA MATERIAL solid
  DENSITY = constant rho = 1.
  Thermal Conductivity = constant k = 1.
  Specific Heat = Constant cp = 1.0
  Heat Conduction             = basic
END ARIA MATERIAL solid

BEGIN FINITE ELEMENT MODEL VERIFY_DEATH
  DATABASE NAME = input{N}_tet4.e
  decomposition method = rcb
  COORDINATE SYSTEM = CARTESIAN
  DATABASE TYPE = EXODUSII
  USE MATERIAL solid FOR block_1
  USE MATERIAL solid FOR block_1_dead
END FINITE ELEMENT MODEL VERIFY_DEATH

Begin Universal Aria Expressions
   User Expression = Scalar_String_Function F = "(1+t)/sqrt(x*x+y*y+z*z)" User_Tag = exact_soln
   User Expression = Vector_String_Function F_X = "(1+t)*(-1/(x*x+y*y+z*z))*(x/sqrt(x*x+y*y+z*z))" F_Y = "(1+t)*(-1/(x*x+y*y+z*z
   User Expression = Scalar_String_Function F = "sqrt(x*x+y*y+z*z)" User_Tag = radius
   User Expression = Scalar_String_Function F = "1+t" User_Tag = exact_interface
   User Expression = Scalar_String_Function F = "1/sqrt(x*x+y*y+z*z)" User_Tag = exact_src
   User Expression = Scalar_String_Function F = "-1/(1+t)" User_Tag = exact_flux
End
Begin Procedure myProcedure

  Begin Solution Control Description
     Use System Main
```

```
   Begin System Main
     Simulation Start Time            = 0.0
     Simulation Termination Time      = 0.75
     Begin transient MySolveBlock
         Advance myRegion
      End
   End
    begin parameters for transient MySolveBlock
      start time = 0.0
      begin parameters for aria region myRegion
        initial time step size = {0.1*(0.5**(N-1))}
        Predictor-Corrector Tolerance = {0.05*(0.5**(N-1))}
        Maximum Time Step Size = {0.2*(0.5**(N-1))}
        Time Integration Method = bdf2
        time step variation = adaptive
      end
  end
End

Begin Aria Region myRegion

  Begin Heartbeat pp_out
    Legend = on
    Labels = off
    Format = csv
    At Step 1 Interval is 1
    Precision = 6
    Stream Name = errors{N}.dat
    variable is Global time
    variable is Global number_of_nodes as num_nodes
    Global l2 as l2
    Global h1 as h1
    Global linf as linf
    Global l2_interface as l2_interface
    Global linf_interface as linf_interface
  End
  Use Finite Element Model VERIFY_DEATH

  Begin CDFEM Death death_by_temp
    add volume block_1
    Criterion is solution->Temperature > 1.0
  End

  Use Linear Solver solve_temperature

  nonlinear solution strategy = newton
  maximum nonlinear iterations = 2
  nonlinear correction tolerance = 1.0e-12
  nonlinear residual tolerance = 1.0e-12
  nonlinear relaxation factor  = 1.0

  use dof averaged nonlinear residual

  eq energy for temperature on all_blocks using q1 with lumped_mass diff src

  IC for temperature on block_1 = Scalar_String_Function F = "exact_soln"

  BC Dirichlet for Temperature on surface_2 = Scalar_String_Function F = "exact_soln"

  SOURCE for Energy on block_1 = Scalar_String_Function F = "exact_src"

  BC Flux for Energy on surface_1 = Scalar_String_Function F = "exact_flux"
  BC Flux for Energy on surface_block_1_death_by_temp = Scalar_String_Function F = "exact_flux"

  postprocess l_2_norm of scalar_string_function f = "exact_soln - (temperature)" on block_1 as l2
  postprocess l_2_norm of scalar_string_function f = "sqrt((exact_soln_grad_x - (grad_temperature_x))^2+(exact_soln_grad_y - (
  postprocess l_inf_norm of scalar_string_function f = "exact_soln - (temperature)" on block_1 as linf
```

```
    Begin Postprocess l_2_norm
      Location = surface_1
      Location = surface_block_1_death_by_temp
      Output Name = l2_interface
      Source Type = Scalar_String_Function F = "exact_interface - (radius)"
    End
    Begin Postprocess l_inf_norm
      Location = surface_1
      Location = surface_block_1_death_by_temp
      Output Name = linf_interface
      Source Type = Scalar_String_Function F = "exact_interface - (radius)"
    End

    Begin Results Output Label diffusion output
      database name = output{N}.e
      at Step 0, Interval = {2**N}
      Nodal Variables = solution->temperature as TEMP
      Global Variables = linf_interface l2 h1 linf
    End Results Output Label diffusion output

  End Aria Region myRegion

End Procedure myProcedure

END SIERRA ARIA
```

12.4.2. 3D Spherical Shell Enclosure

```
BEGIN SIERRA myJob

  #{include(r2CoFunctions.inc)}

  Title Element Death Test Problem #1

  BEGIN TPETRA EQUATION SOLVER  SOLVE_TEMPERATURE
    BEGIN CG SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      RESIDUAL SCALING = R0
      CONVERGENCE TOLERANCE = 1.000000e-09
    END
  END TPETRA EQUATION SOLVER


  BEGIN TPETRA EQUATION SOLVER  DIRECT_SOLVER
    BEGIN SUPERLU SOLVER
    END
  END TPETRA EQUATION SOLVER


  BEGIN GLOBAL Constants
    Stefan Boltzmann Constant = 5.6704e-08  # W/m2-K4
  End

  #{is_inner = 0}
  {loop(2)}
    BEGIN ARIA MATERIAL ss304_{is_inner}
      Heat Conduction      = Basic
      Density              = Constant rho =  7862.0   $ kg/m**3
      Specific Heat        = Constant  Cp = 10.0    $ J/gm/K
      Thermal Conductivity = Constant   K = 1.0     $ W/m/K

      {if(is_inner)}
        user expression = scalar_string_function user_tag = "T1_curr" \$
          f = "{T1} + {dT1}*(-cos(pi*t)/2.0 + 1.0/2.0)"
```

```
      user expression = scalar_string_function user_tag = "T1_curr_dot" \$
        f = "pi*{dT1}*sin(pi*t)/2.0"

      user expression = scalar_string_function user_tag = "dTi_dT1" \$
        f = "-(-1/{r1} + 1.0/radius)/(1.0/r2 - 1/{r1}) + 1"

      user expression = scalar_string_function user_tag = "dTi_dr2" \$
        f = "(-T1_curr + {Tc})*(-1/{r1} + 1.0/radius)/(pow(r2, 2)*pow(1.0/r2 - 1.0/{r1}, 2))"

      user expression = scalar_string_function user_tag = "exact_soln" \$
        f = "T1_curr + (1.0/radius - 1/{r1})*({Tc} - T1_curr)/(1.0/r2 - 1/{r1})"

      user expression = scalar_string_function user_tag = "exact_soln_dot" \$
        f = "dTi_dT1*T1_curr_dot + dTi_dr2*r2_dot"

      user expression = scalar_string_function user_tag = "dTemp_dr" \$
        f = "-(T1_curr - {Tc})/(radius*radius*(1/{r1} - 1/r2))"
    {else}
      user expression = scalar_string_function user_tag = "dTo_dCo" \$
        f = "-(-1.0/{r3} + 1.0/radius)/(1.0/{r4} - 1/{r3}) + 1.0"

      user expression = scalar_string_function user_tag = "exact_soln" \$
        f = "Co + ({T4} - Co)*(1.0/radius - 1.0/{r3})/(1.0/{r4} - 1/{r3})"

      user expression = scalar_string_function user_tag = "exact_soln_dot" \$
        f = "dTo_dCo*Co_dot"

      user expression = scalar_string_function user_tag = "dTemp_dr" \$
        f = "-(Co - {T4})/(radius*radius*(1.0/{r3} - 1.0/{r4}))"
    {endif}
  END ARIA MATERIAL ss304_{is_inner}
  #{is_inner = 1}
{endloop}

BEGIN ARIA MATERIAL fake
END ARIA MATERIAL fake

BEGIN FINITE ELEMENT MODEL myModel
  Database Name = two_sphere_shells_tet4_m{N}.g
  Coordinate System is cartesian
  decomposition method = rcb

  Use material ss304_0 for block_1 # Outer "case" block
  Use material ss304_1 for block_2 # Inner block that will have death

  #THIS BLOCK SHOULD BE REMOVED EVENTUALLY BUT IS CURRENTLY REQUIRED
  # Commenting it out leads to a segfault because of a null field data pointer for model_coordinates
  # This is very bizarre
  Use material fake for block_2_dead
  #BEGIN PARAMETERS FOR BLOCK block_2_dead
  #  Material fake
  #END PARAMETERS FOR BLOCK block_2_dead

END FINITE ELEMENT MODEL myModel

begin universal aria expressions
  user expression = user_function user_tag = "Co" name=Co_function x=time
  user expression = user_function user_tag = "r2" name=r2_function x=time
  user expression = user_function user_tag = "Co_dot" name=Co_dot_function x=time
  user expression = user_function user_tag = "r2_dot" name=r2_dot_function x=time

  user expression = scalar_string_function f="sqrt(x*x + y*y + z*z)" user_tag = "radius"

  user expression = vector_string_function \$
    f_x="x/radius" \$
    f_y="y/radius" \$
    f_z="z/radius" \$
```

```
    user_tag = "gradRadius"

  user expression = vector_string_function \$
    f_x="dTemp_dr*gradRadius_x" \$
    f_y="dTemp_dr*gradRadius_y" \$
    f_z="dTemp_dr*gradRadius_z" \$
    user_tag = "gradExact_soln"

  user expression = scalar_string_function user_tag = "exact_src" \$
    f = "{rho}*{cp}*exact_soln_dot"
end universal aria expressions

BEGIN PROCEDURE myAriaProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Simulation Start Time           = {tStart}
      Simulation Termination Time     = {tEnd}
      Begin Transient Stepper
        Advance myRegion
      End
    End

    Begin Parameters For Transient Stepper
      Begin Parameters for Aria Region myRegion
        Time Integration Method = BDF2
        Time Step Variation     = fixed #Adaptive
        Initial Time Step Size  = {0.05*0.5**(N)}
        #Minimum Time Step Size  = 1.0e-5
        #Maximum Time Step Size  = 1000.0
        #predictor order = 0
        #Predictor-Corrector Tolerance = 1.e-3
        #Fail Time Step When Time Step Size Ratio Is Below 0.0
      End
    End
  End

  BEGIN ARIA REGION myRegion

    Use Linear Solver solve_temperature #direct_solver
    Use Finite Element Model myModel

    Begin CDFEM Death death_temp1
      add volume block_2
      Criterion is solution->Temperature > 867.011674920813
    End

    Nonlinear Solution Strategy  = newton
    Maximum Nonlinear Iterations = 10
    Nonlinear Residual Tolerance = 1.0e-06
    Nonlinear Correction Tolerance = 1.0e-06
    Nonlinear Relaxation Factor  = 1.0
    Accept Solution After Maximum Nonlinear Iterations = false

    IC for Temperature for all_volumes = scalar_string_Function f="exact_soln"

    BC Dirichlet for Temperature at surface_1 = scalar_string_Function f="exact_soln"
    BC Dirichlet for Temperature at surface_4 = scalar_string_Function f="exact_soln"

    EQ ENERGY for TEMPERATURE on block_1 using Q1 with lumped_mass Diff Src
    EQ ENERGY for TEMPERATURE on block_2 using Q1 with lumped_mass Diff Src

    Source For ENERGY on all_volumes = scalar_string_Function f="exact_src"

    Begin Viewfactor Calculation vf_calc
      Compute Rule           = Hemicube
      Geometric Tolerance    = 1.0e-6
```

```
  Hemicube Resolution     = 500
  Hemicube Max Subdivides = 5
  hemicube min separation     = 5.0
  Output Rule             = Verbose
End

Begin Viewfactor Smoothing smooth
  Method                  = least-squares
  Convergence Tolerance   = 1.0e-06
  Maximum Iterations      = 500
  weight power            = 2.0
  Reciprocity Rule        = average
  Output Rule             = verbose
End

Begin Viewfactor Smoothing no_smooth
  Method                = none
  Convergence Tolerance = 1.0E-08
  Maximum Iterations    = 150
  Weight Power          = 2.0
  Reciprocity Rule      = average
  Output Rule           = Summary
End

Begin Radiosity Solver Rad_Solv
  Coupling                = mason
  Solver                  = chaparral GMRES
  Convergence Tolerance   = 1.0e-08
  Maximum Iterations      = 300
  Output Rule             = none
End

Begin Enclosure Definition enc1
  add surface surface_2
  add surface surface_3
  add surface surface_block_2_death_temp1

  meshed enclosure is block_2_dead
  disable parallel redistribution

  Emissivity = 0.6 On surface_block_2_death_temp1
  Emissivity = 0.6 On surface_2
  Emissivity = 0.7 On surface_3

  Blocking Surfaces
  Use Viewfactor Calculation vf_calc
  Use Viewfactor Smoothing no_smooth
  Use Radiosity Solver Rad_Solv
End

Postprocess Heat_Flux on all_blocks

postprocess average of expression temperature on each_surface as avg_T

postprocess l_2_norm of function "exact_soln - temperature" on all_blocks as l2_err
postprocess l_2_norm of function "exact_soln_dot - dt_temperature" on all_blocks as l2_dot_err
postprocess l_inf_norm of function "exact_soln - TEMPERATURE" on all_blocks as linf_err
postprocess integral of function "\$
  (gradExact_soln_x - grad_temperature_x)^2 + \$
  (gradExact_soln_y - grad_temperature_y)^2 + \$
  (gradExact_soln_z - grad_temperature_z)^2" on all_blocks as H1_err_sq
postprocess global_function "sqrt(H1_err_sq)" as H1_err

postprocess average of expression r2 on block_2 as r2_exact
postprocess max of expression radius on block_2 as r2_num
POSTPROCESS GLOBAL_FUNCTION "r2_exact - r2_num" as r2_err

Begin heartbeat testctrl
```

```
      Stream name = errors{N}.dat
      Timestamp Format    = ""
      Precision           = 8
      Format = csv
      Labels = Off
      Legend = On
      At step 0, Increment is 1
      variable = global time
      variable = global number_of_nodes
      variable = global l2_err
      variable = global l2_dot_err
      variable = global linf_err
      variable = global h1_err
    end

    postprocess value of expression exact_soln on all_blocks as exact_temp
    postprocess value of expression exact_src on all_blocks as exact_temp_src

    BEGIN RESULTS OUTPUT myLABEL diffusion output etc
      Database Name = cdfem_rad_death_m{N}.e
      at step 0, increment = 1

      Title CDFEM Death Test Case #1
      Nodal Variables = Solution->Temperature as T
      Nodal Variables = pp->heat_flux as Heat_Flux
      Nodal Variables = exact_temp
      # Nodal Variables = exact_temp_err
      Nodal Variables = exact_temp_src
      Nodal Variables = is_inside
      Nodal Variables = encl_position linf_err
      nodal variables = time_derivative_at_time->TEMPERATURE as TDOT
      Face Variables = face_area emissivity face_coverage face_temperature irradiance rad_flux radiosity
      Element Variables = current_element_volume initial_element_volume volume_change_ratio l2_err l2_dot_err h1_err
    END

    Begin Heartbeat thermalrace
      stream Name = globals_tet4_m{N}.dat
      precision = 8
      Format = csv
      timestamp format ''
      legend = on
      labels = off
      Variable = Global time as time
      Variable = Global l2_err
      Variable = Global l2_dot
      Variable = Global linf
      Variable = Global h1_err
      Variable = Global enc1_area
      Variable = Global enc1_volume
      Variable = Global avg_T_surface_1
      Variable = Global avg_T_surface_2 # this value is mostly NaNs
      Variable = Global avg_T_surface_block_2_death_temp1
      Variable = Global avg_T_surface_3
      Variable = Global avg_T_surface_4
      Variable = Global r2_exact
      Variable = Global r2_num
      Variable = Global r2_err
      at step 0, increment = 1
    end

  END ARIA REGION myRegion

 END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.5. TIME INTEGRATION

12.5.1. Adaptive Time Integration

12.5.1.1. First Order Fixed

12.5.1.2. First Order Adaptive

12.5.1.3. Second Order Fixed

12.5.1.4. Second Order Adaptive

12.5.1.5. BDF2 Fixed

12.5.1.6. BDF2 Adaptive

12.6. ENCLOSURE RADIATION

12.6.1. 2D Cylindrical Shell Enclosure

```
# T1 = { T1 = 300.0 }
# T2 = { T2 = 444.7977 }
# T3 = { T3 = 956.5915 }
# T4 = { T4 = 1300.0 }
# r1 = { r1 = 0.01 }
# r2 = { r2 = 0.02 }
# r3 = { r3 = 0.03 }
# r4 = { r4 = 0.04 }
# rMidGap = { rMidGap = 0.5*(r2+r3) }
# c_in = { c_in = (T2-T1)/log(r2/r1) }
# c_out = { c_out = (T4-T3)/log(r4/r3) }

BEGIN SIERRA Aria

  Title Verification for two concentric spheres with radiation gap between them

  Begin Global Constants
    Stefan Boltzmann constant = 5.6704E-8
  End Global Constants

  Begin Aria Material inner
    Heat conduction = Basic
    Density = constant rho = 1.0
    Specific heat = constant cp = 1.0
    Thermal conductivity = constant k = 2.0
  End Aria Material inner

  Begin Aria Material outer
    Heat conduction = Basic
    Density = constant rho = 1.0
    Specific heat = constant cp = 1.0
    Thermal conductivity = constant k = 0.35
  End Aria Material outer

  begin universal aria expressions
    user expression = scalar_string_function f="sqrt(x*x + y*y)" user_tag = "radius"

    user expression = vector_string_function \$
      f_x="x/radius" \$
      f_y="y/radius" \$
      user_tag = "gradRadius"

    user expression = scalar_string_function f = "( (radius<={rMidGap}) ? \$
      ({T1} +  {c_in}*log(radius/{r1})) : \$
      ({T4} + {c_out}*log(radius/{r4})) )" \$
      user_tag = "exact_soln"

    user expression = scalar_string_function f = "( (radius<={rMidGap}) ? \$
      ( {c_in}/radius) : \$
      ({c_out}/radius) )" \$
      user_tag = "dTemp_dr"

    user expression = vector_string_function \$
      f_x="dTemp_dr*gradRadius_x" \$
      f_y="dTemp_dr*gradRadius_y" \$
      user_tag = "gradExact_soln"

  end universal aria expressions

  Begin Finite Element Model VERIFY_RAD_GAP
    Database name = input{N}.g
    Coordinate System = Cartesian
```

```
  Database Type = EXODUSII
  Begin Parameters for Block block_1
    Material inner
  End
  Begin Parameters for Block block_2
    Material outer
  End
End Finite Element Model VERIFY_RAD_GAP

BEGIN TPETRA EQUATION SOLVER  SOLVE_TEMPERATURE
  BEGIN CG SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = RHS
    CONVERGENCE TOLERANCE = 1.000000e-14
  END
END TPETRA EQUATION SOLVER

Begin Procedure myProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Begin Sequential Steady
        Advance myRegion
      End
    End
  End

  Begin Aria Region myRegion

    Use Finite Element Model VERIFY_RAD_GAP
    Use Linear Solver solve_temperature

    Nonlinear Solution Strategy = Newton
    maximum nonlinear iterations = 1000
    nonlinear residual tolerance = 1.0e-10
    nonlinear relaxation factor = 1.0

    EQ energy for Temperature for all_volumes using Q1 with Diff
    IC const for all_volumes Temperature = 300.0
    BC const Dirichlet at surface_1 Temperature = 300.0
    BC const Dirichlet at surface_4 Temperature = 1300.0

    begin enclosure definition sph_shell
      add surface surface_2
      add surface surface_3
      blocking surfaces
      use viewfactor calculation vf_calc
      use viewfactor smoothing vf_smooth
      use radiosity solver rad_solver
      #      enclosure id           = 1
      emissivity = 0.50 on surface_2
      emissivity = 0.80 on surface_3
    end enclosure definition sph_shell

    begin viewfactor calculation vf_calc
      bsp tree max depth = 0 and min list length = 25
      compute rule               = hemicube
      geometric tolerance        = 1.0E-6
      hemicube max subdivides     = 5
      hemicube min separation     = 5.0
      hemicube resolution         = 500
      #      check rowsum with tolerance = .001
      output rule                = verbose
    end viewfactor calculation vf_calc
```

```
      begin viewfactor smoothing vf_smooth
        convergence tolerance = 1.0E-10
        method                = least-squares
        weight power          = 2
        maximum iterations    = 150
        reciprocity rule      = average
        output rule           = verbose
      end viewfactor smoothing vf_smooth

      begin radiosity solver rad_solver
        coupling              = mason
        solver                = chaparral gmres
        convergence tolerance = 1.0E-8
        maximum iterations    = 800
        output rule           = verbose
      end radiosity solver rad_solver

      postprocess l_2_norm of function "exact_soln - TEMPERATURE" on all_blocks as l2_err
      postprocess l_inf_norm of function "exact_soln - TEMPERATURE" on all_blocks as lInf_err
      postprocess integral of function "\$
        (gradExact_soln_x - grad_temperature_x)^2 + \$
        (gradExact_soln_y - grad_temperature_y)^2" on all_blocks as H1_err_sq
      postprocess global_function "sqrt(H1_err_sq)" as H1_err

      Begin heartbeat testctrl
        Stream name = errors{N}.dat
        Timestamp Format    = ""
        Precision           = 8
        Format = csv
        Labels = Off
        Legend = On
        At step 0, Increment is 1
        variable = global time
        variable = global number_of_nodes

        variable = global l2_err
        variable = global linf_err
        variable = global H1_err
      end

      Begin Results Output Label diffusion output
        database name = output{N}.e
        at Step 0, increment = 1
        Nodal Variables = solution->temperature as TEMP
      End Results Output Label diffusion output

    End Aria Region myRegion

  End Procedure myProcedure

END SIERRA Aria
```

12.6.2. 2D Annular Enclosure

12.6.3. 3D Spherical Shell Enclosure

12.6.4. 3D Spherical Shell Partial Enclosure

```
# T1 = { T1 = 300.0 }
# T2 = { T2 = 578.225 }
# T3 = { T3 = 1035.02 }
# T4 = { T4 = 1300.0 }
# r1 = { r1 = 0.01 }
```

```
# r2 = { r2 = 0.02 }
# r3 = { r3 = 0.03 }
# r4 = { r4 = 0.04 }
# rMidGap = { rMidGap = 0.5*(r2+r3) }


BEGIN SIERRA Aria

  Title Verification for two concentric spheres with radiation gap between them

  Begin Global Constants
    Stefan Boltzmann constant = 5.6704E-8
  End Global Constants

  begin definition for function emissivity_table
    scale by 1.0
    type = piecewise linear
    begin values
      0.0   0.5
      5000  0.5
    end values
  end definition for function emissivity_table

  Begin Aria Material inner
    Heat conduction = Basic
    Density = constant rho = 1.0
    Specific heat = constant cp = 1.0
    Thermal conductivity = constant k = 2.0
    emissivity = user_function name=emissivity_table X=temperature
  End Aria Material inner

  Begin Aria Material outer
    Heat conduction = Basic
    Density = constant rho = 1.0
    Specific heat = constant cp = 1.0
    Thermal conductivity = constant k = 0.35
    emissivity = constant e = 0.80
  End Aria Material outer

  begin universal aria expressions
    user expression = scalar_string_function f="sqrt(x*x + y*y + z*z)" user_tag = "radius"

    user expression = vector_string_function \$
      f_x="x/radius" \$
      f_y="y/radius" \$
      f_z="z/radius" \$
      user_tag = "gradRadius"

    user expression = scalar_string_function f = "( (radius<={rMidGap}) ? \$
      ({T1} + ({T2} - {T1}) * (1/radius - 1/{r1}) / (1/{r2} - 1/{r1})) : \$
      ({T4} + ({T3} - {T4}) * (1/radius - 1/{r4}) / (1/{r3} - 1/{r4})) )" \$
      user_tag = "exact_soln"

    user expression = scalar_string_function f = "( (radius<={rMidGap}) ? \$
      (({T2} - {T1}) * (-1/(radius*radius)) / (1/{r2} - 1/{r1})) : \$
      (({T3} - {T4}) * (-1/(radius*radius)) / (1/{r3} - 1/{r4})) )" \$
      user_tag = "dTemp_dr"

    user expression = vector_string_function \$
      f_x="dTemp_dr*gradRadius_x" \$
      f_y="dTemp_dr*gradRadius_y" \$
      f_z="dTemp_dr*gradRadius_z" \$
      user_tag = "gradExact_soln"

  end universal aria expressions

  Begin Finite Element Model VERIFY_RAD_GAP
    Database name = sphere_cutout_h{N}.g
```

```
  Coordinate System = Cartesian
  Database Type = EXODUSII
  Begin Parameters for Block block_1
    Material inner
  End
  Begin Parameters for Block block_2
    Material outer
  End
End Finite Element Model VERIFY_RAD_GAP

BEGIN TPETRA EQUATION SOLVER  SOLVE_TEMPERATURE
  BEGIN CG SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = RHS
    CONVERGENCE TOLERANCE = 1.000000e-10
  END
END TPETRA EQUATION SOLVER


Begin Procedure myProcedure

  Begin Solution Control Description
    Use System Main
    Begin System Main
      Begin Sequential Steady
        Advance myRegion
      End
    End
  End

  Begin Aria Region myRegion

    Use Finite Element Model VERIFY_RAD_GAP
    Use Linear Solver solve_temperature

    Nonlinear Solution Strategy = Newton
    maximum nonlinear iterations = 1000
    nonlinear residual tolerance = 1.0e-8
    nonlinear correction tolerance = 1.0e-8
    nonlinear relaxation factor = 1.0

    EQ energy for Temperature for all_volumes using Q1 with Diff
    BC const Dirichlet at surface_1 Temperature = {T1}
    BC const Dirichlet at surface_4 Temperature = {T4}
    BC Dirichlet for Temperature on surface_5 = scalar_string_function f="exact_soln"
    IC for Temperature on all_blocks = scalar_string_function f="exact_soln"

    begin enclosure definition sph_shell
      add surface surface_2
      add surface surface_3
      blocking surfaces
      use viewfactor calculation vf_calc
      use viewfactor smoothing vf_smooth
      use radiosity solver rad_solver
      emissivity = 0.50 on surface_2
      emissivity = 0.80 on surface_3
      Partial Enclosure Emissivity = 0.8
      Partial Enclosure Area = {2.0*PI*0.03*(0.03-0.025)}
      Partial Enclosure Temperature = {T3}
    end enclosure definition sph_shell

    begin viewfactor calculation vf_calc
      bsp tree max depth = 0 and min list length = 25
      compute rule                = hemicube
      geometric tolerance         = 1.0E-10
      hemicube max subdivides     = 5
```

```
      hemicube min separation     = 5.0
      hemicube resolution         = 500
      #      check rowsum with tolerance = .001
      output rule                 = verbose
    end viewfactor calculation vf_calc

    begin viewfactor smoothing vf_smooth
      method                = none
    end viewfactor smoothing vf_smooth

    begin radiosity solver rad_solver
      coupling              = mason
      solver                = chaparral gmres
      convergence tolerance = 1.0E-9
      maximum iterations    = 80
      output rule           = summary
    end radiosity solver rad_solver

    postprocess l_2_norm of function "exact_soln - TEMPERATURE" on all_blocks as l2_err
    postprocess l_inf_norm of function "exact_soln - TEMPERATURE" on all_blocks as lInf_err
    postprocess integral of function "\$
      (gradExact_soln_x - grad_temperature_x)^2 + \$
      (gradExact_soln_y - grad_temperature_y)^2 + \$
      (gradExact_soln_z - grad_temperature_z)^2" on all_blocks as H1_err_sq
    postprocess global_function "sqrt(H1_err_sq)" as H1_err

    postprocess value of expression exact_soln on all_blocks as analytic_temp
    postprocess Value of function "exact_soln - temperature" on all_blocks as temp_error

    Begin heartbeat testctrl
      Stream name = errors{N}.dat
      Timestamp Format    = ""
      Precision           = 8
      Format = csv
      Labels = Off
      Legend = On
      At step 0, Increment is 1
      variable = global time
      variable = global number_of_nodes

      variable = global l2_err
      variable = global linf_err
      variable = global H1_err
    end

    Begin Results Output Label diffusion output
      database name = output{N}.e
      at Step 0, increment = 1
      Nodal Variables = solution->temperature as T
      nodal variables = analytic_temp
      nodal variables = temp_error
      element variables = h1_err l2_err linf_err
    End Results Output Label diffusion output

  End Aria Region myRegion

 End Procedure myProcedure

END SIERRA Aria
```

12.6.5. Fully 2D Enclosure Radiation

12.7. CHEMISTRY

12.7.1. First Order Reaction (Uniform Temperature)

12.7.2. First Order Reaction (Spatially Varying Temperature)

12.7.3. First Order Reaction

12.7.4. DAE and Pressure Test

```
BEGIN SIERRA Aria

  Title Verification Problem for Coupled Chemistry Diffusion

  begin definition for function A_exact
    type = piecewise linear
    begin values
0  1
0.00500000000000000  0.994785408602410
0.0100000000000000  0.990262801895543
0.0150000000000000  0.985996477466782
0.0200000000000000  0.981896043746815
0.0250000000000000  0.977918836400346
0.0300000000000000  0.974039821314594
0.0350000000000000  0.970242500122094
0.0400000000000000  0.966515169579351
0.0450000000000000  0.962849092341749
0.0500000000000000  0.959237494595795
0.0550000000000000  0.955674970187395
0.0600000000000000  0.952157103941654
0.0650000000000000  0.948680221790434
0.0700000000000000  0.945241218451135
0.0750000000000000  0.941837434719033
0.0800000000000000  0.938466567707596
0.0850000000000000  0.935126603669111
0.0900000000000000  0.931815766712535
0.0950000000000000  0.928532478977252
0.100000000000000  0.925275329232036
0.105000000000000  0.922043047782560
0.110000000000000  0.918834486178550
0.115000000000000  0.915648600625087
0.120000000000000  0.912484438289564
0.125000000000000  0.909341125898756
0.130000000000000  0.906217860166351
0.135000000000000  0.903113899697718
0.140000000000000  0.900028558097496
0.145000000000000  0.896961198064531
0.150000000000000  0.893911226303453
0.155000000000000  0.890878089116401
0.160000000000000  0.887861268564900
0.165000000000000  0.884860279112581
0.170000000000000  0.881874664675716
0.175000000000000  0.878903996021445
0.180000000000000  0.875947868463928
0.185000000000000  0.873005899816941
0.190000000000000  0.870077728568197
0.195000000000000  0.867163012246165
0.200000000000000  0.864261425954650
0.205000000000000  0.861372661054132
0.210000000000000  0.858496423971940
0.215000000000000  0.855632435125884
0.220000000000000  0.852780427948145
```

0.225000000000000	0.849940147998007
0.230000000000000	0.847111352153539
0.235000000000000	0.844293807873648
0.240000000000000	0.841487292522989
0.245000000000000	0.838691592753186
0.250000000000000	0.835906503934613
0.255000000000000	0.833131829633665
0.260000000000000	0.830367381131057
0.265000000000000	0.827612976977205
0.270000000000000	0.824868442581183
0.275000000000000	0.822133609830147
0.280000000000000	0.819408316736447
0.285000000000000	0.816692407109965
0.290000000000000	0.813985730253449
0.295000000000000	0.811288140678874
0.300000000000000	0.808599497843030
0.305000000000000	0.805919665900755
0.310000000000000	0.803248513474344
0.315000000000000	0.800585913437838
0.320000000000000	0.797931742715021
0.325000000000000	0.795285882090032
0.330000000000000	0.792648216029640
0.335000000000000	0.790018632516291
0.340000000000000	0.787397022891117
0.345000000000000	0.784783281706188
0.350000000000000	0.782177306585320
0.355000000000000	0.779578998092846
0.360000000000000	0.776988259609771
0.365000000000000	0.774404997216808
0.370000000000000	0.771829119583831
0.375000000000000	0.769260537865287
0.380000000000000	0.766699165601200
0.385000000000000	0.764144918623373
0.390000000000000	0.761597714966469
0.395000000000000	0.759057474783649
0.400000000000000	0.756524120266475
0.405000000000000	0.753997575568824
0.410000000000000	0.751477766734556
0.415000000000000	0.748964621628705
0.420000000000000	0.746458069871993
0.425000000000000	0.743958042778454
0.430000000000000	0.741464473295995
0.435000000000000	0.738977295949722
0.440000000000000	0.736496446787873
0.445000000000000	0.734021863330195
0.450000000000000	0.731553484518662
0.455000000000000	0.729091250670362
0.460000000000000	0.726635103432462
0.465000000000000	0.724184985739139
0.470000000000000	0.721740841770347
0.475000000000000	0.719302616912347
0.480000000000000	0.716870257719902
0.485000000000000	0.714443711880037
0.490000000000000	0.712022928177299
0.495000000000000	0.709607856460437
0.500000000000000	0.707198447610420
0.505000000000000	0.704794653509741
0.510000000000000	0.702396427012934
0.515000000000000	0.700003721918249
0.520000000000000	0.697616492940422
0.525000000000000	0.695234695684496
0.530000000000000	0.692858286620640
0.535000000000000	0.690487223059917
0.540000000000000	0.688121463130957
0.545000000000000	0.685760965757493
0.550000000000000	0.683405690636729
0.555000000000000	0.681055598218483
0.560000000000000	0.678710649685093

0.565000000000000	0.676370806932034
0.570000000000000	0.674036032549222
0.575000000000000	0.671706289802978
0.580000000000000	0.669381542618620
0.585000000000000	0.667061755563646
0.590000000000000	0.664746893831506
0.595000000000000	0.662436923225919
0.600000000000000	0.660131810145710
0.605000000000000	0.657831521570168
0.610000000000000	0.655536025044874
0.615000000000000	0.653245288668007
0.620000000000000	0.650959281077084
0.625000000000000	0.648677971436139
0.630000000000000	0.646401329423306
0.635000000000000	0.644129325218799
0.640000000000000	0.641861929493273
0.645000000000000	0.639599113396547
0.650000000000000	0.637340848546679
0.655000000000000	0.635087107019378
0.660000000000000	0.632837861337737
0.665000000000000	0.630593084462287
0.670000000000000	0.628352749781338
0.675000000000000	0.626116831101620
0.680000000000000	0.623885302639199
0.685000000000000	0.621658139010656
0.690000000000000	0.619435315224536
0.695000000000000	0.617216806673033
0.700000000000000	0.615002589123926
0.705000000000000	0.612792638712738
0.710000000000000	0.610586931935126
0.715000000000000	0.608385445639480
0.720000000000000	0.606188157019737
0.725000000000000	0.603995043608387
0.730000000000000	0.601806083269682
0.735000000000000	0.599621254193030
0.740000000000000	0.597440534886561
0.745000000000000	0.595263904170883
0.750000000000000	0.593091341172999
0.755000000000000	0.590922825320381
0.760000000000000	0.588758336335220
0.765000000000000	0.586597854228808
0.770000000000000	0.584441359296086
0.775000000000000	0.582288832110322
0.780000000000000	0.580140253517936
0.785000000000000	0.577995604633457
0.790000000000000	0.575854866834605
0.795000000000000	0.573718021757508
0.800000000000000	0.571585051292031
0.805000000000000	0.569455937577236
0.810000000000000	0.567330662996938
0.815000000000000	0.565209210175392
0.820000000000000	0.563091561973075
0.825000000000000	0.560977701482574
0.830000000000000	0.558867612024580
0.835000000000000	0.556761277143975
0.840000000000000	0.554658680606020
0.845000000000000	0.552559806392627
0.850000000000000	0.550464638698732
0.855000000000000	0.548373161928744
0.860000000000000	0.546285360693086
0.865000000000000	0.544201219804817
0.870000000000000	0.542120724276332
0.875000000000000	0.540043859316141
0.880000000000000	0.537970610325723
0.885000000000000	0.535900962896454
0.890000000000000	0.533834902806603
0.895000000000000	0.531772416018404
0.900000000000000	0.529713488675192

0.905000000000000	0.527658107098600
0.910000000000000	0.525606257785825
0.915000000000000	0.523557927406959
0.920000000000000	0.521513102802372
0.925000000000000	0.519471770980160
0.930000000000000	0.517433919113649
0.935000000000000	0.515399534538954
0.940000000000000	0.513368604752591
0.945000000000000	0.511341117409146
0.950000000000000	0.509317060318992
0.955000000000000	0.507296421446057
0.960000000000000	0.505279188905642
0.965000000000000	0.503265350962282
0.970000000000000	0.501254896027665
0.975000000000000	0.499247812658581
0.980000000000000	0.497244089554926
0.985000000000000	0.495243715557747
0.990000000000000	0.493246679647326
0.995000000000000	0.491252970941309
1	0.489262578692870
1.00500000000000	0.487275492288921
1.01000000000000	0.485291701248352
1.01500000000000	0.483311195220317
1.02000000000000	0.481333963982549
1.02500000000000	0.479359997439713
1.03000000000000	0.477389285621796
1.03500000000000	0.475421818682528
1.04000000000000	0.473457586897837
1.04500000000000	0.471496580664335
1.05000000000000	0.469538790497839
1.05500000000000	0.467584207031921
1.06000000000000	0.465632821016487
1.06500000000000	0.463684623316386
1.07000000000000	0.461739604910051
1.07500000000000	0.459797756888162
1.08000000000000	0.457859070452345
1.08500000000000	0.455923536913890
1.09000000000000	0.453991147692501
1.09500000000000	0.452061894315070
1.10000000000000	0.450135768414473
1.10500000000000	0.448212761728400
1.11000000000000	0.446292866098199
1.11500000000000	0.444376073467748
1.12000000000000	0.442462375882356
1.12500000000000	0.440551765487674
1.13000000000000	0.438644234528645
1.13500000000000	0.436739775348456
1.14000000000000	0.434838380387535
1.14500000000000	0.432940042182547
1.15000000000000	0.431044753365424
1.15500000000000	0.429152506662412
1.16000000000000	0.427263294893138
1.16500000000000	0.425377110969697
1.17000000000000	0.423493947895753
1.17500000000000	0.421613798765670
1.18000000000000	0.419736656763649
1.18500000000000	0.417862515162895
1.19000000000000	0.415991367324790
1.19500000000000	0.414123206698094
1.20000000000000	0.412258026818156
1.20500000000000	0.410395821306147
1.21000000000000	0.408536583868306
1.21500000000000	0.406680308295205
1.22000000000000	0.404826988461026
1.22500000000000	0.402976618322862
1.23000000000000	0.401129191920022
1.23500000000000	0.399284703373364
1.24000000000000	0.397443146884633

1.24500000000000	0.395604516735819
1.25000000000000	0.393768807288527
1.25500000000000	0.391936012983367
1.26000000000000	0.390106128339349
1.26500000000000	0.388279147953299
1.27000000000000	0.386455066499291
1.27500000000000	0.384633878728082
1.28000000000000	0.382815579466571
1.28500000000000	0.381000163617266
1.29000000000000	0.379187626157768
1.29500000000000	0.377377962140261
1.30000000000000	0.375571166691024
1.30500000000000	0.373767235009946
1.31000000000000	0.371966162370062
1.31500000000000	0.370167944117096
1.32000000000000	0.368372575669019
1.32500000000000	0.366580052515614
1.33000000000000	0.364790370218061
1.33500000000000	0.363003524408529
1.34000000000000	0.361219510789777
1.34500000000000	0.359438325134773
1.35000000000000	0.357659963286319
1.35500000000000	0.355884421156690
1.36000000000000	0.354111694727285
1.36500000000000	0.352341780048286
1.37000000000000	0.350574673238332
1.37500000000000	0.348810370484198
1.38000000000000	0.347048868040494
1.38500000000000	0.345290162229364
1.39000000000000	0.343534249440206
1.39500000000000	0.341781126129396
1.40000000000000	0.340030788820022
1.40500000000000	0.338283234101636
1.41000000000000	0.336538458630006
1.41500000000000	0.334796459126890
1.42000000000000	0.333057232379808
1.42500000000000	0.331320775241834
1.43000000000000	0.329587084631396
1.43500000000000	0.327856157532082
1.44000000000000	0.326127990992461
1.44500000000000	0.324402582125912
1.45000000000000	0.322679928110465
1.45500000000000	0.320960026188650
1.46000000000000	0.319242873667356
1.46500000000000	0.317528467917700
1.47000000000000	0.315816806374914
1.47500000000000	0.314107886538226
1.48000000000000	0.312401705970768
1.48500000000000	0.310698262299481
1.49000000000000	0.308997553215040
1.49500000000000	0.307299576471783
1.50000000000000	0.305604329887650
1.50500000000000	0.303911811344139
1.51000000000000	0.302222018786264
1.51500000000000	0.300534950222526
1.52000000000000	0.298850603724898
1.52500000000000	0.297168977428815
1.53000000000000	0.295490069533177
1.53500000000000	0.293813878300363
1.54000000000000	0.292140402056254
1.54500000000000	0.290469639190267
1.55000000000000	0.288801588155400
1.55500000000000	0.287136247468288
1.56000000000000	0.285473615709267
1.56500000000000	0.283813691522453
1.57000000000000	0.282156473615829
1.57500000000000	0.280501960761346
1.58000000000000	0.278850151795028

1.58500000000000	0.277201045617101
1.59000000000000	0.275554641192117
1.59500000000000	0.273910937549103
1.60000000000000	0.272269933781716
1.60500000000000	0.270631629048411
1.61000000000000	0.268996022572616
1.61500000000000	0.267363113642927
1.62000000000000	0.265732901613310
1.62500000000000	0.264105385903313
1.63000000000000	0.262480565998301
1.63500000000000	0.260858441449687
1.64000000000000	0.259239011875192
1.64500000000000	0.257622276959107
1.65000000000000	0.256008236452574
1.65500000000000	0.254396890173877
1.66000000000000	0.252788238008748
1.66500000000000	0.251182279910685
1.67000000000000	0.249579015901287
1.67500000000000	0.247978446070603
1.68000000000000	0.246380570577489
1.68500000000000	0.244785389649987
1.69000000000000	0.243192903585719
1.69500000000000	0.241603112752287
1.70000000000000	0.240016017587701
1.70500000000000	0.238431618600813
1.71000000000000	0.236849916371772
1.71500000000000	0.235270911552493
1.72000000000000	0.233694604867143
1.72500000000000	0.232120997112647
1.73000000000000	0.230550089159203
1.73500000000000	0.228981881950829
1.74000000000000	0.227416376505910
1.74500000000000	0.225853573917778
1.75000000000000	0.224293475355308
1.75500000000000	0.222736082063521
1.76000000000000	0.221181395364229
1.76500000000000	0.219629416656675
1.77000000000000	0.218080147418217
1.77500000000000	0.216533589205014
1.78000000000000	0.214989743652743
1.78500000000000	0.213448612477341
1.79000000000000	0.211910197475758
1.79500000000000	0.210374500526747
1.80000000000000	0.208841523591664
1.80500000000000	0.207311268715303
1.81000000000000	0.205783738026752
1.81500000000000	0.204258933740271
1.82000000000000	0.202736858156203
1.82500000000000	0.201217513661905
1.83000000000000	0.199700902732713
1.83500000000000	0.198187027932927
1.84000000000000	0.196675891916838
1.84500000000000	0.195167497429766
1.85000000000000	0.193661847309149
1.85500000000000	0.192158944485651
1.86000000000000	0.190658791984302
1.86500000000000	0.189161392925677
1.87000000000000	0.187666750527110
1.87500000000000	0.186174868103935
1.88000000000000	0.184685749070769
1.88500000000000	0.183199396942837
1.89000000000000	0.181715815337323
1.89500000000000	0.180235007974774
1.90000000000000	0.178756978680533
1.90500000000000	0.177281731386227
1.91000000000000	0.175809270131281
1.91500000000000	0.174339599064499
1.92000000000000	0.172872722445669

```
1.92500000000000  0.171408644647231
1.93000000000000  0.169947370155987
1.93500000000000  0.168488903574865
1.94000000000000  0.167033249624732
1.94500000000000  0.165580413146267
1.95000000000000  0.164130399101882
1.95500000000000  0.162683212577707
1.96000000000000  0.161238858785634
1.96500000000000  0.159797343065421
1.97000000000000  0.158358670886860
1.97500000000000  0.156922847852013
1.98000000000000  0.155489879697517
1.98500000000000  0.154059772296954
1.99000000000000  0.152632531663307
1.99500000000000  0.151208163951476
2  0.149786675460890
  end
end

  BEGIN Aria MATERIAL hmx
    density = constant  rho = 1
    specific heat = constant  cp = 1
    heat conduction = basic
    thermal conductivity = constant  k = 1
    pressure = constant value=3

    species names are AA
    species of AA = from_chemeq

    begin parameters for chemeq model hmx
      number of reactions = 1

      species names are AA
      species phases are Condensed
      Concentration units = moles

      Condensed Fraction = 0.0
      Steric Coefficients are 0.0
      Log Preexponential Factors are {log(5)}
      Activation Energies are 10.0
      Energy Releases are 0.0

      Concentration Exponents for AA are 0.0

      Stoichiometric coefficients for AA are -1.0

      # Pressure dependence
      Reference pressure = 2.
      Pressure exponents are 2.
      Pressure = From_Material_Definition

      #Distributed activation energy
      Activation energy st devs are 1.
      extent of reaction based on AA

    end parameters for chemeq model hmx

  end Aria MATERIAL hmx

  BEGIN GLOBAL CONSTANTS
    Ideal Gas Constant = 1.
  END GLOBAL CONSTANTS

  BEGIN FINITE ELEMENT MODEL block
    Database Name = 1block.g
    decomposition method = rib

    Use material hmx for block_1
```

```
  END FINITE ELEMENT MODEL block

  BEGIN TPETRA EQUATION SOLVER  SOLVE_TEMPERATURE
    BEGIN CG SOLVER
      BEGIN JACOBI PRECONDITIONER
      END
      MAXIMUM ITERATIONS = 1000
      RESIDUAL SCALING = NONE
      CONVERGENCE TOLERANCE = 1.000000e-12
    END
  END TPETRA EQUATION SOLVER

Begin Universal Aria Expressions
  user expression = user_function name = A_exact user_tag = "exactAA" x = "time"
End

  BEGIN procedure myProcedure

  begin solution control description
    Use System Main
    Begin System Main
      Simulation Start Time       = 0.0
      Simulation Termination Time = 2.0
      Begin Transient time_block
        advance myregion
      End Transient time_block
    End System Main

      BEGIN parameters for transient time_block
        start time = 0.0
        termination time = 2.0

        BEGIN PARAMETERS FOR Aria REGION myRegion
          time step variation = fixed
          time integration method = second_order
          initial time step size = {0.01*0.5}
        END PARAMETERS FOR Aria REGION myRegion

     END parameters for transient time_block

  end solution control description

    begin aria region myRegion

    postprocess L_2_norm of function "species_AA - exactAA" on all_blocks as l_2_A

    EQ energy for temperature on all_blocks using Q1 with diff mass src
    Source for Energy on all_blocks = chemeq_heating MODEL = hmx
    IC for temperature on all_blocks = constant value=3

    maximum nonlinear iterations = 10
    nonlinear residual tolerance = 1.0E-10
    nonlinear correction tolerance = 1.0E-10
    Nonlinear Relaxation Factor = 1.

    BEGIN CHEMEQ SOLVER FOR hmx
      ODE SOLVER = CVODE ADAMS 12 FUNCTIONAL
      Absolute Tolerance = 1e-12
      Relative Tolerance = 1e-10
      Activation Temperature = 0.0
      species AA = 1.0
    END CHEMEQ SOLVER FOR hmx

    BEGIN RESULTS OUTPUT LABEL diffusion output
       database Name = output.e
       At Step 0, Increment = 1
       Timestep Adjustment Interval = 1
       Title Aria Chem/Diffusion Verification
```

```
      Nodal Variables = solution->temperature as T
      Nodal Variables = exact_A
      Element Variables = AA
   END RESULTS OUTPUT LABEL diffusion output

   USE FINITE ELEMENT MODEL block
   usE LINEAR SOLVER solve_temperature

   Begin Heartbeat pp_out
      Legend = off
      Labels = off
      At Step 1 Interval is 1
      Precision = 6
      Stream Name = error.txt
      variable is Global time
      Global L_2_A
   End


   END Aria REGION myRegion

  END procedure myProcedure

END SIERRA Aria
```

12.8. MISCELLANEOUS

12.8.1. Thermal Postprocessing

```
BEGIN SIERRA myJob

   BEGIN ARIA MATERIAL Kryptonite
      Density                = CONSTANT rho = 1
      Thermal Conductivity   = Constant k   = 1
      Specific Heat          = Constant cp  = 1
      heat conduction     = basic
   END

   BEGIN ARIA MATERIAL Mathite
      Density                = CONSTANT rho = 1
      Thermal Conductivity   = Constant k   = 1
      Specific Heat          = Constant cp  = 1
      heat conduction     = basic
   END

   Begin Global Constants
      Stefan Boltzmann Constant = 5.67e-8
   End


# T0 = {T0 = 400}
# h = {h = 10}
# C0 = {C0 = 2}
# C1 = {C1 = 3}
# C2 = {C2 = 4}
# C3 = {C3 = 0.4}
# eps = { eps = 0.6 }
# sigma = { sigma = 5.67e-8}

   Begin universal aria expressions
      User Expression = Scalar_String_Function \$
        f = "{C0}*(x^2-1) + {C1}*(y^2-0.25) + {C2}*(z^2-0.25) + {C3}*t" \$
        user_tag = arg

      User Expression = Scalar_String_Function \$
        f = "{T0} + exp(arg)" \$
        user_tag = T_mms

      User Expression = Vector_String_Function \$
        f_x = "{C0}*2*x*exp(arg)" \$
        f_y = "{C1}*2*y*exp(arg)" \$
        f_z = "{C2}*2*z*exp(arg)" \$
        user_tag = gradT

      User Expression = Scalar_String_Function \$
        f = "2*exp(arg)*({C0}*(2*{C0}*x^2+1) + {C1}*(2*{C1}*y^2+1) + {C2}*(2*{C2}*z^2+1))" \$
        user_tag = laplacianT

      User Expression = Scalar_String_Function \$
        f = "-laplacianT" \$
        user_tag = mms_src

      User Expression = Scalar_String_Function \$
        f = "T_mms + gradT_z/{h}" \$
        user_tag = conv_Tref

      User Expression = Scalar_String_Function \$
        f = "(T_mms^4 + gradT_y/({eps*sigma}))^0.25" \$
        user_tag = rad_Tref

      User Expression = Scalar_String_Function \$
```

```
      f = "{h}*(T_mms - conv_Tref)" \$
      user_tag = conv_flux_mms

    User Expression = Scalar_String_Function \$
      f = "{sigma*eps}*(T_mms^4 - rad_Tref^4)" \$
      user_tag = rad_flux_mms

 End

 Begin Aria Material surf_2_models
    BC Reference Temperature  = Scalar_String_Function f = "conv_Tref"
    Heat Transfer Coefficient = constant h={h}
 End

 Begin Aria Material surf_3_models
    BC Rad Reference Temperature = Scalar_String_Function f = "rad_Tref"
    Emissivity                   = Constant E={eps}
    Radiation form factor        = Constant F=1.0
 End

BEGIN TPETRA EQUATION SOLVER  SOLVE_TEMPERATURE
  BEGIN CG SOLVER
    BEGIN JACOBI PRECONDITIONER
    END
    MAXIMUM ITERATIONS = 1000
    RESIDUAL SCALING = R0
    CONVERGENCE TOLERANCE = AUTOMATIC
  END
END TPETRA EQUATION SOLVER


 BEGIN FINITE ELEMENT MODEL cube
    database name = cube_two_blocks_hex8_h{N}.g
    coordinate system is cartesian

    # [0,1] x [-0.5,0.5] x [-0.5,0.5]
    BEGIN PARAMETERS FOR BLOCK block_1
       material Kryptonite
    END

    # [-1,0] x [-0.5,0.5] x [-0.5,0.5]
    BEGIN PARAMETERS FOR BLOCK block_2
       material Mathite
    END

    Use Material surf_2_models for surface_2
    Use Material surf_3_models for surface_3

 END FINITE ELEMENT MODEL cube


 BEGIN PROCEDURE myAriaProcedure

    Begin Solution Control Description
       Use System Main
       Begin System Main
            Begin Transient MySolveBlock
             Advance myRegion
          End
          Simulation Max Global Iterations = 1
          Simulation Start Time = 0
          Simulation Termination Time = 1
       End
       Begin Parameters for Transient MySolveBlock
       End
    End

    BEGIN ARIA REGION myRegion
```

```
 use finite element model cube
 use linear solver solve_temperature

 nonlinear solution strategy = newton

 maximum nonlinear iterations = 10
 nonlinear residual tolerance = 1.0e-16
 nonlinear correction tolerance = 1.0e-12
 nonlinear relaxation factor  = 1.0
 use dof averaged nonlinear residual

 EQ ENERGY for TEMPERATURE on all_blocks using Q1 with DIFF SRC

 MESH GROUP Dirichlet_Surface = surface_4 surface_5 surface_6 surface_7
 mesh group block_1_2 = block_1 block_2
 BC Dirichlet for Temperature on Dirichlet_Surface = scalar_string_function f = "T_mms"

 BC Flux for Energy on surface_2 = Generalized_Nat_Conv Power_Output=cf_bc_ipo Flux_Output=cf_bc_ifo
 BC Flux for Energy on surface_3 = Generalized_Rad Power_Output=rf_bc_ipo Flux_Output=rf_bc_ifo

 SOURCE for ENERGY on all_blocks = scalar_string_function f = "mms_src" Power_Output=src_ipo


 postprocess integral of expression conv_flux_mms on surface_2 as cf_bc_ipo_ex
 postprocess integral of expression rad_flux_mms on surface_3 as rf_bc_ipo_ex

 postprocess integral of expression mms_src on block_1_2 as src_ipo_ex

 postprocess average of expression conv_flux_mms on surface_2 as cf_bc_ifo_ex
 postprocess average of expression rad_flux_mms on surface_3 as rf_bc_ifo_ex

 postprocess value of expression T_mms on block_1_2 as T_exact
 postprocess l_2_norm of function "T_mms - TEMPERATURE" on block_1_2 as l2_error

 postprocess value of field nonlinear_solution->temperature at point -0.151720462393008   0.146935733548329  -0.39364140

 postprocess value of field T_exact at point -0.151720462393008   0.146935733548329  -0.393641401879319 as eval_b1_ex

 postprocess value of field nonlinear_solution->temperature at point 0 0.162595269728099  -0.377464159584852 as eval_b1b

 postprocess value of field T_exact at point 0 0.162595269728099  -0.377464159584852 as eval_b1b2_ex

 postprocess value of field nonlinear_solution->temperature at point -0.855758209426849 0.159603369582751 0.5 as eval_s2

 postprocess value of field T_exact at point -0.855758209426849 0.159603369582751 0.5 as eval_s2_ex

postprocess global_function "cf_bc_ipo_ex - cf_bc_ipo" as cf_bc_ipo_err
postprocess global_function "rf_bc_ipo_ex - rf_bc_ipo" as rf_bc_ipo_err
postprocess global_function "cf_bc_ifo_ex - cf_bc_ifo" as cf_bc_ifo_err
postprocess global_function "rf_bc_ifo_ex - rf_bc_ifo" as rf_bc_ifo_err
postprocess global_function "src_ipo_ex - src_ipo" as src_ipo_err
postprocess global_function "eval_b1_ex - eval_b1" as eval_b1_err
postprocess global_function "eval_b1b2_ex - eval_b1b2" as eval_b1b2_err
postprocess global_function "eval_s2_ex - eval_s2" as eval_s2_err

Begin Heartbeat The_Heartbeat
  Stream Name         = globals{N}.txt
  Timestamp Format    = "" # Omit time stamps for diffing purposes.
  Precision           = 8
  At Step 0 increment = 1
  Legend = on
  Labels = off
  Format = csv
  variable is Global time
  variable is Global number_of_nodes as num_nodes
  variable is Global l2_error
  variable is Global cf_bc_ipo_err
```

```
          variable is Global rf_bc_ipo_err
          variable is Global cf_bc_ifo_err
          variable is Global rf_bc_ifo_err
          variable is Global eval_b1_err
          variable is Global eval_b1b2_err
          variable is Global eval_s2_err
        End

      END ARIA REGION myRegion

   END PROCEDURE myAriaProcedure

END SIERRA myJob
```

12.8.2. Postprocess Min/Max

12.8.3. Local Coordinates: Cartesian

12.8.4. Local Coordinates: Cylindrical

This page intentionally left blank.

Appendices

This page intentionally left blank.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	01911	sanddocs@sandia.gov

This page intentionally left blank.

This page intentionally left blank.

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.