TRUCHAS Users Manual

The TELLURIDE Team

Version 2.4.0
February 14, 2008

LA-UR-08-0815

LA-UR-08-0815

This Users Manual for the TRUCHAS multiphysics simulation code is comprised primarily of sample problems that demonstrate the varied capabilities of the code in a tutorial style. Beginning with a simple heat conduction problem, a series of examples guides the new user through problems of increasing complexity and varying physics. The input file for the heat conduction problem is discussed in its entirety. Having established a baseline input file, for subsequent problems we point out only changes that highlight added input parameters or new types of settings. Other chapters explain how to set up parallel runs and how to control code output. The entire set of input files for all problems is reproduced in the Appendix. The last chapter is devoted to expounding some of the algorithmic and operational caveats currently facing TRUCHAS users and developers.

Other documents in the TRUCHAS collection are complementary to this one. Scope and method of calculation for each type of physics simulation are described in the *Physics and Algorithms Manual*. Supported computer architectures, installation of the code, and compiling and building are covered in the *Installation Guide*. Finally, the *Reference Manual* gives a detailed itemization of the keywords used in TRUCHAS's Fortran-style input namelists, as well as their usage and allowed values.

LA-UR-08-0815

LA-UR-08-0815

Contents

Contents i

1 Introduction 1

1.1 Structure . . . 1

1.2 Description of Problems . . . 2

1.3 Units . . . 2

1.4 General Information . . . 4

2 Heat Conduction 5

2.1 TRUCHAS Capabilities Demonstrated . . . 6

2.2 Problem Description . . . 6

2.3 Setup . . . 6

2.4 Running the problem; results . . . 13

2.5 Temperature Variation; Initial Gradients and Time-varying Temperature Boundary Conditions 14

3 Isothermal Phase Change 19

3.1 TRUCHAS Capabilities Demonstrated . . . 19

3.2 Problem Description 20
3.3 Setup 21
3.4 Results 29

4 Binary Alloy Phase Change **33**

4.1 TRUCHAS Capabilities Demonstrated 33
4.2 Problem Description 34
4.3 Setup 36
4.4 Results 38

5 Chemistry **41**

5.1 TRUCHAS Capabilities Demonstrated 41
5.2 Problem Description 42
5.3 Setup 42
5.4 Results 45

6 Flow and Filling **47**

6.1 TRUCHAS Capabilities Demonstrated 48
6.2 Problem Description 48
6.3 Setup 48
6.4 Results 53

7 Multiple Phase Changes + Buoyancy-Driven Flow **55**

7.1 TRUCHAS Capabilities Demonstrated 55

7.2 Problem Description 56
7.3 Setup 56
7.4 Results 60

8 Elastic and Viscoplastic Deformation 63

8.1 TRUCHAS Capabilities Demonstrated 63
8.2 Problem Description 64
8.3 Setup 64
8.4 Results 70

9 Internal Interface Displacements 73

9.1 TRUCHAS Capabilities Demonstrated 73
9.2 Problem Description 74
9.3 Setup 74
9.4 Results 83

10 Induction Heating 85

10.1 TRUCHAS Capabilities Demonstrated 86
10.2 Problem Description 86
10.3 Setup 86
10.4 Results 91

11 Radiation 93

11.1 TRUCHAS Capabilities Demonstrated 93

11.2 Problem Description 94
11.3 Setup 96
11.4 Results 102

12 Diffusion Solver **105**

12.1 Problem Description 106
12.2 Setup 106
12.3 Results 109

13 Parallel Execution **111**

13.1 Building a Parallel Executable 112
13.2 Modifying the Input File 112
13.3 Invoking the Parallel Executable 113
13.4 Modifying the Input File, continued 116

14 Output **119**

14.1 Overview 119
14.2 `OUTPUTS` Namelist Variables 120
14.3 The TBrook.xml File 121
14.4 Postprocessing the XML Output 121

15 Continuing a Run from a Restart Dump **139**

15.1 Overview 139
15.2 Restart Considerations 140

15.3 Options for Restarts: the RESTART namelist 140

16 Caveats **143**

16.1 Use of Discrete Operators 143

16.2 Volume Tracking and Reconstruction 144

16.3 Fluid Flow Algorithm 144

16.4 Phase Change Algorithm 145

16.5 Restrictions Due to SI Units 145

16.6 Solvers 146

16.7 Solid Mechanics Algorithm 146

16.8 Electromagnetics Algorithm 146

16.9 Operational Issues 147

A Input File: Heat Conduction **149**

B Input File:Isothermal Phase Change **153**

C Input File: Binary Alloy Phase Change **161**

D Input File:Chemistry **167**

E Input File: Flow and Filling **173**

F Input File: Multiple Phase Changes + Buoyancy Driven Flow **177**

G Input File: Elastic and ViscoPlastic Deformation **183**

H Input File: Internal Interface Displacements **191**

I Input File: Induction Heating **199**

J Input File: Radiation **203**

K Input File: Diffusion Solver **211**

L Input File: Parallel Heat Conduction **213**

Chapter 1

Introduction

In this Users Manual we describe and demonstrate the capabilities of TRUCHAS. We begin with a simple conductive heat transfer problem and progress to problems involving phase change, fluid flow, radiative heat transfer, chemical reactions, solid deformation, and electromagnetics. Further chapters explain how to run TRUCHAS in parallel on a cluster, how to obtain the different kinds of output that TRUCHAS can generate, and some of the limitations, sensitivities, and uncertainties a user may confront in executing TRUCHAS.

1.1 Structure

Each chapter covering a sample problem is divided into the following sections:

1 Capabilities: a listing of the different capabilities demonstrated by that chapter.

2 Problem Definition: a brief description of the problem.

3 Setup: Dissection of relevant sections of the input file.

4 Results: Presentation of results calculated by TRUCHAS.

5 Input: A description of the input file needed to run the problem. For each problem we describe only the new or different features. Full copies of the input files are available in the Appendix.

1.2 Description of Problems

The list of chapters and the physics they describe are presented in table Table 1.2:

A TRUCHAS input file may contain all of the following namelists. For some applications a reduced set is appropriate; see the following sections.

- Defining a mesh: the MESH namelist
- Specifying physics: the PHYSICS namelist
- Defining material properties: MATERIAL namelists
- Initializing the domain: BODY namelists
- Applying boundary conditions: BC namelists
- Selecting output formats: the OUTPUTS namelist
- Defining numerical methods: the NUMERICS namelist
- Non-linear solvers: NONLINEAR_SOLVER namelists
- Linear solvers: LINEAR_SOLVER namelists
- Defining chemical reactions: CHEMICAL_REACTION namelists
- Defining phase changes: PHASE_CHANGE_PROPERTIES namelists
- Defining electromagnetics : the ELECTROMAGNETICS namelist

1.3 Units

A few physical constants in TRUCHAS are given default values that depend on choosing an SI system of units, which includes the *kelvin* (K) as the unit of temperature. These are

- the gas constant R used in modeling chemical reactions.
- the Stefan-Boltzmann constant used in applying radiation boundary conditions (see the `BC` chapter of the *Reference Manual*)
- the electromagnetic constants ϵ_0 and μ_0; the default values assume an SI system (see the `ELECTROMAGNETICS` chapter of the *Reference Manual*).;

Chapter	Description	HT	HS	IP	BA	CH	FF	BF	SD	DB	EM	RT	3D	PE
2	Heat Conduction	⋆												
3	Phase Change, Pure	⋆	⋆	⋆									⋆	
4	Phase Change, Alloy	⋆			⋆									
5	Chemistry	⋆				⋆								
6	Flow and Filling						⋆							
7	Flow + Multiple Phase Change	⋆		⋆			⋆	⋆						
8	Elastic and Viscoplastic Deformation	⋆							⋆				⋆	
9	Internal Displacements								⋆	⋆			⋆	
12	Induction Heating	⋆	⋆								⋆		⋆	
11	Enclosure Radiation	⋆										⋆	⋆	
13	Heat Conduction in Parallel	⋆											⋆	⋆

Table 1.1: Listing of TRUCHAS capabilities demonstrated in each chapter's sample problem. The abbreviations are: HT, heat transfer; HS, volumetric heat source; IP, isothermal phase change; BA, binary alloy phase change; CH, chemsitry; FF, fluid flow; BF, buoyancy-driven flow; SD, solid deformation; DB, displacement boundary; EM, electromagnetics; RT, radiative transfer; 3D, three-dimensional domain; and PE, parallel execution.

In most uses of temperatures, for example in heat conduction and phase change, only relative temperatures are important. However in radiative heat transfer temperatures must be specified on the absolute Kelvin scale.

The user may change to any unit system if all constants needed are redefined in that system—saving the above constants, the code is unitless. Regardless of the unit system chosen, all values of input data for properties, both material and physical, must be consistent. Failure to ensure this can lead TRUCHAS to quantitatively unphysical results that are challenging to understand and correct.

1.4 General Information

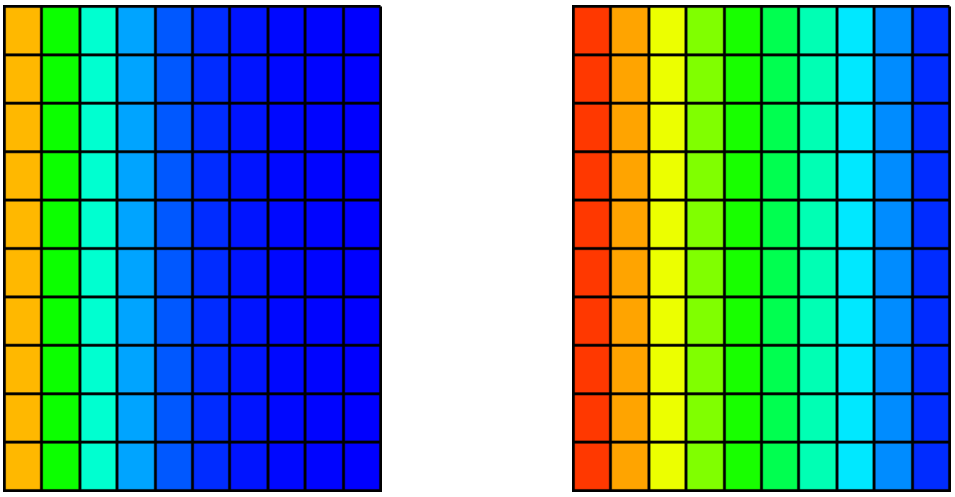
In reading this manual, you will find it useful to have the TRUCHAS *Reference Manual* handy, since we only discuss different namelists with respect to the problems at hand. At best this covers a small fraction of the possible choices.

Before setting up a new problem, we recommend that the user identify the TRUCHAS capabilities needed and use the input files from relevant chapters in this book as a guide in generating one's own input. After studying the *Reference Manual*, if one still has trouble getting a problem to run, one can send e-mail to `telluride-support@lanl.gov`.

On to the sample problems ...

Chapter 2

Heat Conduction



To demonstrate the heat transfer capabilities of TRUCHAS, we set up a simple heat conduction problem on an internally generated, quasi-two-dimensional block orthogonal mesh. The input file is discussed in its entirety, and sample results are presented.

A variation of this problem that demonstrates initial temperature gradients and time-varying temperature boundary conditions is also presented in section Section 2.5.

2.1 TRUCHAS Capabilities Demonstrated

- Heat transfer
- Internal block mesh generation
- Constant temperature boundary condition
- Insulating boundary condition
- Nonlinear solver
- Linear solver
- Constant time step

Also in Section 2.5, the following additional capabilities are demonstrated.

- Initializing a temperature gradient
- Time-varyign temperature boundary condition

2.2 Problem Description

We begin with a unit square domain. The top and bottom of the square are insulated, and the initial temperature of the domain is zero. At time zero we raise the temperature along the right edge to 100 degrees (the unit is K, which is equivalent to a degree Celsius, and the scale need not be absolute temperature), while maintaining the temperature of the left edge at zero. We then solve for the transient evolution of temperature between the two edges, as it approaches a steady state in which there is a linear variation in temperature from low to high.

Units:	SI
Input file:	UMProblems/Chapter_hc/hc.inp
Mesh file:	None

2.3 Setup

The input file for this problem is presented in Appendix A. Here we discuss each section of the input file in detail.

2.3.1 Defining a mesh; the `MESH` namelist

Namelist instances allowed: single

```
&MESH
  ncell   = 10, 10, 1
  coord   = 3*-0.5, 3*0.5
/
```

The MESH namelist defines the simulation domain. As we shall see in the next chapter, we will usually specify the name of a mesh file that TRUCHAS can read, created by some mesh generation software such as CUBIT. If no mesh file is available, as is the case here, TRUCHAS can generate a block mesh internally, using the information presented in a namelist for the number of cells in each of three directions and the coordinate limits.

In this case we define a unit cube extending between -0.5 and 0.5 in each of the three directions, with 10 cells in the x and y directions and a single cell (the default) in the z direction, effectively specifying a two-dimensional domain. Note that since the input file is a Fortran namelist, all Fortran I/O conventions are valid. Consequently, $3 * -0.5$ is interpreted as $-0.5, -0.5, -0.5$. Moreover, all lines in the input file are case-insensitive, so the user can employ the case that makes the content clearest.

See the MESH chapter of the *Reference Manual* for further details on using the MESH namelist.

2.3.2 Specifying physics; the PHYSICS namelist

Namelist instances allowed: single

```
&PHYSICS
  fluid_flow       = .false.
  heat_conduction  = .true.
/
```

The PHYSICS namelist specifies the types of physics we are activating for this problem. By default TRUCHAS runs fluid flow, but for a pure heat conduction problem we want to deactivate it, hence we set `fluid_flow` to `.false.` To activate heat conduction, we set `heat_conduction` to `.true.` For a list of different physics that can be activated refer to the PHYSICS chapter of the *Reference Manual*.

2.3.3 Defining materials; MATERIAL namelists

Namelist instances allowed: multiple, one for each phase of each material in the problem.

```
&MATERIAL
  material_number         = 1
  material_name           = 'solid_Cu'
  material_feature        = 'background'
  density                 = 8960.
  cp_constants            = 385.
  conductivity_constants  = 400.
/
```

Materials namelists define the properties of materials present in the problem. These include the $density$, specific heat C_p, and $thermal\ conductivity$. For a complete list of keywords see the `MATERIAL` chapter of the *Reference Manual*. Each material is assigned a unique ID by the keyword `material_number`. It is by this number that this material will be referred to in other namelists such as `BC` and `BODY`. Here the name `solid_Cu` is used to identify the material in all outputs.

2.3.4 Initializing the domain; `BODY` namelists

Namelist instances allowed: multiple, as many as required to initialize the domain

```
&BODY
  material_number   = 1
  surface_name      = 'background'
  temperature       = 0.
/
```

The `BODY` namelist is used to initialize regions of the simulation domain. In this example we set the material with `material_number` = 1 as the 'background' material and initialize it with temperature $T = 0$. TRUCHAS requires that one and only one material in the list be designated as the 'background' material by being given the `material_feature` 'background.' That material will be assigned to any regions of the domain where no other material is specified—everywhere, in this case.

For more advanced uses of the `BODY` namelist take a look at Chapter 4, Section 4.3.3 and Chapter 6, Section 6.3.4.

2.3.5 Applying boundary conditions; `BC` namelists

Namelist instances allowed: multiple, one for each boundary condition that we must specify.

TRUCHAS analyzes the problem geometry and detects all 'external' faces of bodies, *i.e.* those that do not have a neighboring cell on the other side. *All external faces in a mesh must be contained in a surface that has a temperature boundary condition defined in a* `BC` *namelist.* If at the end of initialization there are external faces without a defined temperature boundary condition, TRUCHAS will exit with a fatal error. Pressure, velocity (*e.g.* applied inflow), and displacement boundary conditions need not be defined on any external surfaces.

For our current problem we have three `BC` namelists, one for each of the three boundary conditions: constant $T = 100$ along the left edge, constant $T = 0$ along the right edge, and insulated along all other external cell boundaries.

2.3.5.1 `BC` namelist 1: Constant T=100 along the left edge, x=-0.5

```
&BC
  surface_name     = 'conic'
  conic_relation   = '='
  conic_x          = 1.
  conic_constant   = 0.5
  conic_tolerance  = 1.e-6
  bc_variable      = 'temperature'
  bc_type          = 'dirichlet'
  bc_value         = 100.
/
```

The setting `surface_name='conic'` means that the boundary lies within `conic_tolerance` of a surface described by the conic equation:

$$\begin{aligned}\mathcal{F} = \;& \texttt{conic\_constant} \\ &+ \texttt{conic\_xx} * x^2 + \texttt{conic\_x} * x \\ &+ \texttt{conic\_yy} * y^2 + \texttt{conic\_y} * y \\ &+ \texttt{conic\_zz} * z^2 + \texttt{conic\_z} * z \\ &+ \texttt{conic\_xy} * x * y + \texttt{conic\_xz} * x * z + \texttt{conic\_yz} * y * z\end{aligned}$$

where x, y, and z are the three coordinates of an external cell face center, and the rest of the terms are explained in the `BC` chapter of the *Reference Manual*. Any face center that has $||\mathcal{F}|| <$ `conic_tolerance`

is considered part of the boundary. Any prefactors not specified in the namelist are assumed to be equal to 0. In this case we specify that along the surface defined by $1 * x = 0.5 \pm 10^{-6}$ we want to set the value of the temperature to 100.

2.3.5.2 BC namelist 2: Constant T=0 along the right edge, x=0.5

```
\&BC
  surface_name     =  'conic'
  conic_relation   =  '='
  conic_x          =  1.
  conic_constant   =  -0.5
  conic_tolerance  =  1.e-6
  bc_variable      =  'temperature'
  bc_type          =  'dirichlet'
  bc_value         =  0.
/
```

This is similar to the first namelist instance, except that we set the temperature to zero along $x = 0.5 \pm 10^{-6}$.

2.3.5.3 BC namelist 3: Insulate all other external boundaries

```
&BC
  surface_name       =  'external material boundary'
  surface_materials  =  1
  bc_variable        =  'temperature'
  bc_type            =  'hneumann'
/
```

This namelist specifies that for any external surfaces (`'external material boundary'`) where the surface is made of material 1 (`'solid_Cu'`), insulate it. `'hneumann'` indicates a homogeneous Neumann ('zero flux') condition.

2.3.6 Selecting output formats; the OUTPUTS namelist

Namelist instances allowed: single

```
&OUTPUTS
  output_t                  =  0., 3.e3
  output_dt                 =  300.
  xml_output_dt_multiplier  =  1
/
```

The `OUTPUTS` namelist controls what files are generated during a run. This namelist says that we want outputs from $t = 0$ till $t = 3000$ at intervals of 300 seconds. *The upper limit of the output determines how long the code runs.*

The `xml_output_dt_multiplier = 1` setting indicates that we want XML output at every `output_dt` (= 300s in this case). This XML output is then interrogated by a postprocessor parser to generate graphical or restart formats. Note that the output time step is different from the actual time step used in the calculation.

See Chapter 14 and the `OUTPUTS` chapter of the *Reference Manual* for details.

2.3.7 Numerics; the `NUMERICS` namelist

Namelist instances allowed: single

The `NUMERICS` namelist is where we define details of the numerical solution methods for each physics type in the problem. This has two parts: the numerical controls themselves and the pertinent solvers. For a detailed description see the `NUMERICS` chapter of the *Reference Manual*.

```
&NUMERICS
  dt_constant                =  100.
  energy_nonlinear_solution  =  'nk-energy'
/
```

Here we set the time step to a constant 100 seconds and indicate that we want to use the nonlinear solver with the label 'nk-energy' for heat transfer. Note that we can have multiple solvers defined in TRUCHAS input files and pick and choose between them. The only constraint is that any solver referenced in the `NUMERICS` namelist must exist in the input file (linear solvers in `LINEAR_SOLVER` namelists, and nonlinear solvers in `NONLINEAR_SOLVER namelists`) and must be unique.

2.3.8 Nonlinear solvers; NONLINEAR_SOLVER namelists

Namelist instances allowed: multiple, one for each solver referenced in the NUMERICS namelist. Available nonlinear solvers are described in the NONLINEAR_SOLVER chapter of the *Reference Manual*.

```
&NONLINEAR_SOLVER
  name                    =  'nk-energy'
  method                  =  'nk'
  linear_solver_name      =  'energy-linear'
  convergence_criterion   =  1.e-8
  use_damper              =  .true.
  perturbation_parameter  =  1.e-6
  damper_parameters       =  0.5, 2., 1., 1.
/
```

This nonlinear solver namelist is labeled 'nk-energy'. It uses the Newton-Krylov (nk) method for solving the heat transfer equations. The linear solver to be used to initialize this method is indicated as the one with the label 'energy-linear'. For a detailed description of the rest of the parameters take a look at the NUMERICS chapter of the *Reference Manual*.

2.3.9 Linear solvers; LINEAR_SOLVER namelists

Namelist instances allowed: multiple, one for each of the linear solvers referenced in the input file.

In this file only one linear solver is referenced, so only one linear solver namelist is needed.

```
&LINEAR_SOLVER
  name                    =  'energy-linear'
  method                  =  'gmres'
  preconditioning_method  =  'ssor'
  preconditioning_steps   =  2
  relaxation_parameter    =  1.4
  convergence_criterion   =  1.e-5
/
```

Here we specify that the nonlinear solver should use the GMRES method with a two stage SSOR preconditioner. The convergence criterion is set to 1.e-5.

2.4 Running the problem; results

The next step is to run the problem. This is done by issuing the following command:

```
% truchas hc.inp
```

Ensure that `truchas`, the executable program, is in your path, and that `hc.inp` lies in the current working directory.

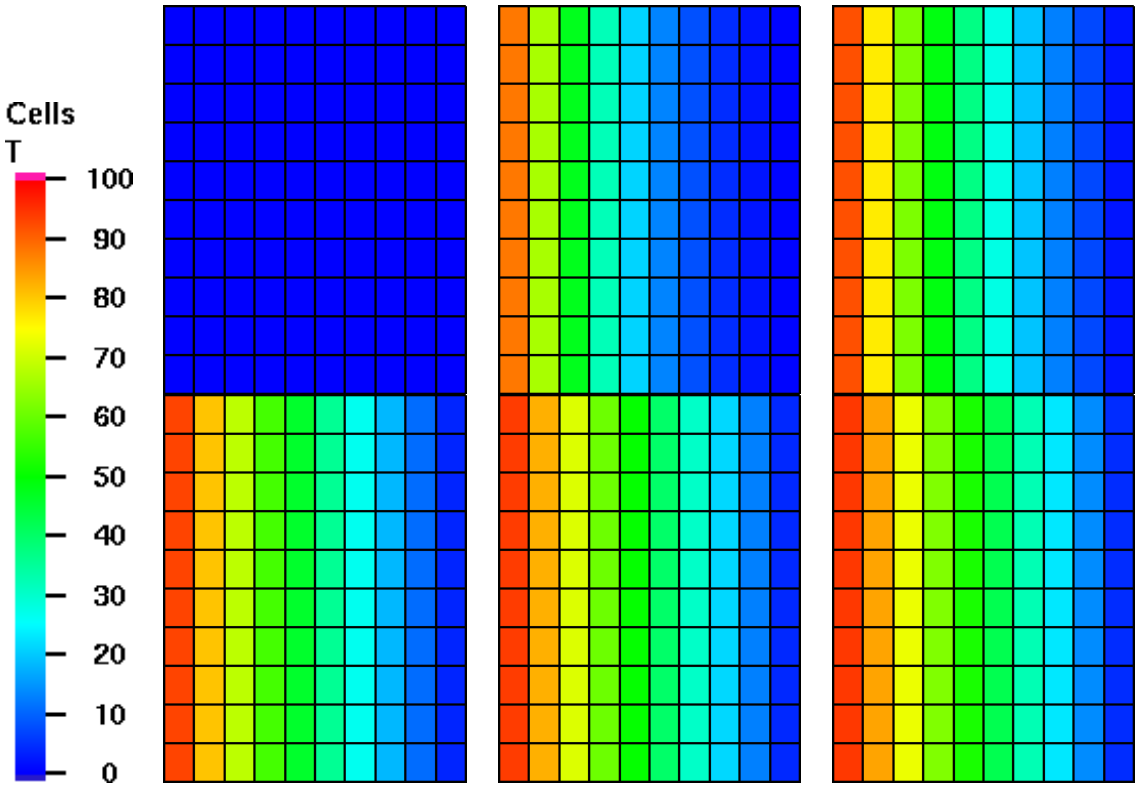


Figure 2.1: Time progression of the temperature profile within the Cu block at times 0, 600 and 1200 s (top row) and 1800, 2400, and 3000 s (bottom). The temperature scale is shown on the left, and the view is down the z-axis. As expected, the temperature of the block starts rising from right to left until it reaches a steady state profile at around 1800 s.

While the code runs, it periodically outputs lines to the screen updating the user about the status of the calculation. If execution of the `truchas` code needs to be halted for any reason, it can be stopped gracefully—only while performing time step cycling, not while initializing— by first obtaining the process IDs (PIDs) of running processes:

```
ps -x
```

and then sending a URG signal:

```
kill -s URG nnnnn
```

where ‘nnnnn’ is the lowest process ID of the `truchas` processes, of which there will be one for each CPU in a parallel job. Ignore the `mpi` job that spawned the `truchas` processes; the signal must go the the zeroeth `truchas` process.

After TRUCHAS has finished, the XML output from the program is parsed (see Section 14.4 for a tutorial on the parser) to provide the `GMV` files that are presented in Figure 2.1. The temperature scale for the problem is on the left hand side, with blue representing 0 deg and red representing 100 deg.

2.5 Temperature Variation; Initial Gradients and Time-varying Temperature Boundary Conditions

In this section, the problem discussed above is modified to use two additional TRUCHAS options:

- Initializing a temperature gradient
- Time-varying temperature boundary condition

The modified problem sets up conditions that are equivalent to the final state of the original problem, and then varies it through a time-varying boundary condition.

2.5.1 Modified Problem Description

On the same unit square domain, the temperature is intialized with a linear gradient in the X direction from 100 degrees at X=-0.5 to 0 degrees at X=0.5. This is the steady state of the original problem. The initial boundary conditions are the same as before, but at X=0.5, the boundary temperature is raised to 100 degrees over 1000 seconds. We solve for the transient temperature evolution as it approaches a uniform steady state temperature of 100 degrees everywhere.

2.5.2 Modified Setup

Three namelists are changed to created this modified setup: BODY, one BC, and OUTPUTS.

Input file: UMProblems/Chapter_hc/hctg.inp

2.5.2.1 Adding an Initial Temperature Gradient

The BODY namelist is modified to add the gradient. The base temperature is changed to 100 degrees and gradient parameters are inserted to establish a gradient in the X direction that lowers the temperature to zero degrees at X = 0.5.

```
&BODY
  material_number   = 1
  surface_name      = 'background'
  temperature       = 100.
  TG_Origin         = -0.5, 0.0, 0.0
  TG_Axis           =  1.0, 0.0, 0.0
  TG_Z_Constants    =  -1.0, 0.0, 0.0
/
```

The temperature gradients that can be specified in TRUCHAS are specified with respect to a gradient origin TG_Origin and a gradient direction (axis) TG_Axis. Variation can be specified both in the axis direction, referred to a the gradient dZ direction, and radially from that axis, referred to as the gradient dR direction.

The functional form for calculating the temparature at a cell center is given by

$$\begin{aligned} T(dZ,dR) &= T_{Origin} \\ &* \ (1 \ + \ \texttt{TG\_Z\_Constants}(1) * dZ^{\texttt{TG\_Z\_Exponents}(1)} \\ &\quad + \ \texttt{TG\_Z\_Constants}(2) * dZ^{\texttt{TG\_Z\_Exponents}(2)} \\ &\quad + \ \texttt{TG\_Z\_Constants}(3) * dZ^{\texttt{TG\_Z\_Exponents}(3)} \\ &\quad + \ \texttt{TG\_R\_Constants}(1) * dR^{\texttt{TG\_R\_Exponents}(1)} \\ &\quad + \ \texttt{TG\_R\_Constants}(2) * dR^{\texttt{TG\_R\_Exponents}(2)} \\ &\quad + \ \texttt{TG\_R\_Constants}(3) * dR^{\texttt{TG\_R\_Exponents}(3)}) \end{aligned}$$

where the temperature at the gradient origin, T_{Origin}, is the BODY namelist temperature. The other parameters are other, optional BODY namelist entries.

In calculating the values at a cell center, the value of dZ is calculated as the cell center's offset in the gradient axis direction from the gradient origin. The value of dR is calculated as the perpendicular offset from the gradient axis to the cell center.

In this problem, we specify the gradient origin at (X,Y,Z)=(-0.5,0,0) and the gradient axis in the X direction (1,0,0). Defaulting the dZ exponents and setting TG_Z_Constants to (-1,0,0) gives linear decrease in X, dropping to zero degrees at X=0.5 (100% decrease in 1 unit length). No variation perpendicular to the axis (dR) is used.

While not used here, upper and lower bounds are also supported in dZ and dR to limit the domain on which the gradient is applied.. See the BODY chapter of the *Reference Manual* for defaults and meanings of all the temperature gradient BODY namelist options.

2.5.2.2 Time Variation of Temperature Boundary Conditions

The second BC namelist is modified to vary the temperature on that boundary (X=0.5) from zero to 100 degrees over 1000 seconds.

```
&BC
    surface_name                = 'conic'
    conic_relation              = '='
    conic_x                     = 1.
    conic_constant              = -0.5
    conic_tolerance             = 1.e-6
    bc_variable                 = 'temperature'
    bc_type                     = 'dirichlet'
    bc_value                    = 0., 1000.0, 100.0
/
```

The only change is in the addition of two values to the BC_VALUE entry. The two additional values represent a (time, value) pair, specifying a new boundary value at the stated time. In this case, we are specifying that the temperature at X=0.5 will be 100 degrees at 1000 seconds into the simulation. TRUCHAS linearly interpolates in time, with the result of a linear increase in the boundary temperature from zero to 100 over the first 1000 seconds of the simulation.

2.5.2.3 Extending the Simulation Time Interval

The final modification is to the OUTPUTS namelist to extend the overall time interval. This is done to allow additional time for equilibration.

```
&OUTPUTS
    output_t                    = 0., 4500.
    output_dt                   = 300.
/
```

2.5.3 Modified Problem Results

This problem starts with a linear temperature gradient in the X direction. The conditions change uniformly over the first 1000 seconds of the simulation. After that time, the temperature equilibrates to a uniform 100 degrees.

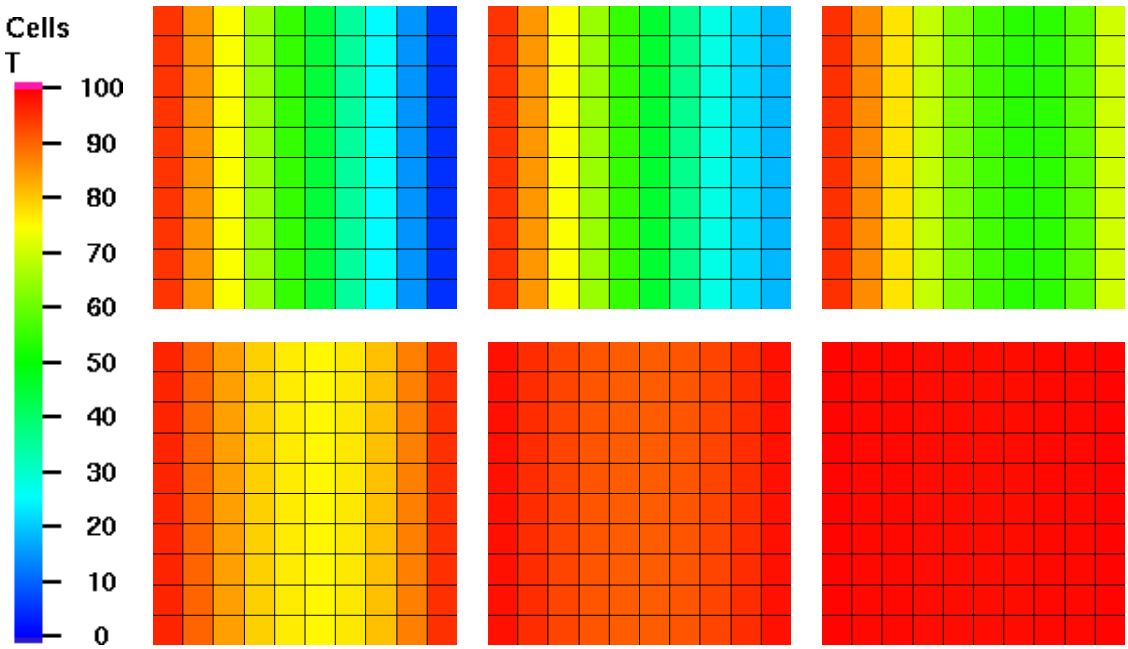
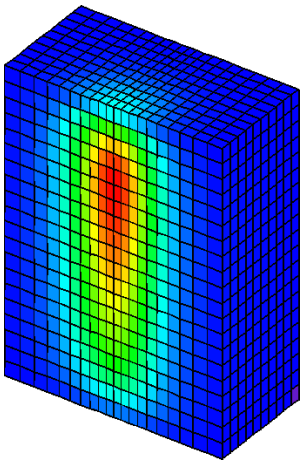


Figure 2.2: Time progression of the temperature profile within the Cu block at times 0, 300 and 900 s (top row) and 1500, 2400, and 4500 s (bottom). The temperature scale is shown on the left, and the view is down the z-axis. As expected, the temperature of the block equilibrates towards a uniform temperature. The temperature is within one degree everywhere by 4500 s.

Chapter 3

Isothermal Phase Change



In this chapter we add an internal heat source, an isothermal phase transition, and a sensitivity analysis to the heat conduction problem discussed in the previous chapter. The inputs discussed for this problem are: definition of isothermal phase changes, importing unstructured meshes, convective heat transfer boundary conditions, design sensitivity parameters, and design sensitivity response functions.

3.1 TRUCHAS Capabilities Demonstrated

- **3D calculation**
- Heat transfer
- **Heat source**

- **Isothermal phase change**
- **Importing unstructured meshes**
- Insulating boundary condition
- **Convective heat transfer boundary condition**
- **Sensitivity Variables**
- **Sensitivity Functions**

3.2 Problem Description

We discuss a heat conduction problem that now includes pure material phase change, namely melting and resolidification. The graph of enthalpy vs. temperature is presented in Figure 3.1. We also introduce: the application of a heat source; and a sensitivity analysis.

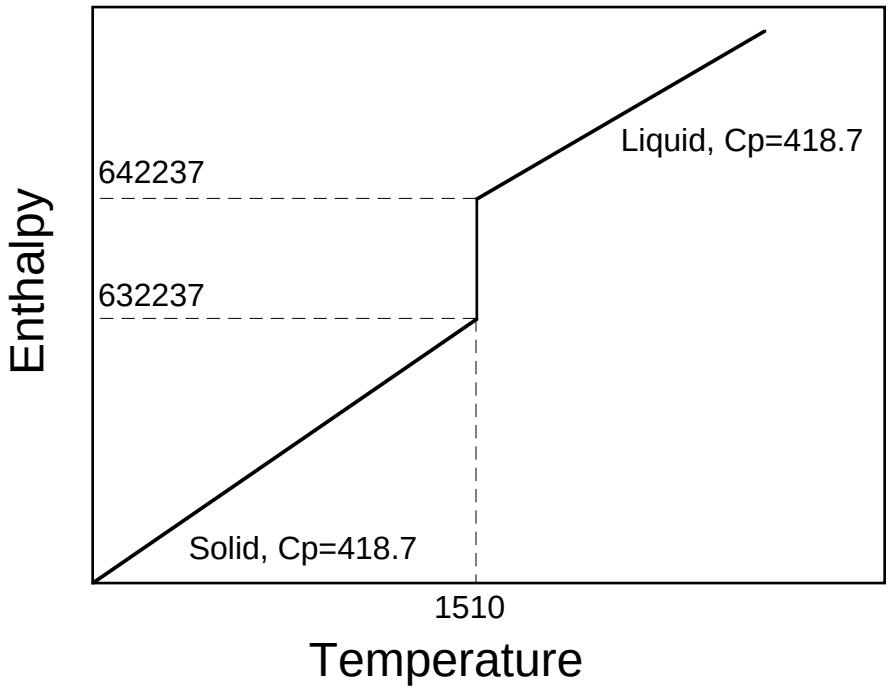


Figure 3.1: Enthalpy vs. temperature for the problem presented in this chapter.

All surfaces of the domain have a convective boundary condition where heat is convected to an ambient temperature of 10.0 degrees Celsius, the exception is the $x = -0.5$ face, which is insulated. The domain is initially at a temperature of 100 degrees Celsius. A volumetric heat source in the shape of an ellipsoid is applied to the domain for a duration of 15 seconds during which time some of the material undergoes melting. At $t = 15$ seconds the heat source is turned off and the material begins to cool and resolidify.

In the sensitivity analysis we determine the effects of changing the boundary conditions. We investigate changes in the ambient temperature and heat transfer coefficient on the top ($z = 1.0$) surface of the domain. We plot the temperature sensitivity and also request the sensitivity values for the temperature at the location $(x, y, z) = (-0.5, 0., 1.0)$, *i.e.* the back/top surface where the heat source is centered.

Units:	SI
Input file:	UMProblems/Chapter_hc_hs_ipc/hc_hs_ipc.inp
Mesh file:	UMProblems/Chapter_hc_hs_ipc/hc_hs_ipc_mesh.txt

3.3 Setup

The complete input file for this problem is presented in Appendix B. Here we discuss sections which are different from those of the previous chapter in detail.

3.3.1 Defining a mesh file

```
&MESH
  mesh_file}        = 'hc_hs_ipc_mesh.txt'
  mesh_file_format  = 'genesis'
/
```

In this problem we read in an external mesh from the file 'hc_hs_ipc_mesh.txt' which has been generated by some external mesh generation software and exported in a 'genesis' text format. Note that we do not have to include any more information about the simulation domain. TRUCHAS deduces this information from the mesh file itself.

For the current problem the domain is $-0.5 < x < 0.5, -1.0 < y < 1.0, -1.0 < z < 1.0$ and decomposed with 20 intervals along the y and z axis and 10 intervals along the x axis. The mesh is orthogonal but not uniform, with higher resolution nearer to the $y = 0$ plane.

3.3.2 Specifying more physics

```
&PHYSICS
  fluid_flow        = .false.
  heat_conduction   = .true.
  phase_change      = .true.
```

```
/
```

The new line here is `phase_change = .true.` This line indicates to TRUCHAS that the materials in this problem undergo at least one phase change.

3.3.3 Defining a heat source; the HEAT_SOURCE namelist

Namelist instances allowed: single

```
&HEAT_SOURCE
  heat_source_type      = 'moving gaussian'
  heat_source_constant  =  1.0e+09
  heat_source_center    = -0.5,  0.0,  1.0
  heat_source_radius    =  0.25, 0.25, 2.0
  heat_source_time      =  0.0, 15.0
  travel_speed          =  0.0,  0.0,  0.0
/
```

This `HEAT_SOURCE` namelist defines an ellipsoid-shaped volumetric heat source with a Gaussian distribution. The *Physics and Algorithms Manual* gives a more detailed description and equations that define the heat source. The entry `heat_source_center` defines the centroid of the ellipsoid at time $t = 0$, and `heat_source_radius` defines the shape of the ellipsoid. For this problem the heat source centroid is located on the edge of the domain, centered in y and at the top of the domain in the z direction. The heat source is defined so that it covers the domain in the z direction , since the third component of `heat_source_radius` = 2.0. The idea of this application is to mimic a welding heat source that fully penetrates the solid material. `travel_speed` defines the velocity of the heat source, which is stationary for this problem. The `heat_source_constant` parameter defines the strength of the heat source, while the `heat_source_time` supplies the start and end times that define the duration of the heat source.

3.3.4 Defining materials

Remember that there must be one `MATERIAL` namelist for each phase of each material in a problem. Thus to TRUCHAS the liquid and solid phases of the same substance are two different materials.

```
&MATERIAL
  Material_Number          = 1
```

```
  Material_Name            = 'liquid'
  Density                  = 7850.0
  cp_constants             = 418.7
  conductivity_constants   = 2570.0
/

\&MATERIAL \\
  Material_Number         =  2
  Material_Name           = 'solid'
  Material_Feature        = 'background'
  Density                 = 7850.0
  cp_constants            = 418.7
  conductivity_constants  = 3070.0
/
```

The two distinct materials are connected via a phase change. The phase change parameters are then defined within a Phase_Change_Properties namelist.

3.3.5 Applying boundary conditions

The conic section method of specifying boundary surfaces is the same as in the first problem, but several wrinkles are added in the type of boundary condition.

```
&BC
  BC_Name          = 'bottom'
  Surface_Name     = 'conic'
  Conic_Relation   = '='
  Conic_Z          = 1.0
  Conic_Constant   = 1.0
  Conic_Tolerance  = 1.0e-5
  BC_Variable      = 'temperature'
  BC_Type          = 'htc'
  BC_Value         = 12e04
  T0_Convection    = 10.0
/
```

Here we introduce a different type of boundary condition, convective heat transfer, that defines how an external surface exchanges heat with its surroundings. `BC_Type = 'htc'` denotes a convective boundary

condition on this surface. The convective boundary condition requires two additional terms, the first being the value of the heat transfer coefficient (hence the acronym `'htc'`), and the second being the ambient temperature. `BC_Value` supplies the value for the heat transfer coefficient and `T0_Convection` defines the ambient temperature. We have also given this boundary condition a name, *viz.* 'bottom.'

If we let q'' be the rate of heat transfer per unit area into the surface defined in the namelist above, then q'' is given as

$$q'' = -\texttt{BC\_Value} * (T - \texttt{T0\_Convection}) \tag{3.1}$$

where T is the temperature at the surface of the domain.

The remainder of the `BC` namelists are similar. See Chapter 2 for a description of the terms not described above.

3.3.6 Selecting output formats

One new option is added in the `OUTPUTS` namelist, directing the code to produce 'short outputs' at each output_dt time. These are diagnostic summary prints that are useful for monitoring runtime progress and state.

```
&OUTPUTS
  output_t                    =  0.0, 45.0
  output_dt                   =  1.0
  XML_output_dt_multiplier    =  1
  short_output_dt_multiplier  =  1
/
```

For details see the `OUTPUTS` chapter of the *Reference Manual*.

3.3.7 Numerics

Here the `NUMERICS` namelist has two parts: physical constraints on the numerical methods themselves; and specification of the relevant solvers.

```
&NUMERICS
  dt_constant                 = 0.40
```

```
  cycle_max                   = 200
  energy_nonlinear_solution   = 'ht_nl_solver'
  sfront_number               = 1.0
  fourier_number              = 0.85
/
```

The `sfront_number` is a limit on the maximum step size of the calculation. A detailed description of this parameter can be found in the `NUMERICS` chapter of the *Reference Manual*. The added difficulty with running combined heat transfer and phase change is that the user could experience some difficulty when the nonlinear solver for the energy equation fails to converge. If the Newton-Krylov solver fails to converge the program will exit with a message of "`FATAL: Newton-Krylov iteration limit!`" Sometimes this problem can be resolved by changing the parameters in the `NONLINEAR_SOLVER` and `LINEAR_SOLVER` namelists, in particular the `maximum_iterations` settings (see the *Reference Manual* for a list of parameters). Often a quicker solution to this problem is to restrict the time step, either by lowering the Fourier number (`fourier_number`) or by setting `dt_maximum` in the `NUMERICS` namelist.

The `cycle_max` number limits the number of time steps the calculation will take to complete.

3.3.8 Nonlinear solvers

```
&NONLINEAR_SOLVER
  name                    = 'ht_nl_solver'
  method                  = 'nk'
  linear_solver_name      = 'gmres-ssor'
  convergence_criterion   =  1.0e-08
  maximum_iterations      =  100
  use_damper              = .false.
  perturbation_parameter  =  1.0e-06
  Damper_Parameters       =  0.8, 1.5, 1.0, 1.0
/
```

Differences between this namelist and the one in the previous chapter are in specific choices of convergence parameters, *e.g.* the addition of an entry for `maximum_iterations`, which was not found to be necessary for the first problem. See Appendices D and E of the *Truchas Physics and Algorithms*for a discussion on determining effective convergence criteria for solvers.

3.3.9 Linear solvers

This namelist is similar to the one in the previous chapter except for the addition of a maximum allowed number of iterations.

3.3.10 Isothermal phase change; PHASE_CHANGE_PROPERTIES namelists

Namelist instances allowed: Multiple, one per phase change.

```
&PHASE_CHANGE_PROPERTIES
  phase_change_active         = .true.
  phase_change_type           = 'isothermal'
  phase_change_model          = 'none'
  hi_temp_phase_material_ID   = 1
  lo_temp_phase_material_ID   = 2
  latent_heat                 = 10000.0
  melting_temperature         = 1510
/
```

Phase changes in TRUCHAS are specified via a PHASE_CHANGE_PROPERTIES namelist. The entries in this namelist specify: that we have a phase change, that it is isothermal and therefore does not require a phase change model, the material numbers of the high and low temperature phase materials, the phase transition temperature (in this case for melting/solidification), and the heat of transformation. Chapter 4 provides an example of a non-isothermal phase change.

For details on the different kinds of phase changes and their corresponding parameters see the PHASE_CHANGE_PROPERTI chapter in the *Truchas Reference Manual*.

3.3.11 Sensitivity variables; SENS_VARIABLE namelists

Namelist instances allowed: multiple, one per sensitivity variable declared.

```
&SENS_VARIABLE
  sens_variable_name         = 'top t0_convection'
  sens_variable_type         = 'BC'
  sens_variable_identifier   = 'top'
```

```
  sens_variable_component    = 't0_convection'
  sens_variable_pert         = 0.1
/
```

In TRUCHAS sensitivities of the response and all sensitivity functions are computed with respect to each of the sensitivity variables specified via the SENS_VARIABLE namelists. Within this namelist:

- the `sens_variable_name` record is the name assigned to the sensitivity variable, *i.e.* `'top t0_convection'`;
- the `sens_variable_type` record identifies the sensitivity variable as a parameter in a BC namelist;
- the `sens_variable_identifier` record is the name of the BC namelist, *i.e.* `'top'`;
- the `sens_variable_component` record indicates the parameter on the identified BC namelist that is the sensitivity variable, *i.e.* `'t0_convection'`; and
- the `sens_variable_pert` record indicates the size of the perturbation, *i.e.* 0.1, for the sensitivity analysis. This value should be approximately 1% of the nominal variable value.

The above namelist invokes a sensitivity analysis in which the sensitivities of the temperature, enthalpy, and volume fraction fields and the sensitivity function are computed with respect the `t0_convection` = 10.0 ambient temperature over the top $z = 1.0$ surface of the block. Sensitivities are written to the XML output file and can be post-processed like any other response field, e.g. temperature. In order to perform the sensitivity analysis, on each time step the residual vector is recomputed with ambient temperature values of `t0_convection` = 10.1 and `t0_convection` = 9.9, and then *one linear* pseudo analysis is performed as described in Appendix E of the *Truchas Physics and Algorithms*.

The second SENS_VARIABLE namelist is reproduced below where it is seen that sensitivities are to be computed with respect to the `bc_value` = 12e04 heat transfer coefficient over the top $z = 1.0$ surface of the block using a `sens_variable_pert` = 120 perturbation.

```
&SENS_VARIABLE
  sens_variable_name         = 'top bc_value 1'
  sens_variable_type         = 'BC'
  sens_variable_identifier   = 'top'
  sens_variable_component    = 'bc_value'
  sens_variable_pert         = 120
/
```

See the SENS_VARIABLE chapter of the *Truchas Reference Manual* for further details.

3.3.12 Sensitivity functions; SENS_FUNCTION namelists

Namelist instances allowed: multiple, one for each sensitivity function we wish to specify.

```
&SENS_FUNCTION
  sens_function_name         = 'T: 2 cells diagonal from BCT'
  sens_function_type         = 'T'
  sens_function_Rparameters  = -0.45, 0.1788230, .75
/
```

The value and sensitivities with respect to the sensitivity variables for each of the response functions defined via the SENS_FUNCTION namelists are computed and output to the XML file so that they may be accessed via a Python script. *Note that sensitivity functions can be defined even if sensitivity variables are not present.* Within this namelist:

- The sens_function_name record is the name assigned to the sensitivity function, *i.e.* 'T: 2 cells diagonal from BCT';
- the sens_function_type record indicates the type of function, *i.e.* T for a cell temperature; and
- the sens_variable_Rparameters record indicates the real parameters that are used to define the function;
- here the $(x, y, z) = (-0.45, 0.1788230, .75)$ coordinates are used to identify the cell with zero sensitivity at t=40 seconds, discussed momentarily.

The second SENS_FUNCTION namelist, replicated below, is used to compute the enthalpy and enthalpy sensitivities for the same cell.

```
&SENS_FUNCTION
  sens_function_name          = 'T: 2 cells diagonal from BCT'
  sens_function_type          = 'H'
  sens_function_Rparameters   = -0.45, 0.1788230, .75
/
```

See the SENS_FUNCTION chapter of the *Truchas Reference Manual* for further details on using the SENS_FUNCTION namelist.

3.4 Results

The next step is to run the problem, using the same TRUCHAS invocation as before but the new input file:

```
% truchas hc_hs_ipc.inp
```

Ensure that the mesh file `hc_hs_ipc_mesh.txt` lies in the current working directory.

Sample results from the analysis are shown in Figure 3.2. The colors are temperature distributions with blue being cold and red being hot. Temperature sensitivities appear in Figure 3.3, where it is seen that an increase in the ambient temperature would result in a higher temperature, whereas an increase in the heat transfer coefficient would result in a lower temperature. These results are expected, as is the fact that the region near the top surface is the most sensitive.

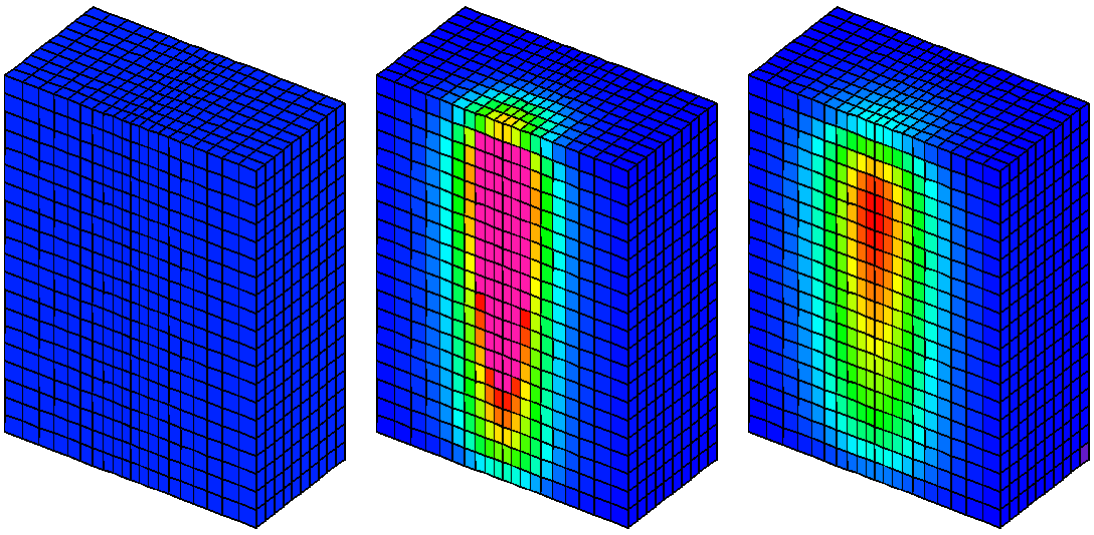


Figure 3.2: Example problem showing a source term that heats a pure material and causes melting. The source term is extinguished at time t=15.0 seconds and the material undergoes cooling and resolidification. Color contours represent the temperature distribution. The snapshots are shown at time t=0, 20, and 40s.

It is interesting to note that two cells near the top center are undergoing a phase transformation at t=40 s, and hence the temperature in these regions and at this time is insensitive to boundary condition perturbations, although the enthalpy (not pictured) is highly sensitive. To investigate this phenomena more thoroughly, `SENS_FUNCTION` namelists are used to monitor the temperature and enthalpy and their sensitivities of the leftmost cell of interest. Figure 3.4 illustrates the time histories of the temperature and temperature sensitivities with respect to the ambient temperature, *i.e.* `'top t0\protect\_convection'`, and the

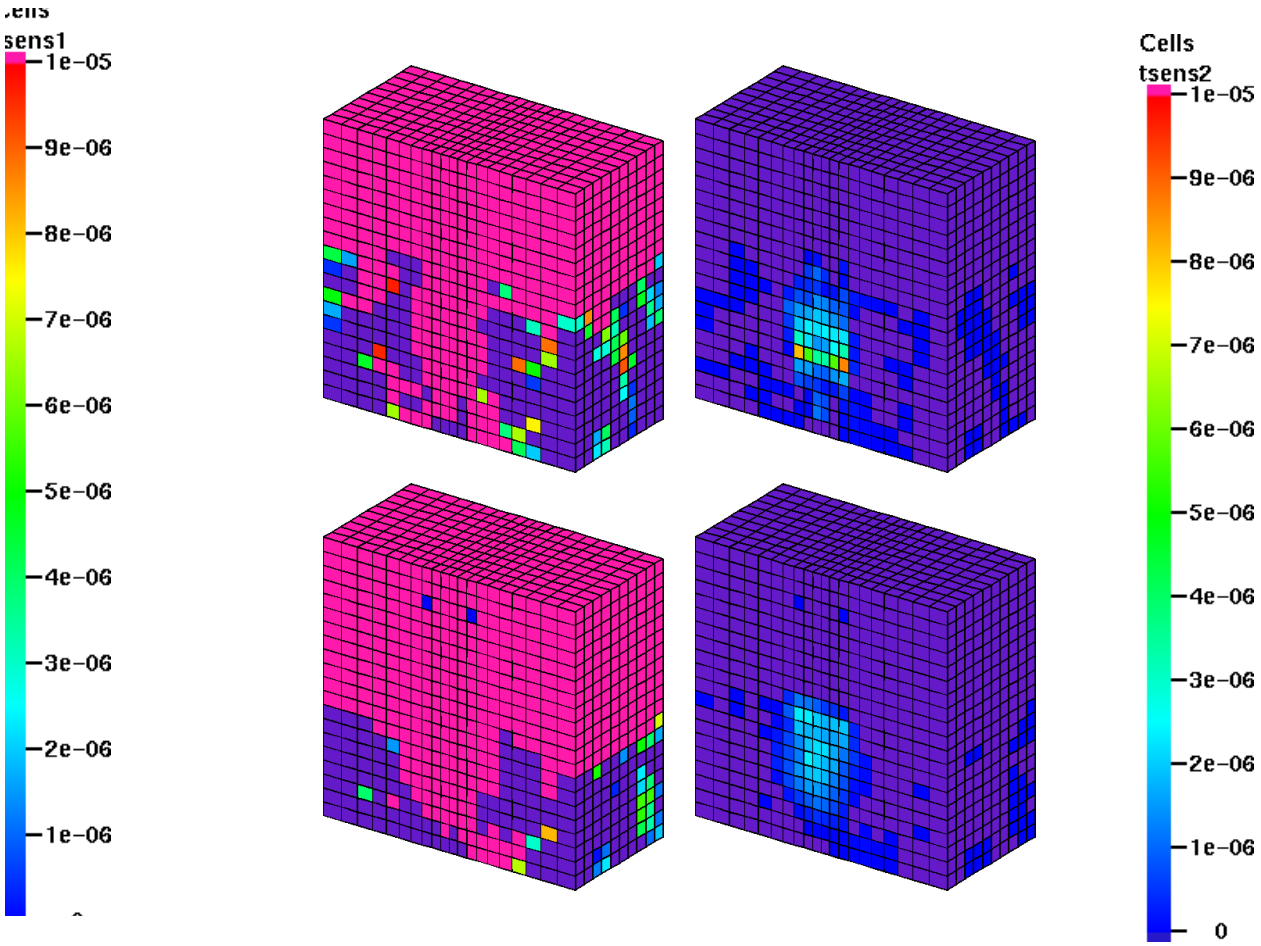


Figure 3.3: Temperature sensitivity contours: top and bottom rows are at times t=20 and 40 seconds and left and right columns are the sensitivities with respect to ambient temperature (variable 1) and heat transfer coefficient (variable 2).

heat transfer coefficient, *i.e.* `'top bc\protect\_value 1'`. Figure 3.5 analogously illustrates time histories of the enthalpy and enthalpy sensitivities. The jumps in the sensitivities are attributed to the fact that two cells adjacent to our cell of interest experience phase transformations at t=30 seconds. For similar reasons sensitivity jumps also occur at t=44 seconds.

–

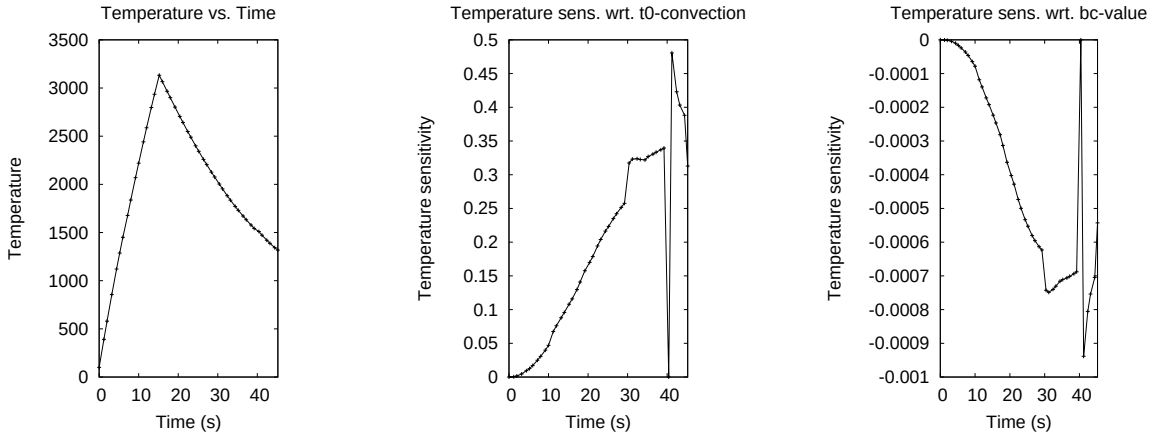


Figure 3.4: Time history of the temperature and temperature sensitivities.

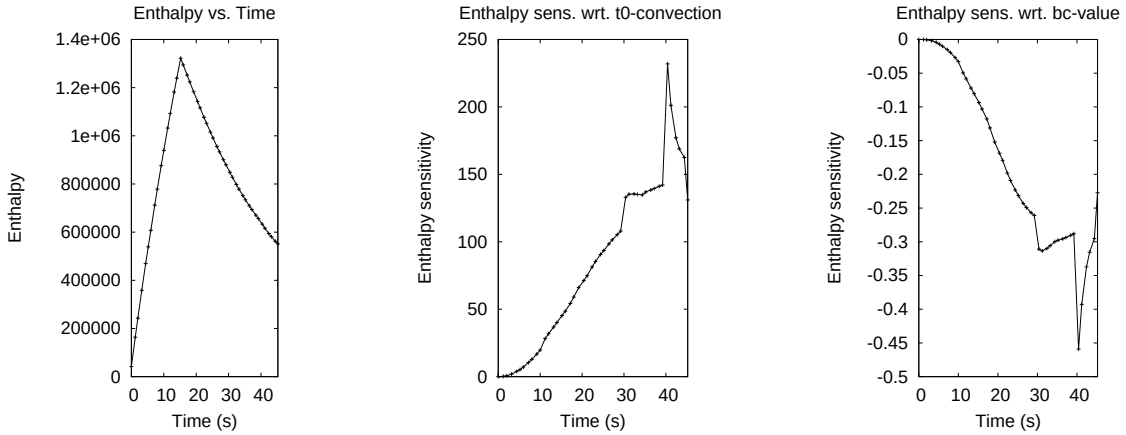
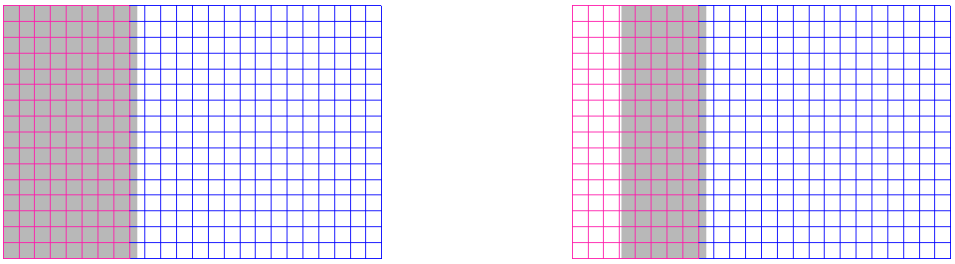


Figure 3.5: Time history of the enthalpy and enthalpy sensitivities.

Chapter 4

Binary Alloy Phase Change



This chapter presents the solidification of a binary alloy following the Scheil model for solidification. New input features discussed in this chapter are parameters for setting up a binary alloy phase transition and partial initialization of a domain using the BODY namelist.

4.1 TRUCHAS Capabilities Demonstrated

- Binary Alloy Phase Change

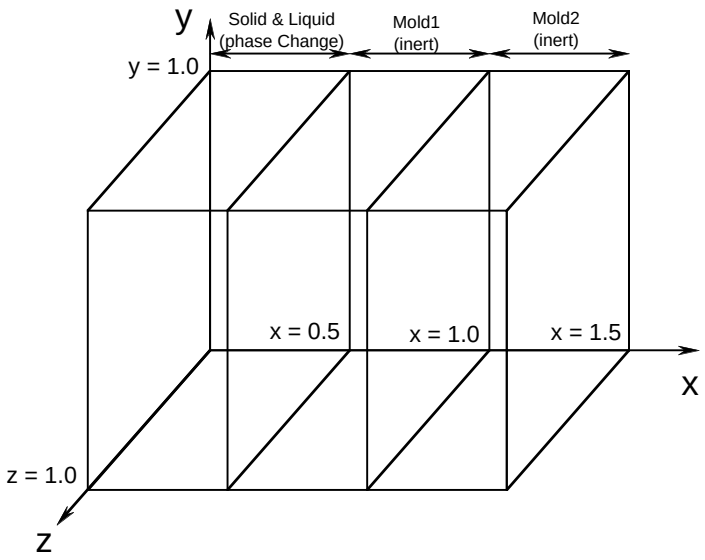


Figure 4.1: Simulation domain for the current chapter. The Mold areas, filled with materials Mold1, and Mold2 are created using a BODY namelist.

- Partial Domain Initialization Using BODY Namelist
- Output of Thermodynamic and Fluid Information

4.2 Problem Description

This chapter demonstrates the setup of a Binary Alloy Phase Change using TRUCHAS. The simulation domain is presented in Figure 4.1. The graph of enthalpy vs. temperature is presented in Figure 4.2, and the phase diagram is presented in Figure 4.3. Initially the temperature in the entire domain is 1000. At time zero we drop the temperature along the left and right edges to 100. The drop in temperature causes the liquid to solidify and a solidification front moves in from the left edge of the problem towards the right. We solve for the progression of the solidification front, and for the temperature profile between the two edges.

Units:	SI
Input file:	UMProblems/Chapter_hc_bapc/hc_bapc.inp
Mesh file:	None

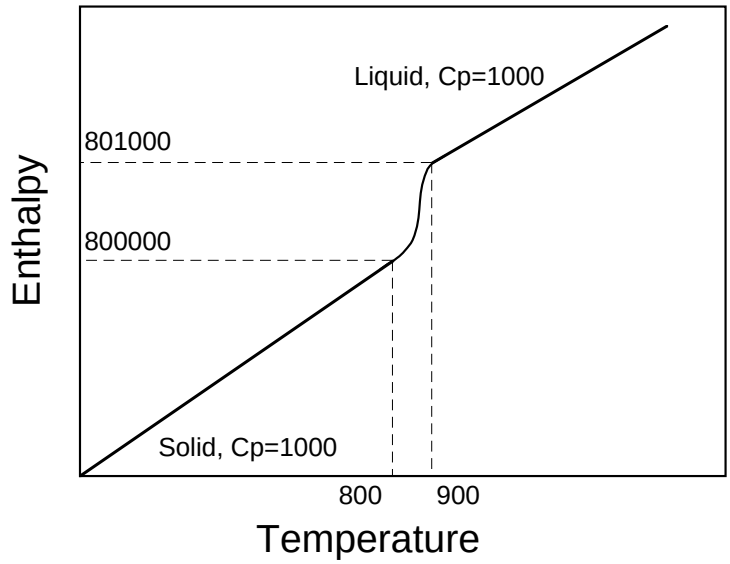


Figure 4.2: Enthalpy Vs. temperature for the problem presented in this chapter.

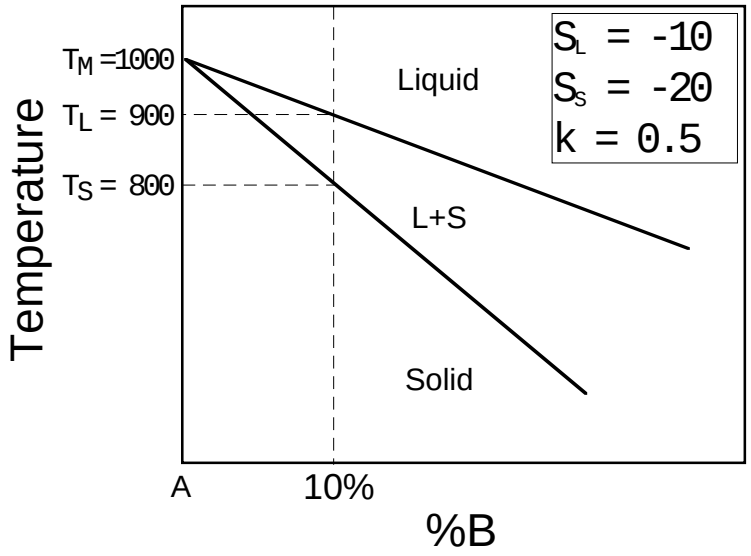


Figure 4.3: Phase diagram for the problem presented in this chapter.

4.3 Setup

The complete input file for this problem is presented in Appendix C. In this chapter we discuss the inputs related to binary alloy phase change and partial domain initialization in detail.

4.3.1 Specifying physics

```
&PHYSICS
  fluid_flow       = .false.
  phase_change     = .true.
  heat_conduction  = .true.
/
```

The new line here, `phase_change = .true.`, directs the code to look for a binary alloy phase change block (Section 4.3.5) and read it in. There is only *one type* of phase change that can be defined at any given time. In the case of alloy phase changes, there is the additional restriction that *only one binary alloy phase change is allowed*. This restriction will be relaxed in a future release of the code.

See Chapter 2 Section 2.3.2 and the `PHYSICS` chapter of the *Truchas Reference Manual* for details.

4.3.2 Defining materials

Material definition follows previous examples. Four materials are defined, `'Liquid'`, `'Solid'`, `'Mold1'`, and `Mold2`, which is made the background material.

4.3.3 Initializing a domain with box surfaces

In this chapter we present a new body type, a 'box'. The second body namelist below converts the block of domain centered at (0.25, 0.5, 0.5) and having extents (0.5, 1.0, 1.0) into material 1 (the liquid) with solute species concentration 10%. The temperature of the box is set to 1000. Although this box is aligned with the cells, in general bodies need not be aligned with the internal mesh. The third namelist converts another section of the domain to material 3. Note that the `'box'` construction can initialize either the inside or the outside of the box. See Chapter 2 Section 2.3.4 and the `BODY` chapter of the *Truchas Reference Manual* for details.

```
&BODY
  surface_name     = 'background'
  material_number  = 4
  temperature      = 1000
/

&BODY
  surface_name     = 'box'
  length           = 0.5 , 1 , 1
  fill             = 'inside'
  translation_pt   = 0.25 , 0.5 , 0.5
  material_number  = 1
  concentration    = 10
  temperature      = 1000
/

&BODY
  surface_name     = 'box'
  length           = 0.5 , 1 , 1
  fill             = 'inside'
  translation_pt   = 0.75 , 0.5 , 0.5
  material_number  = 3
  temperature      = 1000
/
```

4.3.4 Selecting output formats

For this problem we select an additional type of output, the 'long output.' This allows the user to examine results cell by cell without postprocessing. The new keyword `'long_output_dt _multiplier'` tells the code how often to produce these detailed outputs, in units of `'output _dt'`, and the line with `'long_edit_bounding_coords'` sets pairs of coordinates between which the long output will be given.

```
&OUTPUTS
  output_t                    = 0 , 5000
  output_dt                   = 500
  XML_output_dt_multiplier    = 1
  short_output_dt_multiplier  = 1
  long_output_dt_multiplier   = 1
  long_edit_bounding_coords   = 0, 1.5, 0, 1, 0, 1
```

```
/
```

See Chapter 14 and the OUTPUTS chapter of the *Truchas Reference Manual* for details.

4.3.5 Phase Change Parameters

Here we describe the binary alloy phase changes whose parameters are as displayed in Figure 4.2 and Figure 4.3. The Scheil model is employed for the phase change transformation. The latent heat is 1000, and the slope of the liquidus is -10. The partition coefficient is 0.5. Note that the problem is overspecified because the liquidus temperature is specified in this namelist, as well as the concentration in the BODY namelist. If these two numbers are not consistent with each other and the phase diagram, the code will not run.

```
&PHASE_CHANGE_PROPERTIES
  phase_change_active              = .true.
  phase_change_type                = 'alloy'
  phase_change_model               = 'Scheil'
  hi_temp_phase_material_ID        = 1
  lo_temp_phase_material_ID        = 2
  latent_heat                      = 1000.0
  liquidus_slope                   = -10.0
  liquidus_temp                    = 900.0
  melting_temperature_solvent      = 1000.0
  partition_coefficient_constants  = 0.50
  eutectic_temperature             = 0.0
/
```

See the PHASE_CHANGE_PROPERTIES chapter of the *Truchas Reference Manual* for an explanation of these settings and a comprehensive listing of all parameters.

4.4 Results

Sample results from the program plotted using gmv are presented in Figure 4.4.

_

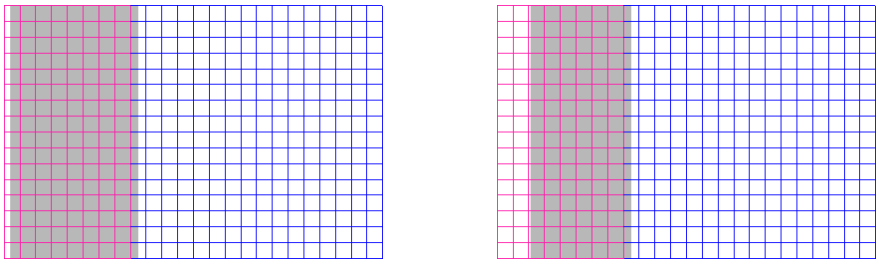
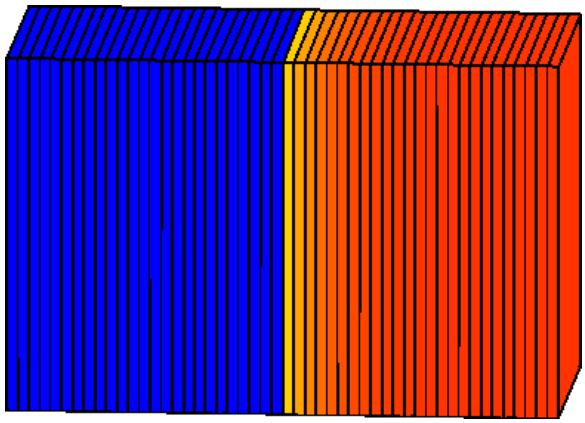


Figure 4.4: Sample results from the calculation at time=500 and 2500s showing progression of solidification of the alloy. The colors show the volume fraction of liquid where blue is 0 and red is 1. The gray areas highlight sections of the domain where the volume fraction of the solid is greater than 0.25, in essence showing the solidification front.

Chapter 5

Chemistry



In this chapter we discuss the capability of TRUCHAS to model chemical reactions along with other physics, in this case heat conduction.

5.1 TRUCHAS Capabilities Demonstrated

- Chemical reaction

5.2 Problem Description

We add chemistry to the heat conduction problem. The left half of the domain is filled with a mold material, and reactants make up the right half of the domain. The reactants are initially at 550°C while the mold material is at 400°C. At time zero, we set the temperature of the left edge of the domain to 100°C and follow the progress of the chemical reaction giving us products from the reactants as a function of time.

Units:	SI
Input file:	UMProblems/Chapter_chem/chem.inp
Mesh file:	None

5.3 Setup

The entire input file for this problem is presented in Appendix D. We will focus on the aspects of this file that define a chemical reaction as a pure material phase transition. This chapter is primarily concerned with defining parameters in the `CHEMICAL_REACTION` namelist.

5.3.1 Defining a chemical reaction

The `CHEMICAL_REACTION` namelist identifies the different chemical reactions that are present during the simulation. Currently only one model is available for chemical reactions, but this list will soon increase. The current model describes a self-catalytic reaction through the following equation:

$$\begin{aligned} \frac{\partial}{\partial t}C(t) &= (k_1 + k_2 * C(t)^m)(C_{max} - C(t))^n \\ k_1 &= k_1^o * e^{-E_1^a/RT} \\ k_2 &= k_2^o * e^{-E_2^a/RT} \end{aligned} \tag{5.1}$$

where C is concentration of the product, t is time, k_1^o and k_2^o the reaction constants for the two stages and are both a function of temperature, T. The order of the reaction is given by $n + m$. R is the gas constant. It is further assumed that the heat of reaction, H_{tot} is evenly distributed as the reaction progresses. i.e.

$$\frac{\partial H}{\partial t} = H_{tot}\frac{\partial C}{\partial t} \tag{5.2}$$

NOTE: Because R is defined in SI units, the temperature must be the absolute temperature (K), the activation energies, E_1^a and E_2^a, should be in consistent units (J), and the the time and rates k_1, k_2, should be consistent.

```
&CHEMICAL_REACTION
  cr_m               = 1,
  cr_n               = 1,
  cr_reactants_ID    = 2,
  cr_products_ID     = 1,
  cr_Cmax            = 0.99,
  cr_ko1             = 1.0e5,
  cr_ko2             = 0.0
  cr_Ea1             = 7.0e4,
  cr_Ea2             = 0.0,
  cr_Htot            = 2000
/
```

The different variables in the `CHEMICAL_REACTIONS` namelist are named as in equation Equation 5.1. For this particular set of parameters, the equation becomes:

$$\begin{aligned} \frac{\partial}{\partial t}C(t) &= (k_1 + k_2 * C(t)^1)(0.99 - C(t))^1 \\ k_1 &= 10^5 * e^{-7.0*10^4/8.312T} \\ k_2 &= 0.0 \end{aligned} \tag{5.3}$$

where C is the concentration of material 1, t is time, and T is temperature.

For additional details see the `CHEMICAL_REACTIONS` chapter of the *Truchas Reference Manual*.

5.3.2 Specifying physics

This namelist is the same as for a phase change problem.

5.3.3 Defining materials

Setting up materials that react is similar to setting up a material with more than one phase.

```
&MATERIAL
  material_name            = 'Product'
  material_number          = 1
  conductivity_constants   = 0.31
  density                  = 30.
  Cp_constants             = 100.
/

&MATERIAL
  material_name            = 'Reactant'
  material_number          = 2
  material_feature         = 'background'
  conductivity_constants   = 0.31
  density                  = 30.
  Cp_constants             = 100.
/

&MATERIAL

  material_name            = 'Mold'
  material_number          = 3
  conductivity_constants   = 0.1
  density                  = 30.
  Cp_constants             = 10.
/
```

5.3.4 Initializing the domain

For this problem we again use a box to initialize part of the domain.

```
&BODY
  material_number   = 3,
  surface_name      = 'box',
  fill              = 'inside',
  temperature       = 400.,
```

```
  translation_pt      =0.25, 0.50, 0.50,
  length             = 0.50, 1.0, 1.0
/

&BODY
  surface_name       = 'background',
  fill               = 'nodefault',
  material_number    = 2,
  temperature        = 550.
/
```

5.3.5 Linear Solvers

In this file only one linear solver, 'SSOR-GMRES', is referenced, so only one linear solver namelist exists. For this problem it is found that there is benefit in choosing a stopping criterion for the solver different from the default, `'||r||'`.

```
&LINEAR_SOLVER
  name                     = 'SSOR-GMRES',
  method                   = 'GMRES',
  convergence_criterion    = 1.e-4
  preconditioning_method   = 'SSOR',
  relaxation_parameter     = 0.9,
  preconditioning_steps    = 3,
  stopping_criterion       = '$||r||/||r0||$'
/
```

See Chapter 2 Section 2.3.9 and the `LINEAR_SOLVER` chapter of the *Truchas Reference Manual* for a comprehensive listing of all parameter settings, as well as recommendations for choice of stopping criteria for different solver applications in TRUCHAS.

5.4 Results

Results of running TRUCHAS with this input are shown in Figure 5.1. Plotted is the variation in concentration of products as a function of time. Note that the reaction does not progress as fast near the mold as it does in the rest of the domain. This is because the mold is held at a lower temperature than the reactant materials and in addition is in contact with a heat sink at 100°C along its left edge.

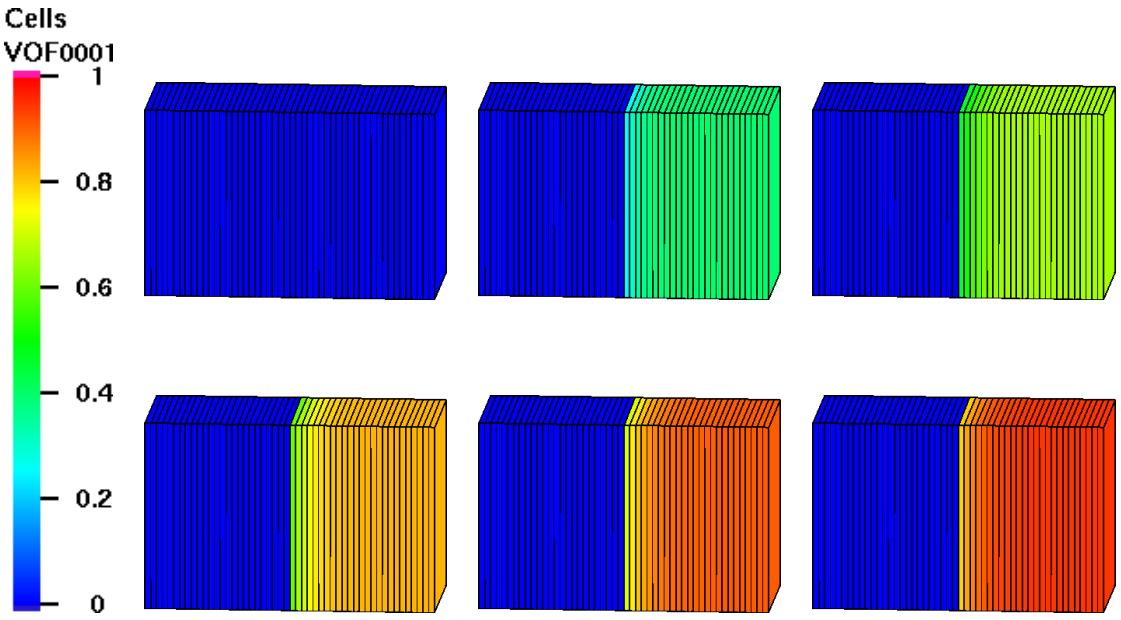
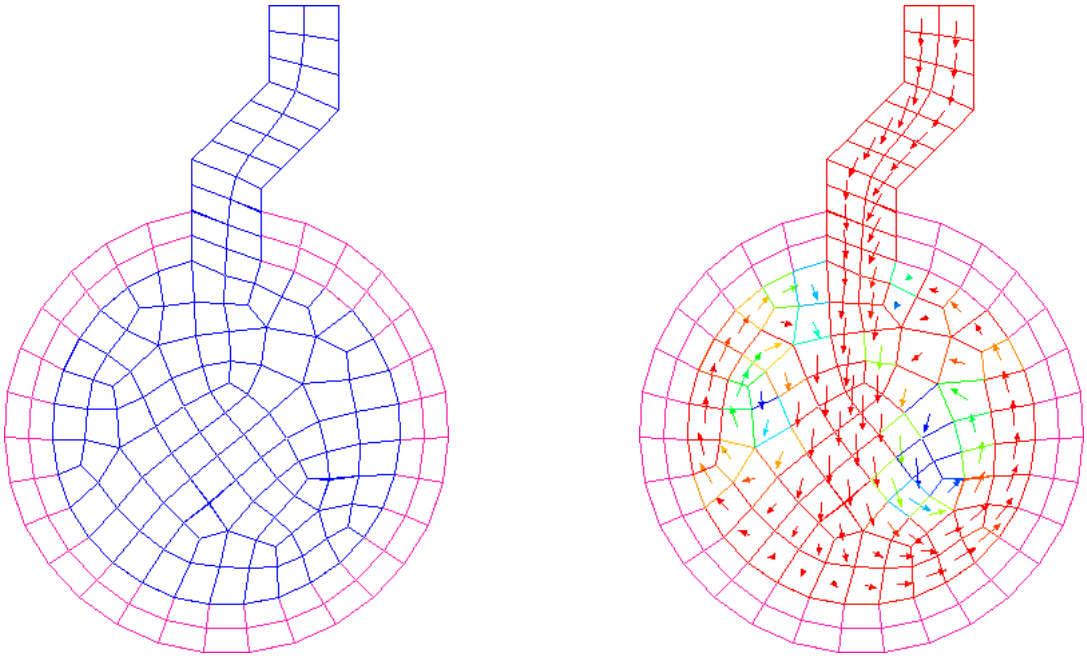


Figure 5.1: Time progression of concentration of Material ID 1. The concentration scale is shown on the left. As is expected, the progression of the reaction for reactants in contact with the mold (along the left edge) is slower than that in the rest of the domain.

Chapter 6

Flow and Filling



We now change gears and introduce flow, a feature that is independent of heat transfer. We demonstrate how to set up the filling of a cylindrical mold. Sections of the input file that are discussed in this chapter are additions to the physics capabilities, *viz.* assigning material IDs to materials read from an imported mesh file, an inflow boundary condition, and a variable time step.

6.1 TRUCHAS Capabilities Demonstrated

- Flow
- Assigning Material IDs to Mesh Materials
- Scaling an externally generated mesh
- Inflow Boundary Condition
- Variable Time Step

6.2 Problem Description

In this chapter we demonstrate a problem where a cylindrical object is filled with a fluid. The cylinder is initially filled with void. Fluid enters from an inlet through a boundary condition that sets a velocity at which material flows into the mesh domain through a boundary, called an inflow boundary condition.

One important point to note is that currently TRUCHAS cannot stop a fill boundary condition after either a certain volume of fluid has entered the domain or after a given amount of time has elapsed. One has to stop the simulation, change the boundary condition in the input file, and restart the simulation with new boundary conditions to achieve this. This will change in an upcoming release of the code.

Units:	CGS
Input file:	UMProblems/Chapter_flow/flow.inp
Mesh file:	UMProblems/Chapter 6/flow_mesh.txt

6.3 Setup

The input file for this problem is presented in Appendix E. In this chapter we discuss variables related to flow as well as boundary conditions related to filling.

6.3.1 Defining a mesh

```
&MESH
  mesh_file               = 'flow_mesh.txt'
  mesh_file_format        = 'genesis'
  coordinate_scale_factor  =  100.
/
```

The new line in this file is `coordinate_scale_factor = 100` which directs TRUCHAS to scale all coordinates in the input mesh file `flow_mesh.txt` by a factor of 100. In this case, the mesh was generated using SI units whereas this calculation is set up using CGS units. The scale factor of 100 converts meters to centimeters. See the `MESH` of the *Truchas Reference Manual* for a comprehensive listing of all fields that can occur in this namelist.

6.3.2 Specifying physics

Flow is the only type of TRUCHAS physics that is on by default. In all the preceding examples with heat conduction physics turned on, flow physics needed to be explicitly set to `.false.` to prevent it being on. In this problem we explicitly set fluid flow on, whereas we need not switch heat conduction off.

```
&PHYSICS
  fluid_flow  = .true.
  body_force  =  0.0, -981.0, 0.0
  inviscid    = .true.
/
```

The `body_force` invokes a force due to gravity. We also turn on inviscid flow capabilities. See the `PHYSICS` chapter of the *Truchas Reference Manual* for further details on these variables.

6.3.3 Defining materials

```
&MATERIAL
  material_number   = 1
  material_name     = 'water'
  priority          = 1
  density           = 1.
/

\&MATERIAL
  material_number   = 2
  material_name     = 'void'
  priority          = 2
  density           = 0.
  material_feature  = 'background'
/
```

```
\&MATERIAL
  immobile          = .true.
  material_name     = 'mold'
  material_number   = 3
  priority          = 3
  density           = 1000
/
```

The `'immobile = .true.'` setting for material 3 indicates that for the purpose of flow calculations, this material does not move; only materials 1 and 2 flow. The `'priority'` fields in the materials namelists define the priorities with which we deal with materials when computing flow. See Chapter 2 Section 2.3.4 and the `MATERIAL` chapter in the *Truchas Reference Manual* for details on these variables.

6.3.4 Initializing the domain

The initialization of the domain in this chapter is done differently from the previous chapters. Here we read in a mesh file (see Section 6.3.1), and assign block IDs in the mesh file to regions of the TRUCHAS domain that can then be assigned material numbers. This mechanism can only assign entire mesh blocks uniformly to single (possibly mixed) materials.

```
&BODY
  surface_name          = 'from mesh file'
  mesh_material_number  = 1
  material_number       = 2
  temperature           = 0.
/

&BODY
  surface_name          = 'from mesh file'
  mesh_material_number  = 2
  material_number       = 3
  temperature           = 0.
/
```

A line `surface_name = 'from mesh file'` indicates that we have to initialize the body from a block in the mesh file. One examines the mesh file and identifies the block number to which to assign the current material. On the line for `mesh_material_number` this block ID is on the right hand side, and

presence of a `material_number` assignment in the same namelist completes the assignment of a material number to the mesh block. See the `BODY` chapter of the *Truchas Reference Manual* for further options.

6.3.5 Applying boundary conditions

An important difference exists in boundary conditions when flow is run without heat conduction. Whereas heat conduction solutions require some temperature boundary condition to be set on every external surface of the computational domain—and there is no default condition— flow can safely assume a default free-slip velocity boundary condition on external surfaces. Therefore in this problem only the inflow boundary condition on one surface needs to be explicitly set. All other surfaces are implicitly free-slip by omission of BC namelists to set them otherwise.

```
&BC
  surface_name         = 'conic'
  conic_relation       = '='
  conic_y              =  1.0
  conic_constant       =  -12.2
  conic_tolerance      = 1.0e-6
  bounding_box         = -6.5, 6.5, 12.18, 12.22, -10, 10
  BC_variable          = 'velocity'
  BC_type              = 'Dirichlet'
  BC_value             = 0.0, -1000., 0.0
  inflow_material      = 1
  inflow_temperature   = 0.0
/
```

Here we apply a boundary condition on the surface determined by the equation $Y = 12.2$ that lies within the bounding box with corners (-6.5, 12.18, -10) and (6.5, 12.22, 10). Note that the coordinates of the bounding box are specified as x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}. The advantage of using a `bounding_box` is that we can now define different boundary conditions on different sections of a single surface. The input is defined with a constant velocity boundary condition with velocity of -1000 cm/s in the y direction. The material that enters has ID 1 and zero temperature. See Chapter 2 Section 2.3.5 and the `BC` chapter of the *Truchas Reference Manual* for all types of boundary conditions.

6.3.6 Numerics

TRUCHAS flow physics has a considerable number of controls that must be understood in order to use the code optimally. In the `NUMERICS` namelist we define a variable time step that can grow or shrink

automatically in response to changing flow conditions, and we provide direction to the code on how to solve the flow problem. The reader should peruse the material in the *Truchas Physics and Algorithms* to learn the variety of flow methods installed in TRUCHAS. We discuss a few of the options below.

```
&NUMERICS
  dt_init                    = 1.e-4
  dt_grow                    = 1.05
  dt_min                     = 1.e-7
  dt_max                     = 1.
  discrete_ops_type          = 'ortho'
  projection_linear_solution = 'projection-solver'
  Courant_number             = 0.4
  volume_track_interfaces    = .true.
  volume_track_Brents_method = .true.
  volume_track_iter_tol      = 1.0e-12
  cutVOF                     = 1.0e-08
/
```

The initial time step is 10^{-4}s, and this is allowed to grow at most by the ratio given in `dt_grow`. The limits on the smallest and largest time steps in the simulation are given by `dt_min` and `dt_max` respectively. For incompressible flow another type of solver, projection, must be defined. The Courant number, defined in the *Truchas Physics and Algorithms*, is the basic flow constraint. It must be less than 1, and the default is 0.5, so the choice here is conservative.

Volume tracking is the method by which TRUCHAS and many other multimaterial simulation codes keep track of more than one material in a computational cell. Our method is known as a 'volume of fluid' or VOF method, and `cutVOF` is a limit for a tiny fraction of a cell in a material, below which it will be "cut" from the cell and no longer tracked. Technical details are in the *Truchas Physics and Algorithms* and an explanation of all `NUMERICS` namelist options is given in the *Truchas Reference Manual*.

6.3.7 Linear solvers

```
&LINEAR_SOLVER
  name                   = 'projection-solver'
  method                 = 'GMRES'
  preconditioning_method = 'SSOR'
  preconditioning_steps  = 2
  relaxation_parameter   = 1.4
  convergence_criterion  = 1.e-8
```

```
  maximum_iterations      = 50
/
```

This namelist specifies the solver for the projection solution and is similar to previous examples. See Chapter 2 Section 2.3.9 and the `LINEAR_SOLVER` chapter of the *Truchas Reference Manual* for a comprehensive listing of all possible variables.

6.4 Results

Sample results from the program plotted using gmv are presented in Figure 6.1.

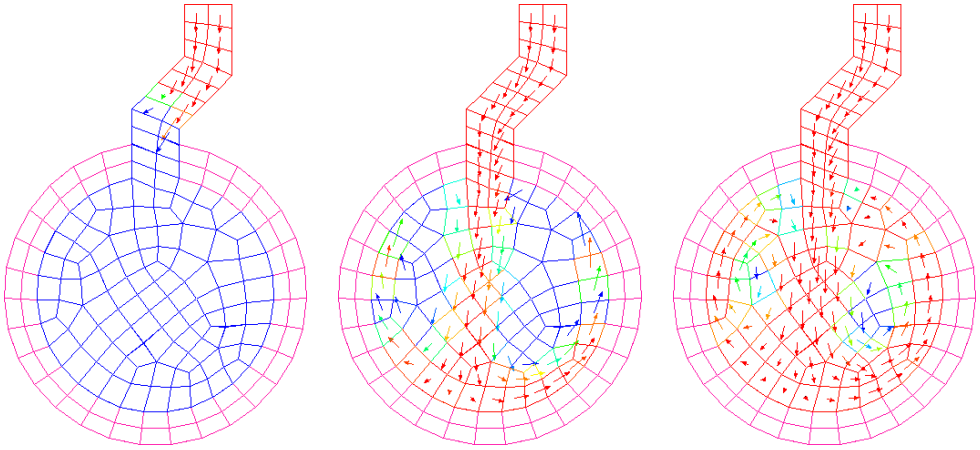
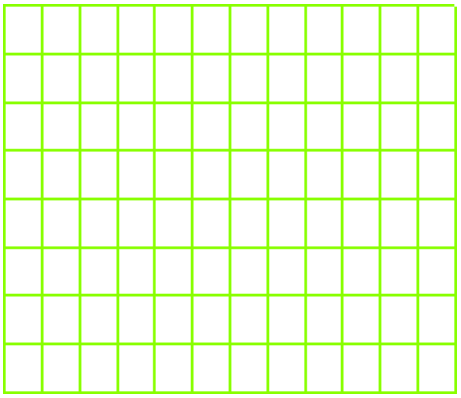


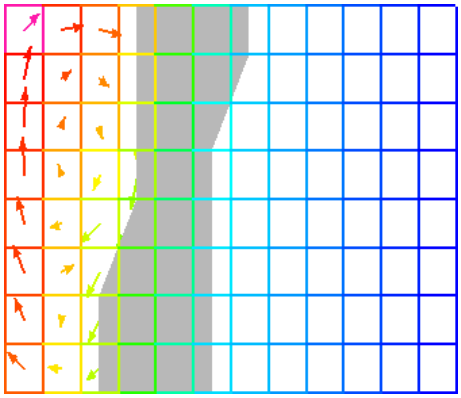
Figure 6.1: Sample results at 0.0, 0.025 and 0.04 s. The figures are colored by density with pink representing the mold materials, dark blue representing void and colors between dark blue and red representing the liquid. The arrows indicate the velocity of liquid.

–

Chapter 7

Multiple Phase Changes + Buoyancy-Driven Flow





This chapter combines the heat transfer, phase change and flow capabilities in a problem that computes buoyancy-driven flow. New features introduced in this chapter are definitions of multiple phase changes and temperature-dependent properties that induce buoyancy-driven flow.

7.1 TRUCHAS Capabilities Demonstrated

- Temperature-Dependent Density
- Multiple Isothermal Phase Changes

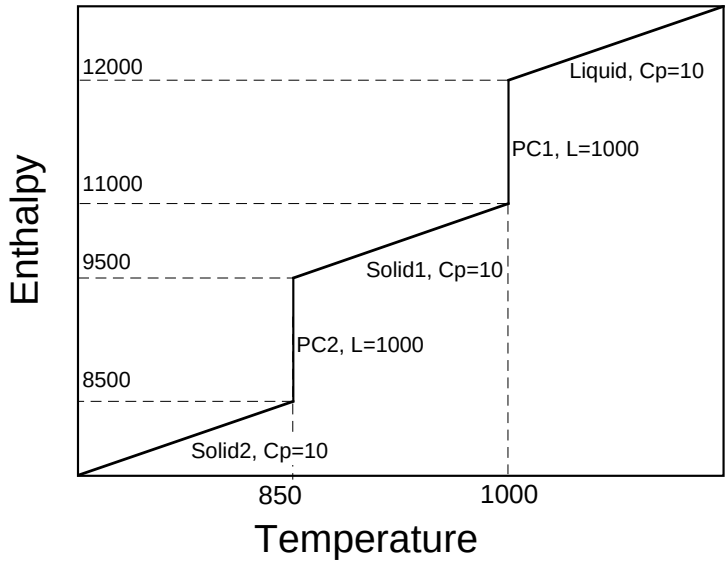


Figure 7.1: Enthalpy Vs. Temperature for the problem presented in this chapter.

7.2 Problem Description

This chapter demonstrates multiple isothermal phase changes along with buoyancy-driven flow. The graph of enthalpy vs. temperature is presented in Figure 7.1. Initially the temperature in the entire domain is 1010, and it is liquid. At time zero we drop the temperature along the right edge of the domain to 700 while at the same time raising the temperature along the left edge to 1200. At steady state, the left side of the domain will be liquid, solid 1 will be in the middle, and the right side of the domain will be solid 2. Due to flow driven by buoyancy, the solidification front will not be straight, but will have a lip at the bottom.

Units:	SI
Input file:	UMProblems/Chapter_hc_mipc_flow/hc_mipc_flow.inp
Mesh file:	None

7.3 Setup

The input file for this problem is presented in Appendix F. In this chapter we discuss the variables related to setting up multiple phase changes and temperature-dependent density in detail.

7.3.1 Specifying physics

In this problem for the first time we have fluid flow, heat conduction, and phase change physics active at the same time. This is the most complex situation we have treated.

```
&PHYSICS
  fluid_flow       = .true.
  phase_change     = .true.
  heat_conduction  = .true.
  body_force       =  0.0, -9.81, 0.0
  inviscid         = .true.
/
```

By default, the Boussinesq approximation is active, as explained in the *Truchas Physics and Algorithms*. See Chapter 3 Section 3.3.2 and Chapter 6 Section 6.3.2 for details on the rest of the variables. Although this should really be a viscous problem to get buoyancy-driven flow correct, we run it here as an inviscid problem for demonstration purposes, because the viscous problem runs much more slowly.

7.3.2 Defining materials

A new feature introduced here is the temperature-dependent density for material 1, the liquid. This temperature dependence is given as a polynomial with the following form:

$$\rho = \rho_0 * (1 + \alpha * (T - T_{ref})^n).$$

Using the parameters for the liquid $n =$ `density_change_exponents_T = 1.`, gives us:

$$\rho = 5000 * (1 + -10^{-5} * (T - 1000)).$$

```
&MATERIAL
  material_name                =’liquid’
  material_number              = 1
  material_feature             = ’background’
  priority                     = 1
  density                      = 5000
  density_change_relation      = ’temperature_polynomial’
  density_change_coefficients_T  = -1.e-5
  density_change_exponents_T     = 1.
```

```
  reference_temperature          = 1000.
  conductivity_constants         = 250
  Cp_constants                   = 10
/

&MATERIAL
  immobile                = .true.
  material_name           = 'solid1'
  material_number         = 2
  priority                = 2
  density                 = 5000
  conductivity_constants  = 150
  Cp_constants            = 10
/

&MATERIAL
  immobile                = .true.
  material_name           = 'solid2'
  material_number         = 3
  conductivity_constants  = 1000
  density                 = 5000
  Cp_constants            = 10
  priority                = 3
/
```

See the `MATERIAL` chapter of the *Truchas Reference Manual* for further details on temperature functions for material properties.

7.3.3 Initializing the domain

Here we are again using a block mesh, internally generated, and we initialize it with a fill of `'nodefault'`, because the entire domain starts filled with our material.

```
&BODY
  surface_Name     = 'background'
  fill             = 'nodefault'
  material_Number  = 1
  temperature      = 1010
/
```

See the BODY chapter of the *Truchas Reference Manual* for details.

7.3.4 Applying boundary conditions

In this problem we apply only temperature boundary conditions, yet flow will be induced. In Appendix F it will be seen that we apply a constant temperature BC on the two faces of our box domain that are perpendicular to the x axis, and zero flux BCs on the other four faces. This will induce convective circulation around the box as well as phase changes.

7.3.5 Numerics

For this problem we must specify solvers for both the energy and projection solutions. Other items in this namelist are similar to previous problems.

```
&NUMERICS
  dt_init                    = 1.e-3
  dt_grow                    = 1.05
  dt_min                     = 1.e-7
  dt_max                     = 2.e0
  discrete_ops_type          = 'ortho'
  energy_nonlinear_solution  = 'NK-energy'
  projection_linear_solution = 'projection-solver'
  Courant_number             = 0.4
  volume_track_interfaces    = .true.
  volume_track_Brents_method = .true.
  volume_track_iter_tol      = 1.0e-12
  cutVOF                     = 1.0e-08
/
```

See Chapter 6 Section 6.3.6 and the NUMERICS chapter of the *Truchas Reference Manual* for a comprehensive listing of all possible variables.

7.3.6 Phase change parameters

The only difference between this and the input in Chapter 3, Section 3.3.10 is that we need to define two phase changes. The fields within the namelists are identical in both cases. Figure 7.1 shows the graph of

enthalpy vs. temperature for these two phase changes.

```
&PHASE_CHANGE_PROPERTIES
  phase_change_active        = .true.
  phase_change_type          = 'isothermal'
  phase_change_model         = 'none'
  hi_temp_phase_material_ID  = 1
  lo_temp_phase_material_ID  = 2
  latent_heat                = 1000.0
  melting_temperature        = 1000.0
/

&PHASE_CHANGE_PROPERTIES
  phase_change_active        = .true.
  phase_change_type          = 'isothermal'
  phase_change_model         = 'none'
  hi_temp_phase_material_ID  = 2
  lo_temp_phase_material_ID  = 3
  latent_heat                = 1000.0
  melting_temperature        = 850.0
/
```

7.4 Results

Sample results from the program plotted using GMV are presented in Figure 7.2. The upper figure is a transient state at 25 s, while the lower figure is the steady state solution with liquid on the left, solid1 in the middle (in gray), and solid2 on the right.

–

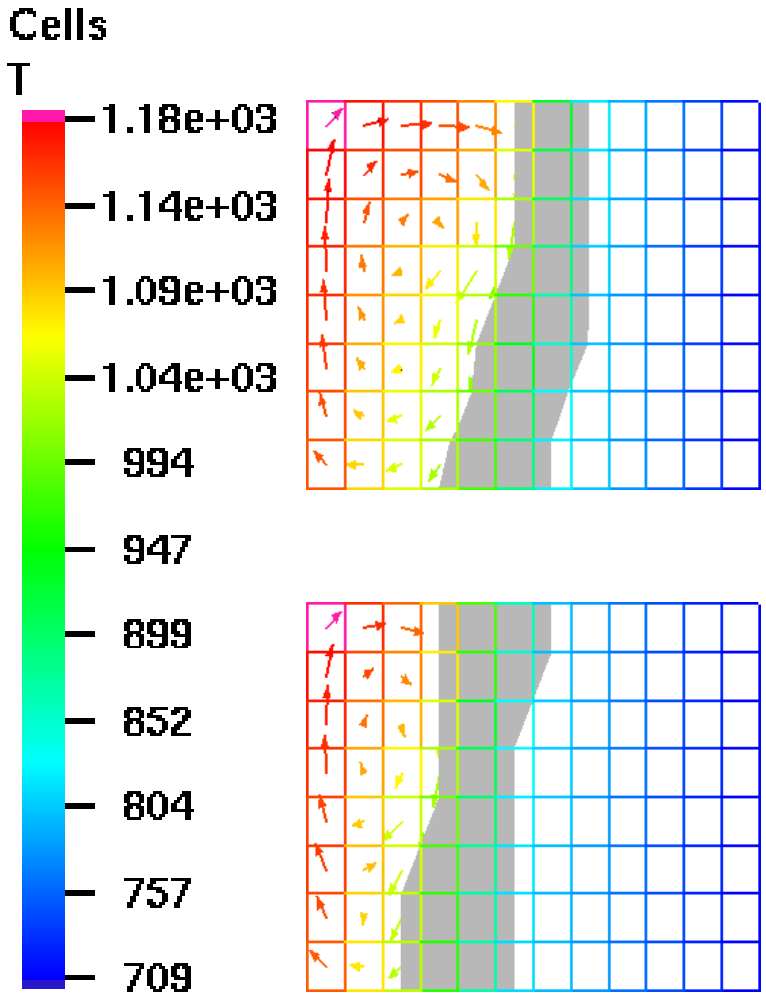
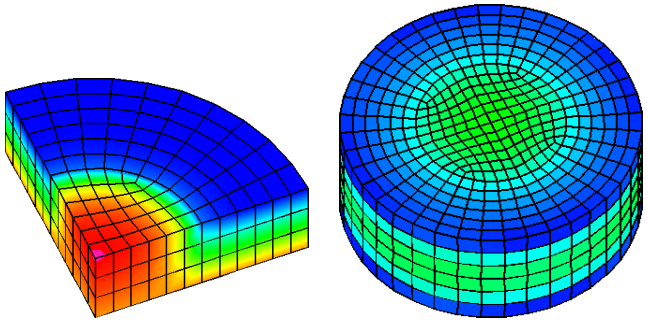


Figure 7.2: Sample results from the calculation at time=100 and 400 s showing the progression of solidification. The colors show the temperature, the arrows indicate the flow direction in cells that contain liquid, and the gray areas highlight sections of the domain where the volume fraction of solid1 is greater than 0.25, in essence showing the two solidification fronts.

Chapter 8

Elastic and Viscoplastic Deformation



In this chapter we set up a simple problem to demonstrate the solid deformation capability of TRUCHAS. The problem uses heat conduction (see Chapter 2) and introduces input for solid mechanics, including new material property input, boundary conditions, and solver input specific to the solid mechanics solution.

8.1 TRUCHAS Capabilities Demonstrated

- Solid mechanics
- Elastic stress and strain
- Viscoplastic strain

8.2 Problem Description

The problem consists of a ring of aluminum alloy in contact with a center plug of graphite. The outer diameter of the aluminum ring is 101.6 mm and the diameter of the plug (and inner diameter of the ring) is 50.8 mm. The plug and ring are 12.7 mm thick. Both materials have their stress-free reference temperature set to 630 degrees and both bodies have an initial temperature of 455 degrees. An initial elastic stress field is calculated, driven by the difference in the coefficients of thermal expansion for the two materials. Initial conditions were chosen to induce stresses that will produce plastic flow in the aluminum ring near the interface with the graphite.

The ring/plug assembly is allowed to cool with convective heat transfer boundary conditions on the top and bottom. Stresses and strains evolve with time due to the changing temperature field and viscoplastic deformation in the aluminum. There is no relative movement at the interface between the aluminum and graphite.

Units:	SI
Input file:	UMProblems/Chapter_vp/vp.inp
Mesh file:	UMProblems/Chapter_vp/vp_mesh.txt

8.3 Setup

The complete input file for this problem is presented in Appendix G. Here we discuss previously unseen sections of the input file in detail.

8.3.1 Defining a mesh

```
&MESH
  mesh_file          ='vp_mesh.txt'
  mesh_file_format   ='genesis'
/
```

The `MESH` namelist reads a file created by CUBIT and translated to Genesis text format. In addition to element blocks, this input file contains "side sets." As defined by CUBIT, these are sets of contiguous or non-contiguous surfaces that can be grouped, written into the mesh file, and read by TRUCHAS. Side sets can then be referred to by side set number in the input BC namelists as a convenient way to specify which surfaces shall acquire a boundary condition.

For this problem the solution should be axisymmetric, and the domain represented by the mesh is only 1/8 of the full size ring and plug. Symmetry planes are modeled by specifying appropriate boundary conditions.

8.3.2 Specifying physics

This problem activates only heat conduction and solid mechanics.

```
&PHYSICS
  fluid_flow        = .false.
  heat_conduction   = .true.
  solid_mechanics   = .true.
  phase_change      = .false.
/
```

8.3.3 Defining materials

Now we must give our materials elastic-plastic physical properties. The graphite material is elastic, while the aluminum is elastic-viscoplastic. Currently the Mechanical Threshold Stress (MTS) model or a simple power law creep model are the only choices for a plasticity model. Other choices may be added in future releases. More information about the viscoplastic models model is in the *Truchas Physics and Algorithms* and *Truchas Reference Manual*. In order to calculate elastic stresses, it is necessary to specify elastic constants (`Lame1_constants` and `Lame2_constants`), the coefficient of thermal expansion, and the stress reference temperature. If `viscoplastic _model` is not equal to `'elastic_only'`, then viscoplastic model parameters must be specified.

```
&MATERIAL
  immobile                      = .true.
  material_number               = 1
  priority                      = 1
  material_name                 = 'graphite',
  density                       = 1.7e+03
  Cp_constants                  = 1.925e+03
  conductivity_constants        = 1.95e+02
  Lame1_constants               = 3.4e+9
  Lame2_constants               = 2.76e+9
  CTE_constants                 = 7.0e-06
  stress_reference_temperature  = 6.30e+02
```

```
  material_feature                = 'background'
  viscoplastic_model              = 'elastic_only'
/

&MATERIAL
  immobile                        = .true.
  material_number                 = 2
  priority                        = 2
  material_name                   = '5754 aluminum'
  density                         = 2.70e+03
  Cp_constants                    = 9.00e+02
  conductivity_constants          = 2.40e+02
  Lame1_constants                 = 5.20e+10
  Lame2_constants                 = 2.60e+10
  CTE_constants                   = 2.20e-05
  stress_reference_temperature    = 6.30e+02
  viscoplastic_model              = 'MTS'
  MTS_k                           = 1.38e-23
  MTS_mu_0                        = 28.815e9
  MTS_sig_a                       = 10.0e6
  MTS_d                           = 3.440e9
  MTS_temp_0                      = 215.0
  MTS_b                           = 2.86e-10
  MTS_edot_0i                     = 1.0e7
  MTS_g_0i                        = 3.6
  MTS_q_i                         = 1.5
  MTS_p_i                         = 0.5
  MTS_sig_i                       = 107.2e6
/
```

8.3.4 Initializing the domain

The bodies are defined in the mesh file, as described in Chapter 6. The temperature is also specified for each body, and in this case we set the initial temperature different from the reference temperatures specified for each material in the material namelist, so that the material will begin out of equilibrium.

When solid mechanics physics is activated, an initial calculation of the displacement field and linear elastic stresses is always computed before the first time step. If temperatures of bodies are the same as the stress reference temperature of the materials assigned to those bodies, the displacements, stresses, and strains should be zero. In this case the temperature difference and the difference in thermal expansion between the two materials will result in non-zero stresses and displacements.

```
&BODY
  material_number        = 1
  mesh_material_number   = 1
  surface_name           = 'from mesh file'
  temperature            = 4.55e+02
/

&BODY
  material_number        = 2
  mesh_material_number   = 2
  surface_name           = 'from mesh file'
  temperature            = 4.55e+02
/
```

8.3.5 Applying boundary conditions

This time we do not set our boundary conditions on plane surfaces using degenerate conic sections. Instead we assign them to side sets in the mesh file.

```
&BC
  surface_name   = 'from mesh file'
  mesh_surface   = 6
  BC_variable    = 'temperature'
  BC_type        = 'HNeumann'
/

&BC
  surface_name   = 'from mesh file'
  mesh_surface   = 6
  BC_variable    = 'displacement'
  BC_type        = 'z-displacement'
  BC_value       = 0.0e0
/
```

The first two BC namelists specify a symmetry boundary on the bottom surface of the mesh ($z = 0$). The `mesh_surface` entry corresponds to the number of the side set defined in CUBIT (in this case, 6). For a symmetry boundary the thermal boundary condition must be zero flux (`'HNeumann'`), and the displacement BC should constrain the displacement normal to the surface to be zero. Also for solid mechanics the traction BC tangential to the interface (x and y directions in this case) must be zero, which is the default.

```
&BC
  surface_name    = 'from mesh file'
  mesh_surface    = 5
  BC_variable     = 'temperature'
  BC_type         = 'HTC'
  t0_convection   = 2.98e+02
  BC_value        = 1.0e3
/
```

The third BC namelist specifies a convection boundary condition for temperature on the top surface of the mesh. Solid mechanics boundary conditions default to zero traction in the x, y, and z directions.

The fourth and fifth BC namelists specify symmetry conditions for the plane defined by $x = 0$, in the same manner as for BC namelists 1 and 2. This plane corresponds to side set number 3.

The sixth and seventh BC namelists specify symmetry conditions for the plane defined by $y = 0$, in the same manner as for BC namelists 1 and 2. This plane corresponds to side set number 2.

Finally, the eighth BC namelist applies a convection temperature condition to the outer radius of the mesh, corresponding to side set number 4.

8.3.6 Selecting output formats

Outputs are selected for this problem in familiar ways. The problem is set to run for a short time (3 seconds) because the run time on some platforms can be rather long to observe substantial plastic flow. The user may want to run the problem to 100 seconds to see more interesting results.

8.3.7 Numerics

For solid mechanics we add new options to the `NUMERICS` namelist. There is no well-defined stability criterion for the nonlinear viscoplastic solution, so choosing appropriate time step parameters may be difficult. No time step control can be based on elastic strain. For plastic strain integration the `strain_limit` parameter is used to control the accuracy as well as the method used. This parameter should be set to a value that is the minimum plastic strain *increment* that is significant for a time step. Lastly, a new nonlinear solver must be specified for the displacement solution.

```
&NUMERICS
```

```
  dt_max                              = 5.0e-1
  dt_grow                             = 1.2
  dt_init                             = 2.0e-2
  dt_min                              = 1.0e-8
  strain_limit                        = 1.0e-12
  energy_nonlinear_solution           = 'conduction nonlin'
  displacement_nonlinear_solution     = 'displacement nonlin'
/
```

8.3.8 Nonlinear Solvers

There are two nonlinear solvers to define. Since the heat transfer solution involves conduction only, that solver's convergence criterion can be fairly loose. The solid mechanics solution uses the accelerated inexact Newton (AIN) method. In general the AIN nonlinear solver is more robust than Newton-Krylov for solid mechanics problems, especially for viscoplastic or contact problems.

```
&NONLINEAR_SOLVER
  name                      ='conduction nonlin'
  method                    ='NK'
  linear_solver_name        ='conduction NK'
  convergence_criterion     =1e-02
  output_mode               ='none'
  use_damper                =.false.
  perturbation_parameter    =1e-06
/

&NONLINEAR_SOLVER
  name                      ='displacement nonlin'
  method                    ='AIN'
  linear_solver_name        ='displacement AIN'
  convergence_criterion     = 1e-8
  AIN_max_vectors           = 30
  AIN_vector_tolerance      = 30
  maximum_iterations        = 200
/
```

8.3.9 Linear Solvers

There are also two linear solvers defined, one for the conduction solution, and one for the nonlinear elastic-viscoplastic solution. In this release the preconditioning method for the displacement solvers must be `'TM_SSOR'`, `'TM_diag'` or `'none'`.

```
&LINEAR_SOLVER
  name                     = 'conduction NK'
  method                   = 'FGMRES'
  preconditioning_method   = 'SSOR'
  preconditioning_steps    = 2
  stopping_criterion       = '$||r||/||r0||$'
  convergence_criterion    = 1.0e-4
/

&LINEAR_SOLVER
  name                     = 'displacement AIN'
  method                   = 'none'
  preconditioning_steps    = 4
  relaxation_parameter     = 1.0
  preconditioning_method   = 'TM_SSOR'
```

8.4 Results

As advertised, the results show that the aluminum ring shrinks and flows plasitcally around the graphite plug.

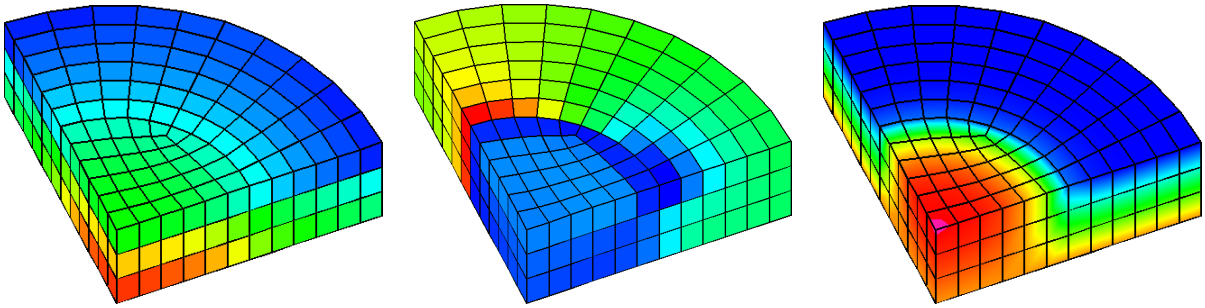
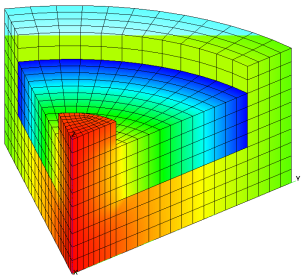


Figure 8.1: Plots of T, σ_{xx} and Δz are shown for times t = 3.0 sec

Chapter 9

Internal Interface Displacements



In this chapter we set up a simple problem to demonstrate the contact boundary condition for internal interfaces. The problem is a linear thermo-elastic solution of displacements, stresses, and strains for a solid aluminum ring in a graphite mold. The use of gap elements in order to model internal interfaces that can open gaps or accommodate interfacial sliding is demonstrated.

9.1 TRUCHAS Capabilities Demonstrated

- Gap element interfaces
- Contact boundary conditions

9.2 Problem Description

The problem consists of a ring of aluminum alloy initially in contact with a graphite mold. The mold consists of a cylindrical disk with a cavity in the shape of a ring. The mesh represents a 60 degree segment of the mold, and the boundary conditions are chosen to give a solution that is axisymmetric. Both materials have a stress reference temperature set to 700^{o}C, and the initial temperature of both bodies is set to 600^{o}C. All materials are solid, and there is no fluid flow or heat transfer in the problem.

The simulation only solves for the initial displacement, stress and strain fields caused by the difference between the initial temperature and the reference temperature and the difference in thermal expansion coefficients. All of the interfaces between the ring and graphite mold use either the `'contact'` or `'normal-constraint'` boundary condition. The contact BC prevents penetration of the materials on either side of the interface but allows separation and sliding displacement along the interface. The normal constraint condition allows sliding but does not allow either separation or penetration. Both sides of the interface can also be completely unconstrained by using the `'free-interface'` condition. The `'free-interface'` boundary condition does not prohibit interpenetration of the two bodies on either side of the interface.

Units:	SI
Input file:	UMProblems/Chapter_disp/disp.inp
Mesh file:	UMProblems/Chapter_disp/disp.g

9.3 Setup

The complete input file for this problem is presented in Appendix H. Here we discuss previously unseen sections of the input file in detail.

9.3.1 Defining a mesh

```
&MESH
  mesh_file         = 'disp.g'
  mesh_file_format  = 'exodusII'
```

In addition to the familiar mesh blocks and several side sets, The mesh file for this problem also contains element blocks whose members are “gap elements”.

Gap elements are mesh cells (initially) containing no volume that are used to model interfaces between two solid bodies. The `'contact'` BC type is only used with gap elements, as are `'free-interface'` and `'normal-constraint'` BCs. Gap elements provide connectivity between pairs of nodes that are at the same position but connected to elements on either side of the interface. Gap elements are only supported in EXODUS meshes, and are specified by block number in the `MESH` namelist.

Gap elements can be of zero or finite thickness. The easiest method of producing meshes with zero thickness gap elements is to use the CUBIT mesh generator and a mesh processor provided with the TRUCHAS distribution. The following description assumes the the user knows how to generate a mesh with CUBIT and create an EXODUS mesh file that can be read by TRUCHAS.

9.3.1.1 Mesh Generation

When creating the mesh using CUBIT, the internal surfaces that require gap elements must be assigned to side sets. These surfaces must correspond to geometric surfaces in the CUBIT model, which will put element faces on that surface. For example, if the surface number is 2 and the user wishes to label the side set as number 3 the line command would be:

```
> sideset 3 surface 2
```

The side set identifier specified in the CUBIT command is needed for the TRUCHAS `BC` namelist input variable `mesh_surface`.

There are some requirements or limitations in how the side sets are defined.

- The behavior of the interface constraints at corners or edges depends on how the side sets are defined. If a corner is formed by multiple side sets, TRUCHAS will add a separate constraint for each side set. This is probably what the user wants. If there is a sharp corner within a single side set, then an average normal at the corner will be calculated by TRUCHAS and the displacements at the corner may produce poor results. This is a TRUCHAS feature that will probably not change.

- Currently the maximum number of side sets that can intersect at a node is 3. This allows the user to construct a corner that is constrained on three sides, and should be adequate in most cases. This is a limitation of the TRUCHAS code and will probably not change soon.

- This version of the mesh preprocessor has not been tested for tet meshes. In any case tetrahedral elements are not recommended for solid mechanics.

9.3.1.2 Mesh Preprocessing

The mesh processor for creating gap elements takes a binary EXODUS mesh file as input and generates another mesh file with additional nodes and gap elements. The mesh processor is obtained by compiling the "addgaps" program by typing "make" in
`truchas/tools/mesh_processors/addgaps`. The command line syntax for the addgaps program is

```
addgaps input_file output_file sideset_id1 [sideset_id2 ...]
```

You will be prompted for a new element block ID number for each side set. This will create the mesh file 'output_file', with new element blocks for each side set. New element block numbers for each sideset will be written to a file called `addgaps.log`. See
`truchas/tools/mesh_processors/addgaps/addgaps.1` for more information.

It should also be possible to create gap elements manually in the mesh generator with a finite thickness. The element block specified as having gap elements must be a layer of cells one cell thick with cells on both sides of the layer. If the gap element block is not in this configuration the run will fail.

9.3.1.3 TRUCHAS Input

If solid mechanics interface boundary conditions are to be specified, the `MESH` namelist must have the `gap_element_blocks` set to a list of element blocks that are to be treated as gap elements. Also, the `BC` namelist must have the `surface_name` set to `'from mesh file'` and the `mesh_surface` set to the side set ID in the mesh file. A separate BC namelist is required for each side set.

If the simulation includes fluid flow, a `BODY` namelist must be defined for each gap element block that assigns a material with the attribute `immobile` set to `.true.`. This prevents TRUCHAS from attempting to compute flow in gap elements. The gap element material properties are not actually used in any of the physics calculations. However to avoid problems with the heat transfer calculation, the density, conductivity and heat capacity of gap element materials should be set to zero. The elastic constants for the gap material should also not be changed from the defaults of zero.

The side set definitions for this problem are shown in Figure 9.1. The uses of these side sets are listed in the following table.

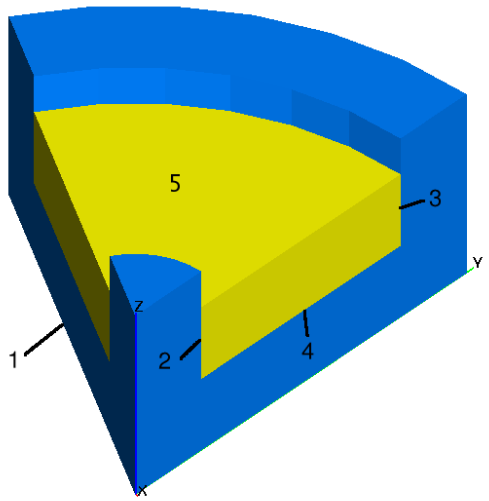


Figure 9.1: Side set definitions for this example.

Side Set	Boundary Condition	Gap Element Block
1	'normal-displacement'	N/A
2	'normal-constraint'	4
3	'contact'	5
4	'contact'	6
5	'z-traction'	N/A

9.3.2 Specifying physics

```
&PHYSICS
  fluid_flow       = .false.
  heat_conduction  = .false.
  solid_mechanics  = .true.
```

This problem uses only solid mechanics.

9.3.3 Defining materials

In this problem the graphite and aluminum materials can both be defined as linear elastic. After their namelists an example of how to define a material for gap elements is given.

```
Define the plug material properties:

&MATERIAL
  material_number               = 1
  priority                      = 1
  material_name                 =’graphite’
  density                       = 1.7e+03
  Cp_constants                  = 1.925e+03
  conductivity_constants        = 1.95e+02
  Lame1_constants               = 3.4e+9
  Lame2_constants               = 2.76e+9
  CTE_constants                 = 7.0e-06
  stress_reference_temperature  = 7.00e+02
  material_feature              = ’background’
  viscoplastic_model            = ’elastic_only’
  immobile                      = .true.
/

Define the ring material properties:

&MATERIAL
  material_number               = 2
  priority                      = 2
  material_name                 = ’5754 aluminum’
  density                       = 2.70e+03
  Cp_constants                  = 9.00e+02
  conductivity_constants        = 2.40e+02
  Lame1_constants               = 5.20e+10
  Lame2_constants               = 2.60e+10
  CTE_constants                 = 2.20e-05
  stress_reference_temperature  = 7.00e+02
  viscoplastic_model            = ’elastic_only’
  immobile                      = .true.
/

Define a suitable gap material:
```

```
&MATERIAL
  material_number         =  3
  priority                =  3
  material_name           = 'gap'
  density                 =  0.0
  Cp_constants            =  0.0
  conductivity_constants  =  0.0
  Lame1_constants         =  0.0
  Lame2_constants         =  0.0
  CTE_constants           =  0.0
  immobile                = .true.
/
```

9.3.4 Initializing the domain

In this problem the initial temperatures of the bodies are different from the reference temperatures specified for each material in the `MATERIAL` namelists. As in the previous viscoplastic problem, an initial calculation of the displacement field and linear elastic stresses is computed before the first time step.

The first two bodies make up the graphite mold, and the third body is the aluminum ring. The last three bodies are gap element blocks created from three side sets as described above.

9.3.5 Applying boundary conditions

The first two BC namelists set symmetry conditions for the bottom of the mold ($z = 0$) and the plane $x = 0$. Setting displacements to zero mimics a problem in which there is a full cylinder of infinite vertical extent.

```
&BC
  surface_name     = 'conic'
  conic_relation   = '='
  conic_z          =  1.0e+00
  conic_constant   =  0.0
  conic_tolerance  =  1.0e-06
  BC_variable      = 'displacement'
  BC_type          = 'z-displacement'
  BC_value         =  0.0
/
```

```
&BC
  surface_name     = 'conic'
  conic_relation   = '='
  conic_x          =  1.0e+00
  conic_constant   =  0.0
  conic_tolerance  =  1.0e-06
  BC_variable      = 'displacement'
  BC_type          = 'x-displacement'
  BC_value         =  0.0
/
```

The third BC namelist sets the normal displacement to zero for the plane 60^o away from the $x = 0$ plane. The surface is specified using side set number 1, which in this case is easier than using a conic.

```
&BC
  surface_name  = 'from mesh file'
  mesh_surface  =  1
  BC_variable   = 'displacement'
  BC_type       = 'normal-displacement'
  BC_value      =  0.0
/
```

The fourth BC namelist refers to side set 2 at the interface between the ring and the inner diameter of the mold cavity. The `'normal-constraint'` condition is used for this interface. Since the ring will shrink onto the plug, we assume that the ring and plug will not separate. Since the problem is axisymmetric, it is expected that displacements tangential to the interface in the $x - y$ plane will be zero.

```
&BC
  surface_name  = 'from mesh file'
  mesh_surface  =  2
  bc_variable   = 'displacement'
  bc_type       = 'normal-constraint'
  bc_value      =  0.0
/
```

The next BC namelist uses side set 3 at the interface between the ring and outer diameter of the mold cavity. The `'contact'` constraint prevents the ring and mold from penetrating each other, but allows relative separation or displacement.

```
&BC
  surface_name  = 'from mesh file'
  mesh_surface  =  3
  bc_variable   = 'displacement'
  bc_type       = 'contact'
  bc_value      =  0.0
/
```

The fifth BC namelist applies a `'contact'` condition on the interface between the bottom of the ring and the horizontal surface of the mold.

```
&BC
  surface_name  = 'from mesh file'
  mesh_surface  =  4
  bc_variable   = 'displacement'
  bc_type       = 'contact'
  bc_value      =  0.0
/
```

The sixth BC namelist applies a very small pressure (1.0 Pa downward) on the top of the ring. The contact boundary conditions do not constrain the ring from moving in the positive z direction, and this BC ensures that there is a unique solution.

```
&BC
  surface_name  = 'from mesh file'
  mesh_surface  =  5
  bc_variable   = 'displacement'
  bc_type       = 'z-traction'
  bc_value      =  -1.0
/
```

9.3.6 Selecting output formats

The output time is set to run only one time step. Since this is an elastic solution with no heat transfer, the solution is not time-dependent. At this time there is no way to stop a TRUCHAS simulation after the initial elastic solution, so one nonlinear solution will be computed.

9.3.7 Numerics

Because the solution is not time-dependent, time step parameters are not important. The three contact parameters are specified here, but the vales are all equal to the defaults. The parameter `contact_penalty` should probably not be changed for most problems, but it may be necessary to adjust the `contact_distance` or `contact_norm_trac` parameters to obtain reliable convergence.

```
&NUMERICS
  dt_constant                      = 1.0e-1
  displacement_nonlinear_solution  = 'displacement nonlin'
  contact_penalty                  = 1.0e3
  contact_norm_trac                = 1.0e4,
  contact_distance                 = 1.0e-7
/
```

9.3.8 Nonlinear Solvers

In this release the same nonlinear solver is used for both the initial elastic solution and the time-dependent solution.

```
&NONLINEAR_SOLVER
  name                  = 'displacement nonlin'
  method                = 'AIN'
  linear_solver_name    = 'displacement precon'
  convergence_criterion = 1e-3
  AIN_max_vectors       = 30
  AIN_vector_tolerance  = 0.001
  maximum_iterations    = 500
/
```

9.3.9 Linear Solvers

One linear solver is needed for the nonlinear AIN solver.

```
&LINEAR_SOLVER
  name                   = 'displacement precon',
```

```
  method                   = 'none',
  preconditioning_steps    = 4,
  relaxation_parameter     = 1.0,
  preconditioning_method   = 'TM_SSOR',
/
```

9.4 Results

Some output showing interface variables is shown in Figure 9.2. The gap opening displacement and traction normal to the interface are available for plotting. The variables are identified by the element block number.

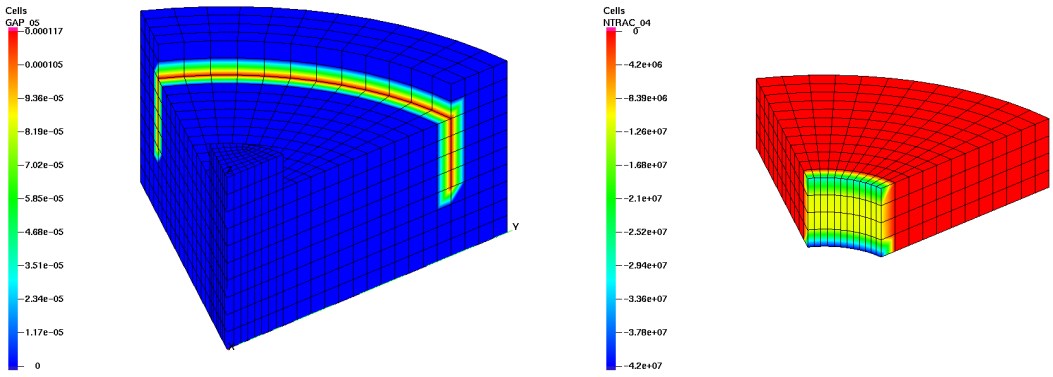


Figure 9.2: Plots of the interface gap for side set 3 (block 5) and normal traction for side set 2 (block 4).

Some stress results are plotted in Figure 9.2. A gmv file with stresses and strains transformed to cylindrical coordinates can be generated using the python script

```
tools/scripts/solid_mech_cyl_coords.py
```

The xx component of stress in Cartesian coordinates and $\theta\theta$ component in cylindrical coordinates are plotted in the figure. The mesh geometry and symmetry boundary conditions produce an axisymmetric solution.

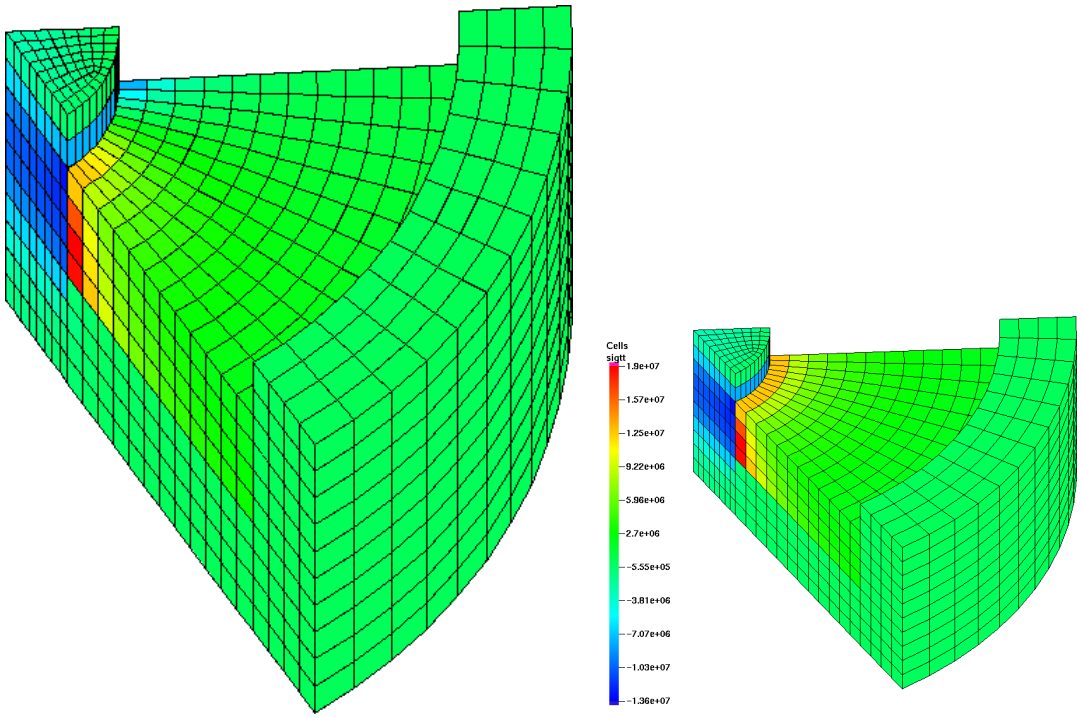
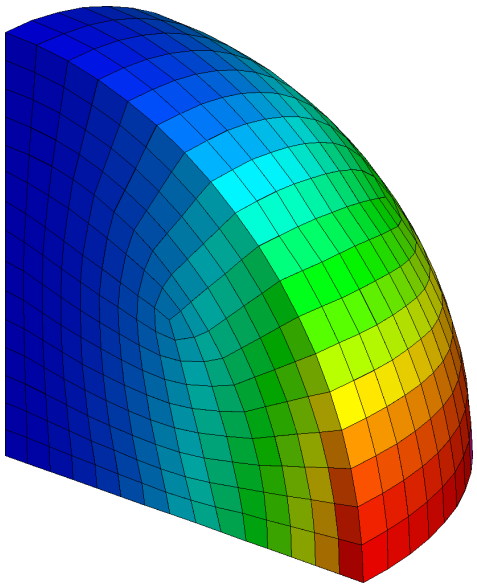


Figure 9.3: Plots of σ_{xx} and $\sigma_{\theta\theta}$

Chapter 10

Induction Heating



In this chapter we present a simple induction heating problem to introduce the electromagnetic (EM) modeling capabilities of TRUCHAS. Inputs related to setting up an EM simulation are discussed in detail.

10.1 TRUCHAS Capabilities Demonstrated

- **Electromagnetics**

10.2 Problem Description

We consider the simple induction heating problem depicted in Figure 10.1. A graphite sphere is positioned at the center of a 3-turn induction coil. An imposed sinusoidal current in the coil produces an alternating magnetic field that penetrates the graphite, inducing azimuthally-directed, alternating electric currents within the graphite. These currents generate a spatially varying heat source through resistive heating (Joule heat). The heat is thermally conducted throughout the graphite and is radiated from the surface of the sphere.

The problem is axially symmetric, and we exploit this symmetry by solving only in the positive octant, applying appropriate boundary conditions on the introduced symmetry planes. The computational domain is the quarter cylinder $\{(x, y, z) \,|\, x, y \geq 0,\ 0 \leq z \leq 0.045,\ \sqrt{x^2 + y^2} \leq 0.045\}$, which encloses the 6 cm diameter sphere and a bit of the surrounding free space within the coil. The coil itself lies outside the domain; its influence is captured by specifying the tangential component of the magnetic field on the exterior boundary. We simulate the heating of the sphere for a short time, starting from a uniform initial temperature of 300 K.

Units:	SI
Input file:	UMProblems/Chapter_ih/ih.inp
Mesh files:	UMProblems/Chapter_ih/ih-hex.gen,
	UMProblems/Chapter_ih/ih-tet.gen

10.3 Setup

The complete input file for this problem is presented in Appendix I. The discussion here focuses on those sections of the input file related to electromagnetics.

10.3.1 Defining the meshes

Alone among TRUCHAS's physics packages, the EM solver requires a tetrahedral mesh. Although it is possible to use such a mesh for all the physics in TRUCHAS, this is not recommended because many of the algorithms perform poorly with this type of mesh. Hence simulations that include electromagnetics are normally done by defining a pair of meshes: (1) a secondary, or alternative, tetrahedral mesh used for the

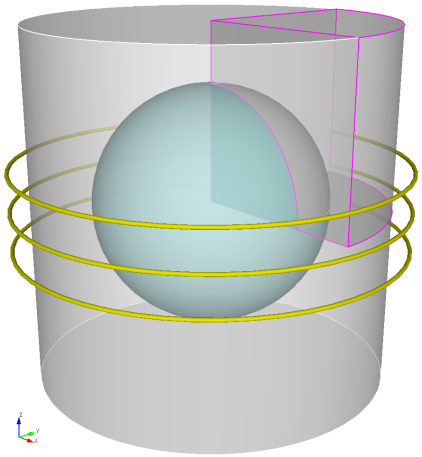


Figure 10.1: Geometry of the induction heating problem. A graphite sphere is positioned at the center of a 3-turn induction coil. EM is modeled on the pictured cylindrical region that encloses the sphere and some of the surrounding free space. Symmetry allows computation to be confined to the highlighted octant.

EM calculations; and (2) the usual (hexahedral-cell) mesh used for the rest of the physics. Quantities are interpolated from one mesh to the other as needed.

```
&MESH
  Mesh_File        = 'ih-hex.gen'
  Mesh_File_Format = 'ExodusII'
/
&ALTMESH
  Altmesh_File     = 'ih-tet.gen'
/
```

The `ALTMESH` namelist describes the tetrahedral mesh used by the EM solver. Here the mesh is read from the Genesis/Exodus-format file `ih-tet.gen`. For this problem we used the external meshing package CUBIT to generate both meshes, which are shown in Figure 10.2. Notice that the free space region is only included in the tet mesh, and that the two meshes do not exactly conform to one another along the curved surface of the sphere, but this is not necessary: the interpolation procedure is able to handle reasonably general pairs of meshes. The hex-to-tet interpolation (used to map material parameters) is constant-preserving, and the tet-to-hex interpolation (used to map the computed Joule heat) is conservative.

There are a few requirements that a hex-tet mesh pair needs to satisfy. Both mesh files must be Genesis/Exodus format, and each element block (also referred to as a mesh material) in the tet mesh that describes a conductive region must be 'contained' within the region described by the hex-mesh element block of the same ID, allowing that some tets may extend slightly outside the region. See Appendix I in *Truchas Physics*

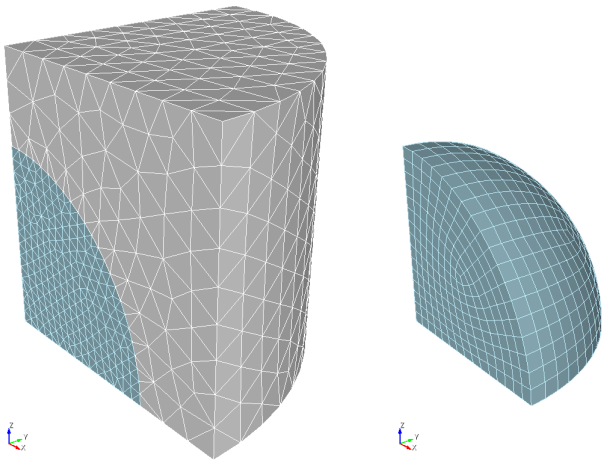


Figure 10.2: Tetrahedral mesh used by the EM solver (left), and the hexahedral mesh used for heat conduction/radiation (right), which omits the free space region.

and Algorithms and the `ALTMESH` chapter of *Truchas Reference Manual* for further details.

10.3.2 Specifying physics

```
&PHYSICS
  fluid_flow       = .false.
  heat_conduction  = .true.
  electromagnetics = .true.
/
```

Enabling `electromagnetics` instructs TRUCHAS to perform an auxiliary EM simulation to calculate the Joule heating distribution for use as a volumetric heat source in heat conduction. It is assumed that the time scale of the alternating EM fields is much shorter than the time scale of the rest of the physics. In this case it suffices to solve Maxwell's equations to a periodic steady state while temporarily freezing the other physics, and then average the rapid temporal variation in the Joule heating distribution over a cycle. In effect, Maxwell's equations are solved over an *inner time* distinct from that of the rest of the physics. In this problem the Joule heating distribution is constant in time, so it is computed just once at the outset of the simulation. In general, however, it is recomputed whenever indicated by changes in the EM material parameters (due to their temperature dependence) or by changes in the magnetic source field.

10.3.3 Defining materials

```
&MATERIAL
  material_number           = 1
  material_name             = 'graphite'
  material_feature          = 'background'
  density                   = 1750.0
  Cp_constants              = 895.0
  conductivity_constants    = 7.80
  EM_conductivity_constants = 5.6e4
  EM_permittivity_constants = 1.0
  EM_permeability_constants = 1.0
/
```

Here we have specified the values for electric conductivity, relative electric permittivity, and relative magnetic permeability for graphite. These are the only material parameters used by the EM solver. The default values are 0 for conductivity and 1 for relative permittivity and permeability, so in this case we could have omitted the latter two from the namelist.

10.3.4 Applying boundary conditions

For this example, the cell faces on the boundary of the domain were grouped into two side sets, labeled 1 and 2 within the mesh file. Faces on the symmetry planes were placed in side set 1, and we specify a homogeneous Neumann condition for temperature to that part of the boundary. The remaining boundary faces—those on the surface of the sphere—were placed in side set 2, and we specify a thermal radiation boundary condition there, with a surface emissivity of 0.9 and an ambient temperature of 300K.

```
&BC
  surface_name = 'from mesh file'
  mesh_surface = 1
  BC_variable  = 'temperature'
  BC_type      = 'HNeumann'
/
&BC
  surface_name = 'from mesh file'
  mesh_surface = 2
  BC_variable  = 'temperature'
  BC_type      = 'radiation'
  BC_value     = 0.9, 300.0
/
```

Notice that we have not specified boundary conditions for the EM fields. These are handled differently; see Section 10.3.6.

10.3.5 Numerics

Here we specify how to perform the heat conduction solve. Parameters for the electromagnetic solve are set in the following namelist.

`NUMERICS` section of the `Reference Manual`.

```
&NUMERICS
  dt_constant               = 0.5
  energy_nonlinear_solution = 'HT NLS'
  HT_discrete_ops_type      = 'SO'
/
```

10.3.6 Electromagnetic Solver

Instances allowed: single `ELECTROMAGNETICS`, and multiple `INDUCTION_COIL`, one for each coil (if any).

```
&ELECTROMAGNETICS
  EM_Domain_Type        = 'quarter_cylinder'
  Source_Frequency      = 2000.0
  Steps_Per_Cycle       = 20
  Maximum_Source_Cycles = 5
  SS_Stopping_Tolerance = 0.01
  Maximum_CG_Iterations = 100
  CG_Stopping_Tolerance = 1.0e-8
  Num_Etasq             = 1.0e-6
/
&INDUCTION_COIL
  Center  = 3*0.0
  Radius  = 0.05
  Length  = 0.04
  NTurns  = 3
  Current = 5000.0
/
```

Here we specify all the variables that control the Joule heat calculation. Several aspects of this specification differ from other types of physics.

First, the boundary conditions for EM are defined internal to the EM package. In this release there is no facility for specifying general boundary conditions for EM simulations. Consequently the computational domain is restricted to a few special cases. `quarter_cylinder` declares that the domain discretized by the alternative mesh (see Section 12.2.1) is bounded by the form $\{(x, y, z) \,|\, x, y \geq 0,\ z_1 \leq z \leq z_2,\ x^2 + y^2 \leq r^2\}$. This ensures that the proper symmetry conditions are applied on the boundaries $x, y = 0$, and that a Dirichlet source field condition is applied to the tangential component of the magnetic field on the remaining boundaries. For this problem, the source field used to set boundary values is that due to a coil whose physical characteristics are specified in the `INDUCTION_COIL` namelist. The coil carries a sinusoidally-varying current with frequency 2000 Hz and peak amplitude of 5000 A. See the `ELECTROMAGNETICS` and `INDUCTION_COIL` chapters of the *Truchas Reference Manual* for the other possible types of domains and source fields.

Second, parameters for the EM solver are defined in the `ELECTROMAGNETICS` namelist. Recall that the Joule heat is computed by evolving Maxwell's equations to a periodic steady state and then averaging the Joule heat over a cycle. Here we use 20 time steps per cycle of the source field. We continue to evolve until the relative difference in the averaged Joule heat in successive cycles is less than 0.01, or we have completed the maximum number of cycles specified by `maximum_source_cycles`. The EM solver uses its own special preconditioned conjugate gradient (CG) solver. Here we limit the number of CG iterations in a linear solve to 100 and stop iterating as soon as the initial residual has been reduced by 10^{-8}. Finally, in this physical regime the coefficient η^2 of the normalized displacement current term is exceedingly small, and we have chosen to replace it with the larger artificial value specified by `num_etasq` in order to improve the numerical robustness of the problem without noticably effecting the results. See the `ELECTROMAGNETICS` chapter of the *Truchas Reference Manual* for a comprehensive list of all possible EM solver variables.

10.4 Results

Sample results from the problem are shown in Figures 10.3 and 10.4.

Figure 10.3 shows the result of the EM solve. We see that Joule heat is concentrated near the equator of the sphere, because the strongest induced current loops are there, and the fields produced by those current loops are in the opposite sense to the induction coil's imposed field, tending to cancel that field in the inner parts of the sphere. Since there are then small or no currents in the inner parts, there is no induced Joule heating there.

30 seconds of heat-up for the sphere under the oscillating imposed coil field is shown in Figure 10.4. Joule heat appears as a volumetric source near the equator and conducts throughout the graphite volume.

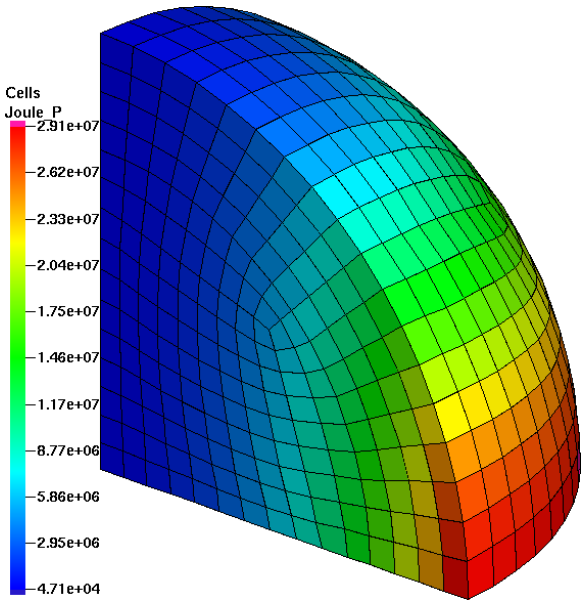


Figure 10.3: Joule heating distribution in the graphite. Data shown on the hex mesh after interpolation from the tetrahedral EM mesh where it was computed.

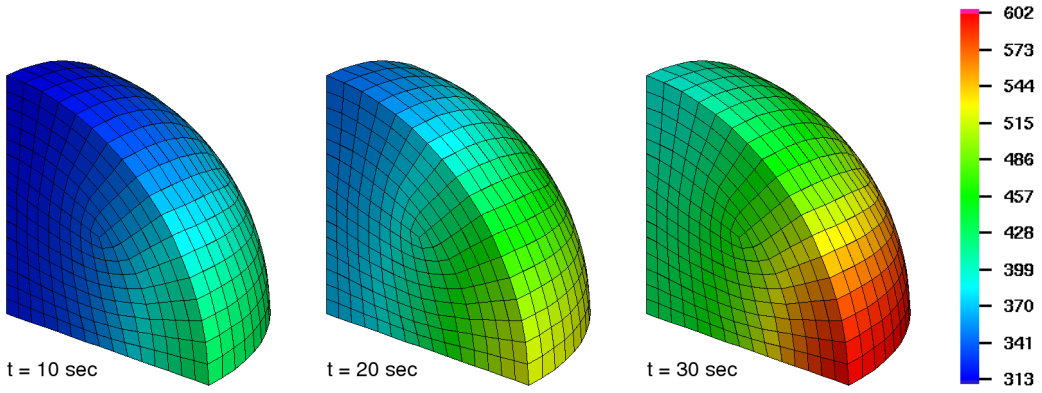
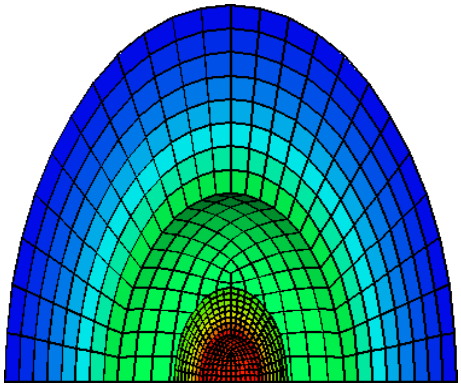


Figure 10.4: Time sequence of the temperature profile showing the heating of the graphite starting from a uniform 300K at $t = 0$.

Chapter 11

Radiation



In this chapter we set up a problem to demonstrate the TRUCHAS package that computes radiative heat transfer using a viewfactor method. This capability is important when treating heat transfer problems where bodies are not touching (so that conduction cannot transfer heat) and are surrounded by vacuum (so that convection cannot transfer heat).

11.1 TRUCHAS Capabilities Demonstrated

- Viewfactor radiative heat transfer

11.2 Problem Description

The selected problem represents two concentric hollow spherical shells separated by a vacuum(Figure 11.1). The two solid regions are denoted by R_1, R_2 and the four surfaces separating solid from vacuum by S_j at radii r_j, $j = 0, 4$. R_1 is bounded by S_0 and S_1 and R_2 by S_2 and S_3. We impose an outward heat flux at the innermost surface S_0. The temperature at the outermost surface S_3 is held constant. Heat conducts through the inner sphere to its outer boundary S_1, where it is radiated into the vacuum separating the inner and outer spheres. Surface S_2 absorbs some of that heat and reflects the rest. The absorbed heat is conducted through the outer shell to the temperature-clamped heat sink at S_3.

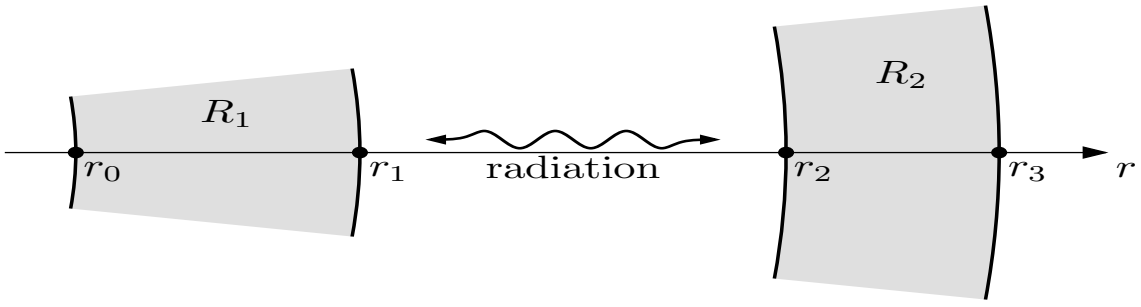


Figure 11.1: Concentric hollow spherical shells, R_1 and R_2, coupled through radiative heat transport within the intervening void.

The derivation uses two principles of physics: (1) conductive heat flux $q(r)$ in a solid is proportional to the negative of the temperature gradient, with thermal conductivity as the linear constant; and (2) in a steady state spherically symmetric configuration of bodies with spherically symmetric sources and sinks of heat, the steady state heat flux at any radius varies inversely as the square of radius, *i. e.* the quantity $r^2 q(r)$ is constant within R_1 and R_2. Thus for r in R_1

$$ {r_0}^2 q_0 = -r^2 k_1 T'(r), \tag{11.1} $$

where k_1 is the thermal conductivity in R_1, from whence it follows that

$$ T(r) = T_1 + \frac{{r_0}^2 q_0}{k_1}\Big(\frac{1}{r} - \frac{1}{r_1}\Big), \quad r \in [r_0, r_1]. \tag{11.2} $$

In particular,

$$ T_0 = T_1 + \frac{q_0}{k_1}\frac{r_0}{r_1}(r_1 - r_0). \tag{11.3} $$

Thus the temperature in R_1 is completely determined once the surface temperature T_1 is determined. Similarly in R_2, ${r_2}^2 q_2 = -r^2 k_2 T'(r)$ from whence it follows that

$$ T(r) = T_3 + \frac{{r_2}^2 q_2}{k_2}\Big(\frac{1}{r} - \frac{1}{r_3}\Big), \quad r \in [r_2, r_3], \tag{11.4} $$

where T_3 is the imposed temperature at r_3. So in particular,

$$T_2 = T_3 + \frac{q_2}{k_2}\frac{r_2}{r_3}(r_3 - r_2). \tag{11.5}$$

Next consider radiative heat transfer in the void region. Let Q_j denote the net radiative flux leaving surface S_j, and Q_j^i the radiative flux *incident* on S_j, $j = 1, 2$. Now Q_j is the sum of the emitted flux and the reflected portion of the incident flux; for diffuse gray-bodies,

$$Q_j = \epsilon_j \sigma {T_j}^4 + (1 - \epsilon_j) Q_j^i, \quad j = 1, 2, \tag{11.6}$$

where $0 < \epsilon_j \leq 1$ is the emissivity of S_j, and σ is the Stefan-Boltzmann constant. The incident flux Q_j^i is defined by the linear relation,

$$\begin{bmatrix} Q_1^i \\ Q_2^i \end{bmatrix} = F \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}, \qquad F = \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix}, \tag{11.7}$$

where f_{ij} is the view factor between surfaces S_i and S_j. For the present geometry, $f_{11} = 0$ since no distinct points on S_1 are visible to one another. Thus $f_{12} = 1 - f_{11} = 1$. Since viewfactors satisfy a reciprocity relation, namely for S_1 and $S - 2$ $A_2 f_{21} = A_1 f_{12}$, where A_j is the area of S_j, $f_{21} = (r_1/r_2)^2$. Finally $f_{22} = 1 - f_{21} = 1 - (r_1/r_2)^2$. To summarize,

$$F = \begin{bmatrix} 0 & 1 \\ (r_1/r_2)^2 & 1 - (r_1/r_2)^2 \end{bmatrix}. \tag{11.8}$$

Next we match the conductive fluxes at S_1 and S_2 to the radiative fluxes. The flux q_1 is the difference between the emitted flux and the absorbed portion of the incident flux

$$\begin{aligned} q_1 &= \epsilon_1 \sigma {T_1}^4 - \epsilon_1 Q_1^i, \\ &= Q_1 - Q_1^i. \end{aligned} \tag{11.9}$$

Similarly,

$$q_2 = Q_2^i - Q_2. \tag{11.10}$$

These relations lead to

$$q_2 = \left(\frac{r_1}{r_2}\right)^2 q_1. \tag{11.11}$$

That is, the second physical principle above is satisfied across the void gap between the two concentric shells. Thus

$$q_2 = \left(\frac{r_0}{r_2}\right)^2 q_0, \tag{11.12}$$

giving the flux at S_2 in terms of the imposed heat flux at S_0.

From this we obtain the single independent equation

$$Q_1 - Q_2 = \left(\frac{r_0}{r_1}\right)^2 q_0. \tag{11.13}$$

Taking this equation together with equation 11.6 for Q_j yields a linear system for Q_1 and Q_2,

$$\begin{bmatrix} 1 & -(1-\epsilon_1) \\ -(1-\epsilon_2)(r_1/r_2)^2 & \epsilon_2 + (1-\epsilon_2)(r_1/r_2)^2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \begin{bmatrix} \epsilon_1 \sigma {T_1}^4 \\ \epsilon_2 \sigma {T_2}^4 \\ q_0 (r_0/r_1)^2 \end{bmatrix}. \tag{11.14}$$

By the Fredholm Alternative this sytem is solvable if and only if the right hand side is orthogonal to the left nullspace of the coefficient matrix. It is easily verified that the nullspace is spanned by the vector $\big(-\epsilon_2, \epsilon_1, \epsilon_2 + \epsilon_1(1-\epsilon_2)(r_1/r_2)^2\big)$. Thus solvability implies

$${T_1}^4 = {T_2}^4 + \frac{q_0}{\sigma}\left(\frac{r_0}{r_1}\right)^2 \left[\frac{1}{\epsilon_1} + \left(\frac{1}{\epsilon_2} - 1\right)\left(\frac{r_1}{r_2}\right)^2\right]. \tag{11.15}$$

This completes our set of relations between the temperatures at the four surfaces.

To run this problem with TRUCHAS, the bodies are initialized with some constant temperature, and the temperature distribution then evolves toward a steady state that depends on the imposed heat flux at S_0, the imposed temperature at S_3, and the material parameters of the solids. Our sample problem is presented in dimensionless units, so that results can readily be compared with the analytic solution.

Units:	dimensionless
Input file:	UMProblems/Chapter_rad/rad.inp
Mesh file:	UMProblems/Chapter_rad/2shell8.gen
Viewfactor file:	UMProblems/Chapter_rad/2shell8-le.vf

11.3 Setup

The input file for this problem is presented in its entirety in Appendix J. In this section we describe new features of the input needed for problems involving radiative heat transfer using a viewfactor method.

11.3.1 Specifying physics

In TRUCHAS the radiative heat transfer package is called "enclosure radiation," to distinguish it from the plain "radiation" option among the boundary conditions.

```
&PHYSICS
  fluid_flow                = .false.
  heat_conduction           = .true.
  Stefan_Boltzmann          = 1.0
/
```

We activate heat conduction physics only. Enclosure radiation physics, like induction heating physics, has its own namelist to set parameters. Note that we have reset the Stefan-Boltzmann constant from its default SI value to unity, in keeping with dimensionless units.

11.3.2 Defining materials

Also in keeping with dimensionless units, we define hypothetical materials `'foo'` and `'bar'` with unit density and specific heat. We give them different constant thermal conductivities to make things interesting. The user can play with these parameters to see how the solution varies.

The third material must be included for a `'background'` material.

```
&MATERIAL
  material_number           = 1
  material_name             = 'foo'
  density                   = 1.0
  Cp_constants              = 1.0
  conductivity_constants    = 4.0
/

&MATERIAL
  material_number           = 2
  material_name             = 'bar'
  density                   = 1.0
  Cp_constants              = 1.0
  conductivity_constants    = 1.0
/

!! Dummy material to satisfy Truchas need for background material.
&MATERIAL
  material_number           = 3
  material_name             = 'void'
  material_feature          = 'background'
  density                   = 0.0
  Cp_constants              = 0.0
  conductivity_constants    = 0.0
  void_temperature          = 0.0
```

```
/
```

11.3.3 Applying boundary conditions

Setting of temperature boundary conditions follows the definition of the problem. A Neumann condition specifies an imposed heat flux (the value being that flux), a Dirichlet condition specifies an imposed temperature (the value being that temperature), and a homogeneous Neumann condition sets the flux to zero.

A new type of BC is specified for the surfaces on either side of the void between the concentric shells. The BC `'enclosure_radiation'` specifies viewfactor radiative transfer in and out of those surfaces. The `'BC_value'` settings have two values. The first is the index of the enclosure to which the surface belongs—because a problam can have multiple enclosures— and the second value is the emissivity ($0 < \epsilon \leq 1$) of the surface. The presence of at least one `'BC_type'` enclosure radiation also alerts TRUCHAS to look for `'RADIATION_ENCLOSURE'` namelists, one per enclosure.

```
!! Imposed flux on inner surface of inner shell.
&BC
  surface_name                = 'from mesh file'
  mesh_surface                = 1
  BC_variable                 = 'temperature'
  BC_type                     = 'Neumann'
  BC_value                    = 16.0
/

!! Radiation from outer surface of inner shell.
&BC
  surface_name                = 'from mesh file'
  mesh_surface                = 2
  BC_variable                 = 'temperature'
  BC_type                     = 'enclosure_radiation'
  BC_value                    = 1, 0.8
/

!! Radiation from inner surface of outer shell.
&BC
  surface_name                = 'from mesh file'
  mesh_surface                = 3
  BC_variable                 = 'temperature'
  BC_type                     = 'enclosure_radiation'
  BC_value                    = 1, 0.2
/

!! Imposed temperature on outer surface of outer shell.
```

```
&BC
  surface_name                = 'from mesh file'
  mesh_surface                = 4
  BC_variable                 = 'temperature'
  BC_type                     = 'Dirichlet'
  BC_value                    = 1.0
/

!! No-flux conditions on symmetry planes.
&BC
  surface_name                = 'from mesh file'
  mesh_surface                = 5
  BC_variable                 = 'temperature'
  BC_type                     = 'HNeumann'
/
```

11.3.4 Specifying enclosure radiation: the `RADIATION_ENCLOSURE` namelist

Namelist instances allowed: multiple, one per enclosure

```
&RADIATION_ENCLOSURE
  enclosureID                 = 1
  partial                     = .false.
  method                      = 'file'
  infile                      = '2shell8-le.vf'
  chaparral_symmetry          = 'x', 'y', 'z'
  linearsolver                = 'RAD LS'
/
```

In this namelist we specify the method for computing the viewfactor-based radiative transfer between the surfaces in one enclosure. We label our enclosure number 1, declare it to be not partial (that is complete with no openings to a surrounding environment), and specify the method `'file'`, meaning that the viewfactors between pairs of cell faces have been precomputed and reside in the file specified by `'infile'`. Note that the viewfactors for the analytic solution derived above were for entire surfaces, whereas to compute the problem with TRUCHAS we must have face-by-face viewfactors that depend on the way the domain is meshed.

For our problem and the specific mesh provided, viewfactors were computed using the code Chaparral. Please see the TRUCHAS *Installation Guide* for information on how to obtain Chaparral. The `'chaparral_symmetry'` line in our input file specifies that in order to create the complete 3D geometry of cell faces that Chaparral requires, the set of cell faces in our one-octant mesh must be reflected about the x, y, and z axes. Without these reflections (copies) of the cell faces, Chaparral would not see that a face on the concave surface S_2

can see the majority of the other faces on S_2 as well as nearly half of the faces on S_1. Chaparral would not compute the correct viewfactors. If our mesh included half of the concentric spheres, only one reflection would be necessary.

In the input file (Appendix J) one will find two alternative lines that could be put in the `RADIATION_ENCLOSURE` namelist, *viz.*

```
  method                        = 'chaparral'
  outfile                       = '2shell8.vf'
```

Because these lines are not between any valid pair of namelist delimiters, they are ignored by the input file parser, as are comments. If they were included in place of the `'method'` and `'infile'` lines—and only if the TRUCHAS code had been compiled including the Chaparral package—then Chaparral would be invoked to compute viewfactors before the heat conduction calculation was performed, and in addition the resulting viewfactors would be written out to a file with the specified name. On a rerun of the problem, say with different material parameter settings, that outfile could be used as an infile with `method = 'file'`. However, if the mesh file were modified, say to refine the mesh, viewfactors would need to be recomputed.

11.3.5 Linear solver

The enclosure radiation package requires a linear solver to compute the radiative heat balance in the enclosure using viewfactors. The solver specified as `'RAD LS'` seems to work well for this problem. As always if convergence of a solver is not being attained, alternative settings should be tried.

```
&LINEAR_SOLVER
  name                          = 'RAD LS'
  method                        = 'cg'
  preconditioning_method        = 'none'
  stopping_criterion            = '||r||/||r0||'
  convergence_criterion         = 1.0e-3
/
```

11.3.6 Selecting output formats

In the `OUTPUTS` namelist is a new parameter, `enclosure_diagnostics`. If this parameter is `.true.`, TRUCHAS will produce GMV files that can be examined to check the reasonableness of the computed viewfactors. If needed surfaces have been inadvertently omitted from a mesh file side set used to specify a radiating surface in an enclosure, often these diagnostic files will reveal hot spots or cold spots in the viewfactor set, or other anomalies.

11.3.7 Numerics

```
&NUMERICS
  dt_constant                 = 4.0e-3
  discrete_ops_type           = 'ortho'
  HT_discrete_ops_type        = 'SO'
  energy_nonlinear_solution   = 'HT NLS'
  enclosure_flux_convergence  = 4.0e-5
/
```

One new line here is `HT_discrete_ops_type`. This setting overrides the global setting of `discrete_ops_type` *only* for heat transfer, namely the computation of the face temperature gradients that are used to compute cell-centered temperatures from face temperatures. The choice here is `'SO'` or Support Operators. Please refer to the *Truchas Physics and Algorithms* for details on various types of computations of discrete operators. The user can remove this line to compare results with `'ortho'` computations.

For problems employing enclosure radiation, we can specify in this namelist a convergence criterion for the flux computation.

11.3.8 Requesting data probes: the PROBE namelist

Namelist instances allowed: multiple, one for each data probe

```
&PROBE
  probe_name    = 'r0'
  probe_coords  = 0.0, 0.0, 0.5
/

&PROBE
  probe_name    = 'r1'
  probe_coords  = 0.0, 0.0, 1.0
/

&PROBE
  probe_name    = 'r2'
  probe_coords  = 0.0, 0.0, 2.0
/

&PROBE
  probe_name    = 'r3'
  probe_coords  = 0.0, 0.0, 4.0
/
```

Data probes are points within the meshed domain in which the user desires ancillary data as a function of time. In this case we want to write out to files the time history data at points at the four radii r_0 through r_3, so that, for example, the temperatures at those points can be compared to the temperatures expected from the analytic solution. We give each data probe a name, so it can be distinguished from others, and we specify the xyz coordinates of the location desired.

A file will be written for each data probe. Any plotting utility can be employed to plot time histories for the variables.

11.4 Results

For this problem one must ensure that the viewfactor input file is in the path, as well as the TRUCHAS executable and the mesh file.

Temperature distributions are shown in Figure 11.2. Heat spreads outward from the innermost surface S_0, reaching surface S_1 quickly. There is then a delay, because radiative heat transfer across the void gap to S_2 is slower than conduction. After S_2 heats up, heat is conducted through to S_3 and the distribution relaxes toward a steady state.

Of course with a time evolution such as this computed with TRUCHAS, a true steady state can never be reached. The approach to a steady state can be seen more readily in Figure 11.3. Expected values of the temperatures at the four probe locations are given at the beginning of the input file, calculated using the equations. The user should verify those analytic values, not forgetting to substitute actual material property values from the `MATERIAL` namelists.

Note that the time histories of the data probes do not exactly approach the expected values. In part this is due to not running the simulation longer, but another effect is at work that the user needs to understand. In TRUCHAS temperatures are computed only at cell centers, whereas in the mesh the probe locations are at or near nodes. TRUCHAS computes cell temperatures using an averaging scheme, so the difference between expected values and probed values is partly due to this difference in location.

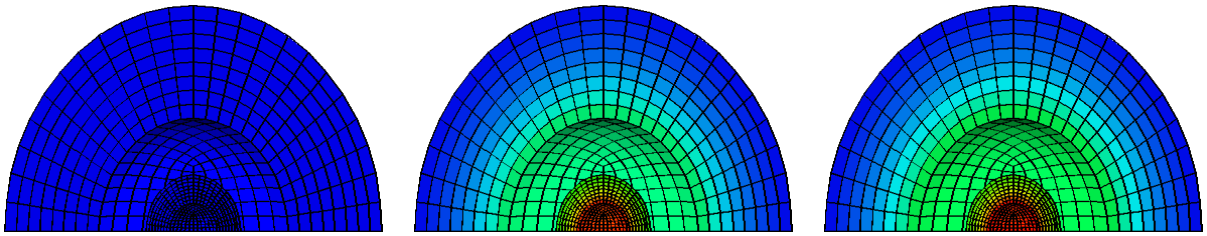


Figure 11.2: Distributions of T are shown for times t = 0.0, 2.0 and 4.0 time units. The viewpoint is in the xy plane, 45 degrees from the axes.

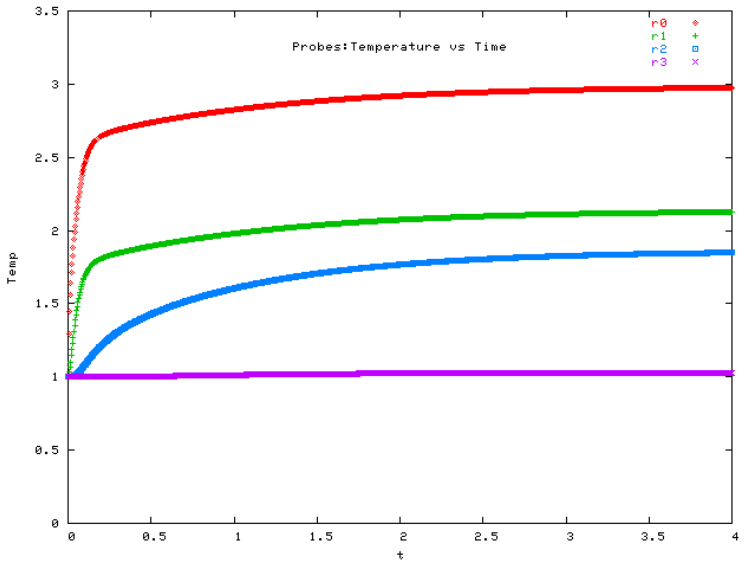
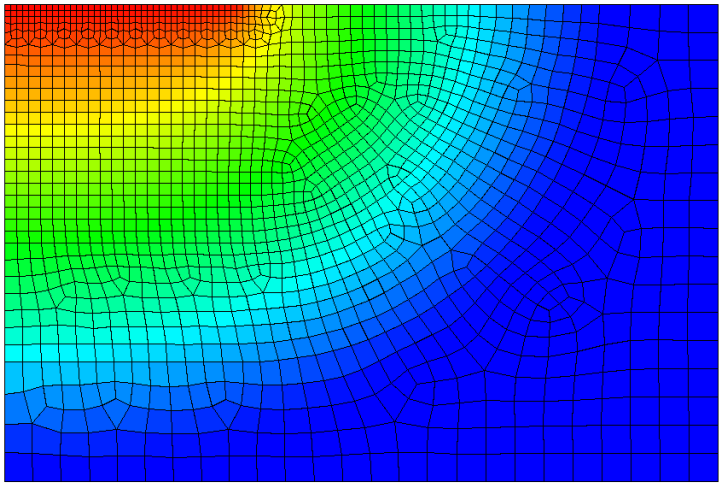


Figure 11.3: Data probe time histories, showing the approach to a steady state.

Chapter 12

Diffusion Solver



This chapter introduces the diffusion solver component of TRUCHAS, which is a new feature of this release. A nonlinear solid-state diffusion problem is used to illustrate its capabilities.

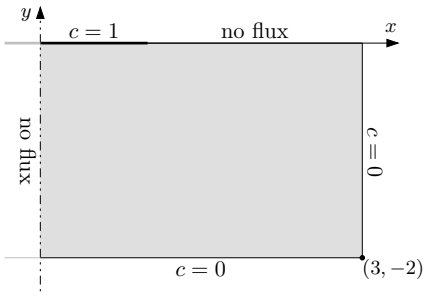


Figure 12.1: Computational domain and boundary conditions for the nonlinear diffusion problem.

12.1 Problem Description

We consider the simple solid-state diffusion problem depicted in Figure 12.1. A solutal species with concentration c diffuses into a substrate material through a portion of the top surface. On the remainder of the top surface and on the left side symmetry plane no flux of species is imposed. Artificial boundaries on the right and bottom sides are introduced to truncate the domain. At $t = 0$ we assume $c = 0$ throughout the domain, and impose $c = 1$ on the inflow part of the top side for $t > 0$. The concentration field $c(x, t)$ satisfies the heat equation with a concentration dependent diffusivity,

$$\frac{\partial c}{\partial t} = \nabla \cdot D(c)\nabla c, \quad D(c) = 0.02 + c.$$

Input file: `UMProblems/Chapter_ds/ds1.inp`
Mesh file: `UMProblems/Chapter_ds/umds1.gen`

12.2 Setup

The complete input file for this problem is presented in Appendix K. The discussion here focuses on those sections of the input file related to the diffusion solver.

12.2.1 Defining the mesh

The mesh, which can be seen in Figure 12.2, is an unstructured, quasi-2D, hexahedral mesh that is one cell thick in the out-of-plane z direction. It was created by Cubit and written in Exodus/Genesis format to the file `umds1.gen`. Note that the diffusion solver requires that meshes be in this format. Thus an internally generated mesh, for example, can not be used with the diffusion solver.

```
&MESH
  Mesh_File = 'umds1.gen'
  Mesh_File_Format = 'ExodusII'
/
```

12.2.2 Specifying physics

Here we enable the diffusion solver, though the specific type of diffusion system to be solved will be specified later. Since fluid flow is enabled by default, we must also disable it.

```
&PHYSICS
  Diffusion_Solver = .true.
  Fluid_Flow       = .false.
/
```

The diffusion solver is not currently co-operable with the other physics solvers, so it is not possible, for example, to enable flow at the same time and solve an advection-diffusion system. This is a limitation that will be eliminated in the next release of TRUCHAS.

12.2.3 Defining materials and bodies

The computational domain consists of a single region to which we associate a single material. All the cells in the Cubit mesh belong to element block 10, so we use a single `BODY` namelist referencing that ID. We also specify 0 as the initial value of the concentration field here.

```
&BODY
  Surface_Name         = 'from mesh file'
  Mesh_Material_Number = 10
  Material_Number      = 1
  Temperature          = 0.0
  Concentration        = 0.0
/
```

The species diffusivity $D(c)$ in the substrate material is specified using the `SPC_Diffusivity_*` variables in the `MATERIAL` namelist for that material. Here it is a polynomial function in the concentration with the indicated powers and corresponding coefficients.

```
&MATERIAL
  Material_Number  = 1
```

```
  Material_Name    = 'substrate'
  Material_Feature = 'background'
  Density          = 1.0,
  SPC_Diffusivity_Relation  = 'concentration polynomial'
  SPC_Diffusivity_Constants = 0.02, 1.0
  SPC_Diffusivity_Exponents = 0, 1
/
```

We can see several idiosyncrasies of TRUCHAS here: an initial value for the (unused) temperature field must be assigned and every material must be assigned a density even if it is not relevant to the problem.

12.2.4 Defining boundary conditions

The boundary conditions for the diffusion solver's concentration variable are specified in the DS_SPECIES_BC namelist instead of the ususal BC namelist. When this Cubit mesh was created, the boundary cell faces were grouped into several suitable side sets, and we reference the IDs of these sets when defining the boundary conditions. Note that this is the *only* way of identifying sets of boundary faces in this namelist.

```
&DS_SPECIES_BC
  Name         = 'inflow boundary'
  Face_Set_IDs = 5
  BC_Type      = 'dirichlet'
  BC_Value     = 1.0
/
&DS_SPECIES_BC
  Name         = 'noflow boundaries'
  Face_Set_IDs = 1, 4, 11, 12
  BC_Type      = 'hneumann'
/
&DS_SPECIES_BC
  Name         = 'artificial boundaries'
  Face_Set_IDs = 2, 3
  BC_Type      = 'dirichlet'
  BC_Value     = 0.0
/
```

12.2.5 Numerics

```
&DIFFUSION_SOLVER
  System_Type  = "species"
  Abs_Conc_Tol = 1.0e-4
```

```
  Rel_Conc_Tol = 1.0e-3
/
```

The diffusion solver is capable of solving different types of coupled diffusion systems. A simple scalar diffusion equation of the form $c_t = \nabla \cdot D(c)\nabla c$, as is our case, is selected by giving the `"species"` keyword. Currently this is the only implemented option.

The diffusion solver uses a variable step-size BDF2 method to integrate the coupled diffusion system. The step sizes are chosen automatically in order to maintain an estimate of the local truncation error of each step within a user-defined range. If $\delta\mathbf{c}$ is a concentration field difference with reference concentration field $\mathbf{c}$, this error is computed using the norm

$$|||\delta\mathbf{c}||| \equiv \max_j |\delta c_j|/(\epsilon + \eta|c_j|).$$

The absolute tolerance ϵ is given by `Abs_Conc_Tol` and the relative tolerance η is given by `Rel_Conc_Tol`.

See the *Truchas Reference Manual* for the other numerical parameters which can be specified in the `DIFFUSION_SOLVER` namelist.

Finally, the initial time step size, which pertains to all physics solvers, is specified in the `NUMERICS` namelist.

```
&NUMERICS
  Dt_Init = 1.0d-6
/
```

12.3 Results

Sample results from the problem are shown in Figure 12.2.

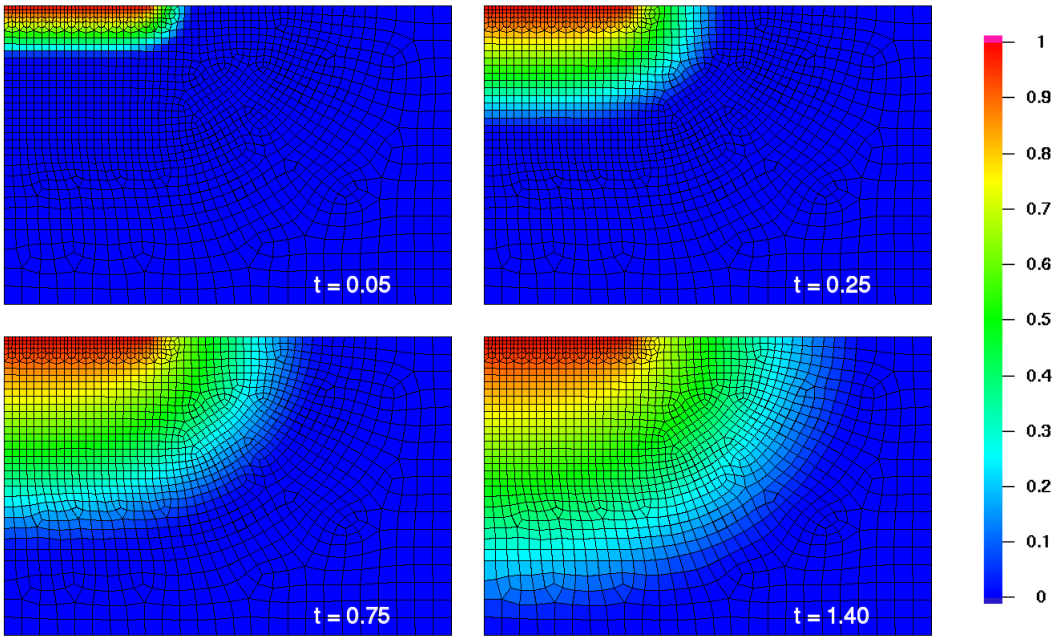
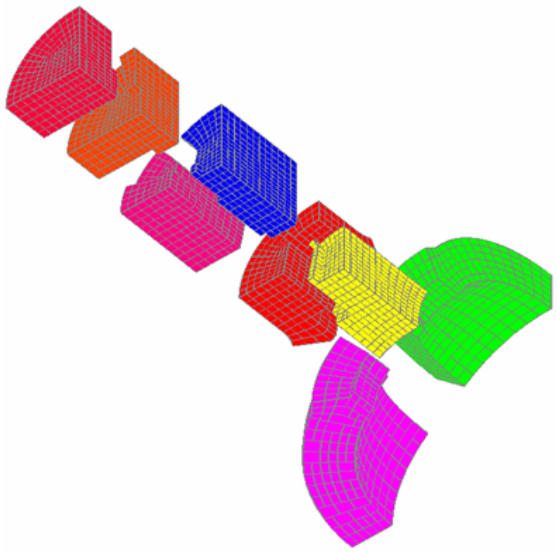


Figure 12.2: Time sequence of the concentration profile.

Chapter 13

Parallel Execution



Parallel execution of TRUCHAS is based on domain decomposition. The problem domain is decomposed into several subdomains, and the computations in each subdomain are done independently in parallel. The results from each subdomain computation are communicated to other subdomains as needed by communication software. The picture above shows a mesh decomposed into several parts, ready for a parallel computation.

TRUCHAS uses the Message Passing Interface (MPI) as the underlying parallel communication interface. There are many MPI implementations, all different. Learning about and interfacing with your particular MPI implementation will be the difficult part of running TRUCHAS in parallel.

Any of the problems in the User Manual can be run in parallel. For this example, we choose a simple problem, the heat conduction problem from Chapter 2. This problem uses a structured mesh, so it can be used to demonstrate methods used in TRUCHAS for domain decomposition on structured and unstructured meshes. The problem is also made more than 1 cell thick in the Z direction to demonstrate 3-dimensional computations.

There are three major steps in running TRUCHAS in parallel:

1. build a parallel executable
2. modify the input file
3. invoke the parallel executable

13.1 Building a Parallel Executable

A different executable has to be built for parallel execution. In a perfect world, the new executable can be built with two commands:

```
% make cleaner
% make all-parallel-debug
```

`-opt` can be substituted for `-debug`. It is essential that the correct MPI header files (mpi.h) and libraries are found during the build. MPI locations vary with the operating system, from machine to machine, often according to the whim of the system administrator. This point is particularly important if you have multiple MPI implementations installed on your computer. TRUCHAS attempts to guess where MPI is installed, using vendor defaults if possible. The locations of the MPI header files and libraries are specified in `src/options/libraries`. If you have any trouble building the executable, look there to make sure you have the correct paths for your MPI implementation.

13.2 Modifying the Input File

The input file for this problem is presented in Appendix L. In this chapter we will only discuss the aspects of this file critical to running the conduction problem in parallel.

A common technique for "commenting" sections of an input file is demonstrated here. Note that there is a namelist named PARALLEL_PARAMETERS and one named xPARALLEL_PARAMETERS. TRUCHAS looks

for a namelist called PARALLEL_PARAMETERS and ignores the namelist with the "x" in the name. The xPARALLEL_PARAMETERS is effectively a comment.

In the simplest case, no modifications to the input file are necessary at all! Running a parallel executable on n processors will result in the problem being decomposed into n subdomains, using Chaco. Chaco is a mesh decomposition/graph partitioning package from Sandia National Laboratory. Chaco is distributed with TRUCHAS.

Sometimes more control over the parallel parameters is desired. For these occasions, we have the PARALLEL_PARAMETERS namelist. A PARALLEL_PARAMETERS namelist can be added to any input file. Serial computations will ignore this namelist; parallel computations will use it.

The simplest PARALLEL_PARAMETERS is shown below:

```
&PARALLEL_PARAMETERS
   partitioner                          = 'chaco'
/
```

Using chaco as the partitioner choice means TRUCHAS will always be able to decompose the mesh into subdomains. This is the default if a PARALLEL_PARAMETERS namelist is not supplied, and is enough to get started. Other choices will be described later.

13.3 Invoking the Parallel Executable

There are many ways to run a parallel executable. Different MPI implementations use different commands. Large parallel machines often have a queuing system you have to cope with. In this example, we assume you are running on a single machine with multiple CPUs. MPI on clusters of computers connected via a network usually require a machines file or an application schema to describe the machines available in the cluster and how to map the processors to the machines. If you want to use clusters of computers, you'll need to look at the documentation for the MPI that you're using.

The next step is to run the problem. The input file resides at

```
UMProblems/Chapter_pll_hc/pll_hc.inp
```

Most MPI implementations come with a launcher program that starts the multiple processes and sets up the communication channels. This launcher is often called mpirun. The man page for mpirun on Irix describes the situation succinctly:

However, several MPI implementations available today use a job launcher called mpirun, and because this command is not part of the MPI standard, each implementation's mpirun command differs in both syntax and functionality.

If you have more than one MPI implementation installed on your computer, it is essential that you use the launcher program that goes with the MPI implementation that you built TRUCHAS with.

On many machines the following command is sufficient to launch the parallel computation:

```
% mpirun -np 2 truchas parallel.inp
```

The option "-np 2" sets the number of processes to 2. It generally isn't a good idea to specify more processes than you have available processors. "-np 1" is useful if you want to use a parallel executable to run a serial computation (instead of building a serial binary).

In the next sections, we'll discuss starting a parallel computation on some common parallel computing configurations.

13.3.1 Linux/LAM

Redhat includes LAM MPI as part of their Linux distribution. This makes using LAM MPI with Redhat Linux very easy. LAM is the default MPI choice for TRUCHAS on Linux.

LAM first requires the user to start the LAM daemons (using lamboot), which LAM then uses for process creation and control. After a parallel computing session, it is the user's responsibility to stop the LAM daemons (using lamhalt). A typical LAM session would consist of commands like the following:

```
% lamboot lamhosts
% mpirun -np 2 truchas parallel.inp
% mpirun -np 2 truchas parallel.inp
% lamhalt
```

Any number of mpirun commands can be executed while the LAM daemons are active. Attempting to use mpirun without having started the daemons will result in an informative error message.

13.3.2 Linux/MPICH

Many sites that run Linux install MPICH MPI (from Argonne National Laboratory) on their parallel computers. MPICH is often considered the reference MPI implementation, and can be installed on many different systems. The following command is all that's needed to run parallel programs using MPICH:

```
% mpirun -np 2 truchas parallel.inp
```

mpirun will need to connect to the local machine (using rsh or ssh) to start the parallel processes. A mechanism for making the connections without requiring a password is usually desirable. Doing this in a secure manner with rsh is difficult. When using ssh, the ssh agent and RSA authentication can be used. See the MPICH documentation and the man pages for ssh and ssh-agent for details

13.3.3 Tru64 and Compaq's MPI

Compaq supplies an MPI implementation for Tru64. No special machine configuration is necessary. The launcher is named dmpirun. The following command is all that's needed to run parallel programs on Tru64:

```
% dmpirun -np 2 truchas parallel.inp
```

13.3.4 Irix and SGI's MPI

From the user's perspective, running in parallel on SGIs running Irix is easy. They have an mpirun command and don't need any special magic. The following command is all that's needed to run parallel programs using SGI's MPI:

```
% mpirun -np 2 truchas parallel.inp
```

For this to work the system administrator needs to install and configure the array services daemon.

13.3.5 AIX and IBM's MPI

AIX has a different scheme for running parallel executables. They use the Parallel Operating Environment, or POE. Computational nodes on an AIX cluster are divided into "pools." The pool configuration can be

found with:

```
% js
```

On the AIX machines that TRUCHAS has access to, pool 0 is used.

To launch a parallel executable on an AIX cluster, a command like the following is used:

```
% poe truchas parallel.inp -rmpool 0 -nodes 1 -procs 2
```

This will allocate 1 machine out of pool 0, and use 2 processors on that machine. To use 16 machines with 4 processors on each (64 processors total), use a command like this:

```
% poe truchas parallel.inp -rmpool 0 -nodes 16 -procs 4
```

13.4 Modifying the Input File, continued

TRUCHAS-generated structured meshes (*i. e.*, not from a mesh file) can be decomposed using a simple block-structured scheme. Note, Chaco will also work in this situation, but the result will not likely be blocks with flat faces and smooth edges. An example is the easiest way to understand the block decomposition method. Assume we are trying to solve a problem on an orthogonal mesh, with 12 cells in the X and Y directions, and 4 cells in the Z direction. This would be specified in the `MESH` namelist by:

```
&MESH
   ...
   ncell                                = 12, 12, 4
   ...
/
```

Suppose we have a 4 processor machine and we want to decompose this problem into 4 simple block subdomains, with each block being 6x6x4 cells. We can specify a `processor_array` in the `PARALLEL_PARAMETERS` namelist:

```
&PARALLEL_PARAMETERS
   partitioner                          = 'cartesian'
   processor_array                      = 2, 2, 1
/
```

Note that the `partitioner` is now specified as `'cartesian'`. The result of this will be to cut the orthogonal mesh into two pieces in the X direction and two pieces in the Y direction, as in Figure 13.4.

Figure 13.2 shows the 4 subdomain decomposition of the same mesh produced by Chaco. Note that while the subdomains are of approximately equal size (as in the cartesian decomposition), the boundaries between subdomains are not as straight.

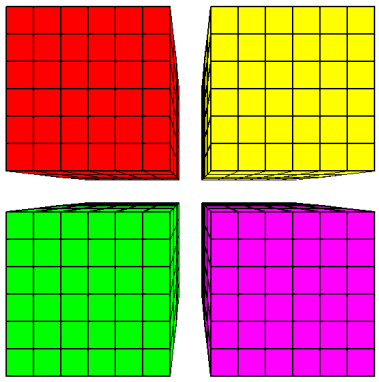


Figure 13.1: The orthogonal mesh is cut into into two pieces in the X direction and two pieces in the Y direction when the 'cartesian' partitioner is chosen.

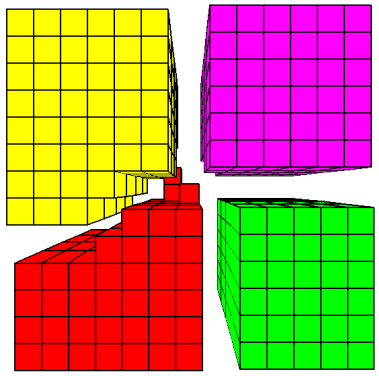


Figure 13.2: The orthogonal mesh is cut into four different subdomains when the 'chaco' partitioner is chosen.

Chapter 14

Output

14.1 Overview

This chapter describes the outputs generated by TRUCHAS upon execution of a problem. In this version a major change has occurred with the output module. The TRUCHAS code now outputs XML data which is parsed by a Python postprocessor. Note that Python version 2.5 or higher is required along with the the Python `numpy` package to successfully parse the XML.

As a result of the output overhaul TRUCHAS now produces the following types of output files:

1. *prefix*.TBrook.xml:provides a summary of the simulation specifications, mesh(es), time step characteristics, and field variables outputted in each time step (Section 14.3). It also supplies references to the XML file containing the mesh information and binary files containing data associated with the field variables.

2. *prefix*.*meshname*.*specifier*.xml:provides a summary of the mesh variables associated with the mesh *meshname* and a reference to the binary file containing data associated with this mesh's variables (see next item). Typically *meshname* is 'DefaultMesh' and *specifier* is a numeric label which is intended to refer to a cycle number. Since the mesh does not currently alter during TRUCHAS simulations, *specifier* is always '00001'.

3. *prefix*.*meshname*.*specifier*.bin:provides the binary file containing the data describing this mesh's variables (i.e cell centroids, cell-vertex connectivity etc).

4. *prefix*.ts.*timestep*.bin:provides the binary file containing the data describing all the field variables outputted in this particular time step.

5. *prefix*.pb.*probenumber_fieldvariable*.bin: provides the binary file containing data for field variable *fieldvariable* at the location of probe *probenumber* for all TRUCHAS cycles. See the `PROBE` section of the `Reference Manual` for instructions on defining probes in the TRUCHAS input file.

6. *prefix*.out:provides the screen output (stdout) of the TRUCHAS simulation.

7. *prefix*.err:provides any error output (stderr) of the TRUCHAS simulation.

Here *prefix* is replaced with the root name of the input file (*e. g.* if `hc.inp` is the input file for a run, "`hc`" will be the prefix).

In the following we will first describe the input variables that control the kind and amount of outputs and then discuss in more detail the different output data generated by TRUCHAS.

14.2 `OUTPUTS` Namelist Variables

Output options in TRUCHAS are specified via input variables in the `OUTPUTS` Namelist Group. See the `OUTPUTS` section of the `Reference Manual` for definitions of all the variables. Here we simply highlight some of the more common variables to illustrate their usage.

The variable `output_t` specifies a sequence of time values at which outputs will be generated. Typically it consists of the problem beginning and end times, *e. g.*,

```
output_t = 0.0, 10.0
```

However, multiple time segments can also be defined to allow more flexible control of outputs, e.g.,

```
output_t = 0.0, 5.0, 10.0
```

In this example, two time segments are defined for outputs. This may be useful in problems in which fewer outputs are needed in a slow developing transient within the first time segment, but more outputs, at finer time intervals, are desirable within the second time segment when much of the interesting physics occurs.

Within each time segment, outputs are generated according to the variable `output_dt`. For the two-time-segment example above, we may have

```
output_dt = 2.5, 1.0
```

which will result in outputs at time values 0.0, 2.5, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0.

The amount of output produced is further controlled by an integer multiplier, which is assigned to the `XML_output_dt_multiplier` variable to obtain the actual time interval for generating TRUCHAS time step output. The default value for this multiplier is one. If the `XML_output_dt _multiplier` is given a value of -1, then time step output data will be written at every computational cycle.

To then obtain graphics formats `GMV`, `TECPLOT`, `ENSIGHT`, restart files, and diagnostics, one must post-process the XML output as detailed in Section 14.4. `GMV` is a LANL graphical postprocessor that is available free of charge at `http://laws.lanl.gov/XCM/gmv/GMVHome.html` while `TECPLOT` (`http://www.amtec.com`) and `EnSight` (`http://www.ensight.com`) are commercial visualization software programs for engineering and scientific computations.

14.3 The TBrook.xml File

The *prefix*.TBrook.xml file contains information on the executable program, *i. e.*, code version number, build date, platform, and on the simulation run which includes date and computer platform of execution, the number of processors, etc. It also echoes some of the relevant problem set-up information, such as domain and mesh sizes, boundary conditions, and initial materials. Time advancement (cycle number, time step size, etc.) and solver iteration information is also written to this file as the calculation progresses. If the problem completes normally, a CPU timing summary is given, with breakdowns for various procedures within TRUCHAS. Note that most of the above output data are also printed to the Unix standard output unit (*i. e.*, terminal screen or tty).

If specified (via the respective `output_dt` multipliers), short and long edit data are also written to this XML file. Short edit data include total mass, volume, momentum, kinetic energy, and enthalpy for each material in the problem, as well as the minimum and maximum values of important solution quantities. Long edit data are values of all relevant solution quantities at cells within a region defined by `long_edit_bounding_coords`. Depending on the problem, the long-edit solution quantities include cell density, pressure, temperature, velocity components, material volume fractions, von Mises equivalent and mean stresses, and total volumetric strain.

14.4 Postprocessing the XML Output

The TRUCHAS repository provides a capable Python script to postprocess the XML output. This Python postprocessor is located in the TRUCHAS repository at

```
tools/scripts/TBrookParse.py
```

In order to become familiar with the new output capability the user is first encouraged to run the posptprocessor interactively by typing

```
python user_tdir/truchas/tools/scripts/TBrookParse.py,
```

(see Section 14.4.1). The directory containing the TRUCHAS repository is `user_tdir`. An interactive prompt is provided to guide the user into producing either graphics or restart files or interrogating physical regions in the domain. We suggest trying all examples in this chapter using the interactive mode first (that is by typing one command at a time, in response to the prompt) before typing multiple commands at the prompt.

With the new TRUCHAS postprocessor graphics, restart and diagnostic capabilities are more flexible. For example, within a given postprocessing session, multiple TRUCHAS simulations can be loaded and multiple graphics and restart files produced. In addition, restart files can be generated on meshes different from what was employed in the original execution. The user also has far more flexibility to choose and analyze physical regions in the domain. For example, a user can obtain maximum, minimum, averages and outputted data of particular field variables in a region defined by combinations of indices, mesh blocks, spatial coordinates, and variable values (see Section 14.4.3).

Once familiar with the postprocessor, for efficiency the user should run the postprocessor commands in batch mode (see Section 14.4.13).

14.4.1 Using the postprocessor interactively

The interface to the Python postprocessor is prompt-driven. That is, at every turn the user is presented with a prompt containing appropriate default choices with error checking on the resulting choice. The default choice will be accepted when a return or a minus sign (-) is typed. In addition, you may 'type ahead'. That is, if you know what prompts are coming up, and their default values or the values you want, you may type commands ahead of time so that you will not be presented with a prompt for those values. One current caveat is that any choice that has a space in it must be enclosed by double quotes (*i. e.*, "`region 1`").

To invoke the postprocessor, type

```
python user_tdir/truchas/tools/scripts/TBrookParse.py
```

this results in the following prompt:

```
Try "load" or "help"
```

```
Command:()
```

If you choose “load” , *i. e.*,

```
Command:() load
```

you will be asked the name of the XML file to be read. If you invoked the Python postprocessor in a directory containing a `.TBrook.xml` file, the default file choice will be that XML file. For purposes of illustration throughout this chapter, assume this choice is `vp.TBrook.xml`. If however you invoked the postprocessor in a directory without a `.TBrook.xml` file, the default choice will be

```
tools/scripts/test_TBrookParse/samples/phasechange_mixed_output/
phasechange_mixed.TBrook.xml
```

which is a test simulation output stored in the TRUCHAS repository.

If you choose “help” at the postprocessor command prompt, *i. e.*,

```
Command:() help
```

the following list of all available postprocessor commands and a brief explanation of them will be produced:

```
Commands are:

['load', 'quit', 'stat', 'statistics', 'end', 'help', 'clean',
'h', 'region', 'probe', 'list', 'deleteAllRegions', 'q', 'write',
'build', 'timeseries', 'query', 'e', 'restart', 'define']
```

Each of the commands will be explained in detail in upcoming sections.

Finally, to quit the postprocessor type `quit`, `q`, `end` or `e`.

14.4.2 List

The user may load multiple files and define multiple regions within one postprocessing session. Hence the `list` command,

```
Command:() list
```

is useful in tracking what files have been loaded, field variables in these files, and regions defined by the user.

Choosing `list files`, *i. e.*,

```
Command:() list files
```

will produce the output

```
Loaded files are:

vp.TBrook.xml
```

Choosing `list variables` will provide a list of variables and cycles in which they occur for each loaded file.

Choosing `list regions` will provide a list of default spatial regions that are automatically set when a file is loaded and a list of regions that the user has specifically defined (see Section 14.4.3 to understand how to define a region). As we have not yet defined a user specified region, the result of

```
Command:() list regions
```

is

```
Regions are:

  allMESH_1 : defined for vp.TBrook.xml on mesh DefaultMesh as
              ['CELL','FACE','VERTEX'](s) with all indices defined.
```

`allMESH_1` indicates that the entire (1)st (and only) mesh in the simulation `vp.TBrook.xml` is a region that can be interrogated. TRUCHAS currently outputs only one mesh (labelled `DefaultMesh`). However, in future, multiple meshes will be outputted. In such a case, multiple regions will be automatically defined and labelled as `allMESH_2`, `allMESH_3` etc.

14.4.3 define:Defining regions

By loading a file into the postprocessor the user automatically has the entire mesh (`allMESH_1`) to interrogate. The user may however want to create a subset of the mesh, *i. e.* his own region to interrogate. The command

```
Command:() region
```

or the command

```
Command:() define
```

allows the user to do this. A user-defined region is a set of cells and/or vertices that satisfy certain criteria. Currently a region is defined using combinations of the following selector criteria:

1. `ID` specification; here the user provides a list or range of cell indices or vertex indices.
2. `VAR` specification; here the user defines a region based on the data values of a particular variable. The region will alter as the variable values change throughout the simulation.
3. `S` specification; here the user provides a box-shaped spatial domain using the specification

   $[x1, y1, z1] \rightarrow [x2, y2, z2]$

   where

   $[x1, y1, z1]$ is the point closest to the origin, and

   $[x2, y2, z2]$ is the point farthest from the origin.

   The result will be a list of cell and vertex indices that lie within this spatial domain.
4. `MB` specification; here the user provides an integer mesh block number which in turn provides a list of cells belonging to this mesh block number. This list comes directly from data in the TRUCHAS output— if the output does not contain any mesh block data then this specification cannot be used.

To define a region employ the command

```
Command:() region
```

or

```
Command:() define
```

To define a region named R_01 read from vp.TBrook.xml using a combination of both the index ID and variable VAR selector criteria type

```
Command:() region vp.TBrook.xml R_01 DefaultMesh ID,VAR
```

At this point, the user will be questioned further about the ID selector. He will be asked if he wants to define his region based on CELL ID or VERTEX ID, to provide a range of indices and to verify the region selected. In this particular example a region will first be defined to contain CELL indices 20 to 40. The required sequence of commands are

```
Command:() CELL 20-40 y
```

(the y command at the end results from the postprocessor request to verify this criteria used to specify the region).

At this point the user will be questioned about the VAR selector. He will be asked what variable and its range of values he would like to use to further define the region. In this particular example, the region will be defined to also contain indices where the pressure P is between 0.5 and 1.0. The required sequence of commands are

```
Command:() P 0.5 1.0 y
```

(again the y command at the end results from the postprocessor request to verify this criteria used to specify the region).

The region is now defined. Note that for efficiency, the user can simply write out all required commands at a single prompt,

```
Command:() region vp.TBrook.xml R_01 DefaultMesh ID,VAR CELL
          20-40 y P 0.5 1.0 y
```

An example series of commands showing how to define a region named R_02 read from vp.TBrook.xml using both the index ID and spatial S selector criteria is

```
Command:() region outputfile R_02 DefaultMesh ID,S
           VERTEX 10-30 y [0.01,0.01,0.01]->[0.05,0.05,0.05] y
```

As a result of the ID and VERTEX 10-30 commands, the region will contain vertices with indices in the range 10-30 (inclusive). As a result of the S and [0.01,0.01,0.01]->[0.05,0.05,0.05] commands the region will also contain all cells and vertices whose positions lie within the box-shaped spatial domain

$$[x1 = 0.01, y1 = 0.01, z1 = 0.01] \rightarrow [x2 = 0.05, y2 = 0.05, z2 = 0.05]$$

Invoking the command 'list regions' now results in the following output:

```
Regions are:

allMESH_1 : defined for vp.TBrook.xml on mesh DefaultMesh as
            ['CELL', 'FACE', 'VERTEX'](s) with all indices defined

R_01      : defined for vp.TBrook.xml on mesh DefaultMesh as
            CELL(s) with IDs: 20-40
R_01      : defined for vp.TBrook.xml on mesh DefaultMesh as
            CELL(s) with P between 0.5 and 1.0

R_02      : defined for vp.TBrook.xml on mesh DefaultMesh as
            VERTEX(s) with IDs: 10-30
R_02      : defined for vp.TBrook.xml on mesh DefaultMesh as
            ['CELL', 'FACE', 'VERTEX'](s) with spatial domain:
            [0.01,0.01,0.01]->[0.05,0.05,0.05]
```

14.4.4 query:Interrogating field values on a region

Having defined regions, the user is able to query field values on those regions, using the command

```
Command:() query
```

The result can be either output to a text file (using the command file) or output to the screen (using the command screen). In the following example the value of the pressure P read from vp.TBrook.xml for all cells in region R_01 for cycles (0-10) (inclusive) results from the following command:

```
Command:() query vp.TBrook.xml screen P R_01 0 10
```

The resulting output is to the screen and looks like:

```
#-------------------------------------------------------------
#       VAR                 TIME                      CYCLE
#        P                  0.000                        0
#-------------------------------------------------------------
#     Index             Position                  Value(s)
        21         2.50,   5.00,   5.00      1.000e+00
        22         7.50,   5.00,   5.00      1.000e+00
        23        12.50,   5.00,   5.00      1.000e+00
etc
etc
```

When querying a vector field (such as `Velocity`) all components of the vector field are presented. The command

```
Command:() query outputfile screen Velocity R_01 - -
```

provides three Velocity components for all cells in region `R_01` using the default beginning and ending cycle limits (`- -`). The resulting output looks like..

```
#-----------------------------------------------------------------------
#       VAR                 TIME                      CYCLE
#    Velocity              0.000                        0
#-----------------------------------------------------------------------
#     Index             Position                  Value(s)
        21       2.50,   5.00,   5.00     6.073e-03 -6.035e-03  0.000e+00
        22       7.50,   5.00,   5.00     6.073e-03 -6.035e-03  0.000e+00
        23      12.50,   5.00,   5.00     6.073e-03 -6.035e-03  0.000e+00
etc
etc
```

14.4.5 stat:Statistics of field variables on a region

By typing the command

```
Command:() statistics
```

or

```
Command:() stat
```

the user can create a set of statistics of a field variable on a region. The results may be outputted to the screen or a file. The statistics currently calculated are maximum, minimum, average, standard deviation, indices and positions indicating where the maximum and minimum are located. In the following example, statistics for `del-rho` read from `vp.TBrook.xml` in the region defined as `R_01` for cycles `(0-1)` (inclusive) are presented to the screen. `del-rho` is a scalar so it has only 1 component (`COMP`). The command is

```
Command:() stat vp.TBrook.xml screen del-rho R_01 0 1
```

which results in the following output

```
-----------------------------------------------------------------
                    TIME:    0.000       CYCLE:        0
-----------------------------------------------------------------
COMP  AVG  MIN   MIN_at (Index,Posn)  MAX   MAX_at (Index,Posn)  STDEV
  1   0.0  0.0   1 [0.0,0.0,0.0]      0.0   1 [0.0,0.0,0.0]      0.0


-----------------------------------------------------------------
                    TIME:    0.100       CYCLE:        1
-----------------------------------------------------------------
COMP AVG  MIN   MIN_at (Index,Posn)   MAX    MAX_at (Index,Posn)    STDEV
 1   0.0  0.0   1 [0.0,0.0,0.0]       0.0    1 [0.0,0.0,0.0]        0.0
```

Statistics for a vector field (such as `Velocity`) are presented for all components of the vector field as shown in the example below. The command

```
Command:() stat vp.TBrook.xml file stat.txt Velocity R_01 0 0
```

provides statistics for all components of the Velocity for cycles `0-0` to a file `stat.txt`. The following output to the file `stat.txt` results:

```
------------------------------------------------------------------
                    TIME:    0.000        CYCLE:        0
------------------------------------------------------------------
COMP AVG  MIN MIN_at (Index,Posn)  MAX  MAX_at (Index,Posn)  STDEV
 1   0.0  0.0   1 [0.0,0.0,0.0]      0.0   1 [0.0,0.0,0.0]      0.0
 2   0.0  0.0   1 [0.0,0.0,0.0]      0.0   1 [0.0,0.0,0.0]      0.0
 3   0.0  0.0   1 [0.0,0.0,0.0]      0.0   1 [0.0,0.0,0.0]      0.0
```

14.4.6 write:Creating visualization files

Within a postprocessing session the user may create `GMV`, `Tecplot`, `EnSight` and/or `VTK` files. To do this, choose the `write` command, *i. e.*,

```
Command:() write
```

To commence writing a `GMV` file from `hc.TBrook.xml` type at the command line prompt

```
Command:() write gmv hc.TBrook.xml
```

To commence writing an `EnSight` file from `vp.TBrook.xml` type

```
Command:() write ensight vp.TBrook.xml
```

To commence writing a `Tecplot` file from `vp.TBrook.xml` type

```
Command:() write tecplot vp.TBrook.xml
```

and to commence writing a `VTK` file from `vp.TBrook.xml` type

```
Command:() write vtk vp.TBrook.xml
```

At this point the user will be prompted as to which mesh to visualize. As TRUCHAS currently outputs only one mesh the only choice will be `DefaultMesh`. In the future TRUCHAS will output multiple meshes so this choice will be extended. A choice of TRUCHAS time step cycles will then be presented to create the

visualization file(s) from. Type `-1` to create visualization files for all available output time steps, otherwise a specific output cycle must be chosen. Currently there is no way to select a few cycles but this will be incorporated in the next release. The choice of output format of the visualization file(s) (`ascii` or `binary`) is next presented.

For example, to write binary `GMV` files for all cycles in the the `vp.TBrook.xml` file type

```
Command:() write gmv vp.TBrook.xml DefaultMesh -1 binary
```

For all other visualization choices other than `EnSight`, the visualization file name is the next presented choice. If all time steps are chosen (i.e `-1`), a root file name choice is presented and all resulting visualization files will have the name `rootname.vis.000*` where `vis` is one of `gmv`, `vtk`, `ensight` or `tecplot` and `000*` represents the TRUCHAS output cycle number(s). Thus, the command,

```
Command:() write gmv vp.TBrook.xml DefaultMesh -1 binary vpvis
```

will ensure that `GMV` files `vpvis.gmv.000*` are produced.

For the `EnSight` choice, a `simulationname.ensight.CASE` and a `simulationname .ensight` directory will be produced. Here `simulationname` is automatically derived from the name of the `TBrook.xml` file being interrogated (in this particular example `simulationname` is `vp`). Thus to create binary `EnSight` files type the command

```
Command:() write ensight vp.TBrook.xml DefaultMesh -1 binary
```

Finally, to prevent unnecessary storage, the user will next have a choice as to which TRUCHAS field variables will be placed in the visualization file(s). The user has the following options at this point

- to choose a particular list of variables; the list should be in the form

  `T,P,density`

  (note, no spaces after the commas)
- to choose the (default) `truchas` option (which will write out all possible variables except sensitivity and solidification diagnostic variables)
- to choose the `all` option (which will write out all field variables).

Continuing on from the above GMV example then the command

```
Command:() write gmv vp.TBrook.xml DefaultMesh -1 binary vpvis truchas
```

will write out the truchas list of variables to the vpvis.gmv.000* gmv files.

14.4.7 restart:Creating restart files

Within a postprocessing session the user may create multiple TRUCHAS restart files, either by reading variables from a TRUCHAS simulation or by mapping field variables to a new mesh. To create a restart file choose the restart command, *i. e.*,

```
Command:() restart
```

To commence writing a restart file from vp.TBrook.xml type at the command line prompt

```
Command:() restart vp.TBrook.xml
```

At this point the user has the option of choosing a standard or a mapped restart. A standard restart is produced by writing the TRUCHAS field variables from the current simulation. A mapped restart is produced by mapping TRUCHAS field variables onto a new mesh and writing out this new mesh and the mapped fields.

If the standard option is chosen, the user will be prompted for which mesh to restart on (as usual DefaultMesh is the only option for now), what time step to get the fields to restart on (choose -1 to create restarts for all available TRUCHAS time steps), the format to write the restart file in (currently binary is the only appropriate choice), and the name of the restart file.

Thus to write a binary restart file vp.rst from vp.TBrook.xml for TRUCHAS cycle 10 type

```
Command:()restart vp.TBrook.xml standard DefaultMesh 10 binary vp.rst
```

If the mapped option is chosen the user can either map one TRUCHAS simulation to one Exodus II mesh or map two TRUCHAS simulations to one Exodus II mesh. In this release only Exodus II mesh files can be read to obtain the destination mesh. To create a mapped restart by mapping fields from vp.TBrook.xml on its DefaultMesh at time step 10 to an Exodus II mesh stored in file vp.exo type

```
Command:() restart vp.TBrook.xml mapped DefaultMesh 10 vp.exo
```

At this point the user will be prompted as to whether the destination mesh (`vp.exo`) coordinates should be rescaled. The default choice for this coordinate scaling factor is the mesh scaling factor employed in the original TRUCHAS simulation (see the TRUCHAS Reference Manual `MESH` section, in particular the variable `Coordinate_Scale_Factor`).

Finally the restart file format (`binary`) and restart file name are chosen.

Thus to map the `vp.TBrook.xml` simulation at cycle 10 to the destination mesh in file `vp.exo`, which is scaled by a factor `0.02`, type

```
Command:() restart vp.TBrook.xml mapped DefaultMesh 10 vp.exo
           0.02 binary vp.rst
```

Once you have completed this mapping process, a new object or 'file' will be created `'mapping1'`. This is viewed by typing

```
Command:() list files
```

which will produce the following output

```
Loaded files are:

vp.TBrook.xml
mapping1
```

This `mapping1` object exists only during the postprocessing session. During this session, it can be utilized just like a `TBrook.xml` file; that is, it can be used to create visualization files, diagnostics, and restart files. For example to create a binary `GMV` file `mapping1.gmv` from the `mapping1` object type the following command

```
Command:() write gmv mapping1 - 0 binary mapping1.gmv all
```

Note the default (`-`) choice of mesh is `ExoMesh` (rather than `DefaultMesh` as would be the case if a TRUCHAS XML file were used to generate the `GMV` file). The `mapping1` object contains the Exodus

destination mesh, which the postprocessor labels as `ExoMesh`. The `mapping1` object also contains a set of field variables which exist at a 'cycle' labelled as `0` by the postprocessor (all mapped objects will contain only 1 'cycle' labelled as `0`).

Currently, only CELL-based TRUCHAS fields are mapped. These fields can be mapped using two different constraints

1. a `CONSERVATIVE` rule (whereby the summation of the field is conserved during the mapping), OR
2. a `CONSTANTS_PRESERVING` rule (whereby a constant field and minimum and maximums are preserved during the mapping).

Refer to the *Truchas Physics and Algorithms* for further details.

To choose what field variables are to be mapped and their mapping rule, edit the `defaultmaps.txt` located in the repository at

```
tools/PythonPackages/TBrookParser/MODutils/MAPutils/defaultmaps.txt
```

When a field is mapped to a destination mesh with a larger domain than the source mesh, the field value will be mapped to zero in regions where the source and destination meshes do not overlap.

If two TRUCHAS simulations are mapped to one `Exodus II` mesh file, the process is similar except now two TRUCHAS XML output files, two source meshes, and two cycles must be specified. For example, to map the fields at cycle 10 from `vp.TBrook.xml` and the fields at cycle 20 from `vp2.TBrook.xml` to the destination mesh `vp.exo` in order to produce the binary restart file `vp.rst`, type

```
Command:() restart vp.TBrook.xml,vp2.TBrook.xml mapped
          DefaultMesh DefaultMesh 10 20 vp.exo 0.02 binary vp.rst
```

Note the syntax here: in specifying the two TRUCHAS XML files there is no space between the commas. Note too you could simply employ the default choices of the source meshes...

```
Command:() restart vp.TBrook.xml,vp2.TBrook.xml mapped
          - - 10 20 vp.exo 0.02 binary vp.rst
```

When mapping two simulations to a destination mesh, it is assumed that the source meshes do not overlap. If the source meshes do overlap, destination field values in the regions where the meshes overlap will be invalid.

14.4.8 probe:Interrogating probe values

TRUCHAS probe variables can also be interrogated by the postprocessor using the command

```
Command:() probe
```

See the PROBE section of the *Truchas Reference Manual* for instructions on defining a TRUCHAS probe variable. The probe name specified in the TRUCHAS input file is used as the designator of the probe. Note again the restriction of the prompt driven interface—if the probe name contains a space, it must be enclosed by double quotes. Multiple fields at the probe location in a specified time interval can be printed to the screen or a file. For example, assume that a probe `top right` has been specified in the TRUCHAS input file. The command to interrogate the `VOF0001` and `Velocity` fields at this probe is

```
Command:() probe vp.TBrook.xml ‘‘top right’’
           screen VOF0001,Velocity 0.01->0.03
```

This command produces the following output:

```
#Probe Name               : top right
#Probe Description        : Testing top right probe
#Probe Coordinates        : [1.04000E-02 2.53000E-02 1.06000E-02 ]
#Closest Cell Index       : 112
#Closest Cell Coordinates : [1.04813E-02 2.53629E-02 1.05994E-02 ]
#Closest Node Index       : 21
#Closest Node Coordinates : [7.84903E-03 2.41568E-02 1.27000E-02 ]
#Time         VOF0001      Velocity1  Velocity2  Velocity3   Cycle
2.00000e-02   0.00000e+00    0.02        0.01     0.0         1
```

The probe name, description, and location are taken directly from the TRUCHAS input file. The nearest cell index and coordinates and node index and coordinates to the probe (calculated within the TRUCHAS simulation) are also presented. Finally the components of the chosen fields are presented at the times and cycles corresponding to the requested time range.

14.4.9 timeseries:Creating a time series of a field variable

There are instances where a user requires a time series of a chosen variable at a particular index, printed to the screen or to a file. The postprocessor command

```
Command:() timeseries
```

allows the user to do this. For example, the command

```
Command:() timeseries vp.TBrook.xml Velocity 200 0 7 screen
```

will output a time series to the screen of the `Velocity` variable at index `200` for TRUCHAS output cycles in the range `(0-7)` inclusive.

14.4.10 build:Building Python extension modules

The Python postprocessor creates a number of Python extension modules. These modules, written in `C` and `F90`, are compiled as shared objects which can then be imported into the Python code. They are needed in order to write ASCII and binary files efficiently and to interface to the TRUCHAS grid mapping and exodus module `F90` code to provide the restart mapping capability. In a majority of cases the user will only need to create these modules once. It may however be useful at times to recreate these modules by recompiling. The postprocessor command

```
Command:() build compilername
```

will build all modules, using compiler `compilername` (i.e `lahey`, `nag` etc). Please ensure when choosing the compiler, your environment is consistent as detailed in the *Truchas Installation Guide* . This command will be automatically executed during the TRUCHAS build process.

14.4.11 clean:Removing Python extension modules

The `clean` command removes all the existing Python extension modules.

```
Command:() clean
```

14.4.12 deleteAllRegions:Delete user-specified regions

The command

```
Command:() deleteAllRegions
```

will remove all user-specified regions for all loaded files, leaving only the automatically created `allMESH_1` regions (see Section 14.4.3 to create a region).

14.4.13 Using the postprocessor in batch mode

Once the user is comfortable with the interactive mode it is recommended that the user place his commands to the postprocessor in a macro file `filename.mac`, and simply run the postprocessor in batch mode using the `-f` option on the Python command line,

```
python user_tdir/truchas/tools/scripts/TBrookParse.py -f filename.mac.
```

For example consider the command employed in Section 14.4.7 to map two TRUCHAS simulations to an `Exodus II` mesh,

```
Command:() restart vp.TBrook.xml,vp2.TBrook.xml mapped
          - - 10 20 vp.exo 0.02 binary vp.rst
```

These commands could also be placed in a macro file `filename.mac` as shown below

```
restart
vp.TBrook.xml,vp2.TBrook.xml
mapped
-
-
10
20
vp.exo
0.02
binary
vp.rst
```

Example `*.mac` files can be found in

```
user_tdir/truchas/tools/scripts/test_TBrookParse/*.mac.
```

–

Chapter 15

Continuing a Run from a Restart Dump

15.1 Overview

Often a simulation cannot be completed to the desired problem time in a single run/job. In this case TRUCHAS allows the user to restart the simulation from the results of a completed run. The mechanism for this is a restart file, which can be created by the postprocessor from one of the binary dump files written during code execution. Steps to restart a run are:

1. Run TRUCHAS in an initial job, producing the binary dump files required by the Python postprocessor as described in Section 14.1, item 4. If `XML_output_dt_multiplier` is nonzero, the code will produce binary output files at the end of each cycle in which problem time reaches the given multiple of `output_dt`. In any case the default is to write a final binary dump, provided the code exits gracefully. If the code does not exit gracefully, and if at least one dump file was written during execution, any one of them can be used to create a restart file.

2. Run the Python postprocessor as described in Section 14.4.7 to read the desired binary dump, create a restart file, and write it out. Standard and mapped restarts are produced in different ways.

3. Copy the previous input file and edit the copy as will be described below.

4. Make sure that the new edited input file and the restart file are in the current directory (or refer to them appropriately).

5. Start another TRUCHAS job, naming the restart file after the `-r:` command line option.

15.2 Restart Considerations

A restart file—whether created as a standard or mapped restart— contains headers and then a sequence of variable fields that are read in by TRUCHAS during the initialization of a restarted run. One might think that these fields would be sufficient for TRUCHAS to initialize. In fact a sequence of things happens when TRUCHAS starts up and finds a `-r:` option on the command line. The most important thing is that a complete input file containing material specifications and boundary conditions, physics switches, solver definitions, etc. must be present to restart the code. Even a mesh file must be present, because TRUCHAS reads the number of cells and nodes from that file, not from the restart file. It also reads all side sets from the mesh file and then knows their IDs, so the BCs must refer to the same side sets as the initial run.

The reason that TRUCHAS does not presume to carry over the physics switches (which types of physics are activated and deactivated) from the restart file is that we have found that we often wish to change the switches upon restarting. For example at the end of induction preheating of a casting assembly, current to the coils might be turned off before the crucible is emptied to fill the mold. We would then restart the code with EM off and flow on.

Material fields, particularly the last known volume fractions of materials in all cells, are read from the restart file. However, all material properties are read from the input file and reinitialized. Here again one might want to change them to track some physical effect. In the restart run's input file the materials namelists must be in the original order, the same as in the initial run, because materials are numbered internally in the order in which they appear in the input file.

Similarly boundary conditions for the restarted run must be set exactly as if it were an initial run. An example of why one must do this is if flow had been initiated with an inflow BC as in the above flow sample problem (Chapter 6). At some time inflow might stop, but one wishes to continue tracking flow in a mold or other container. In the restart run, the BC for the inflow surface would be changed to another type.

Sequences of calculations connected by restarts must use a consistent enthalpy-temperature relationship, because cell enthalpy is transferred by the restart file and used to recompute and re-initialize temperatures. This usually requires that phase-change models not be changed between such calculations. For certain problems this is not suitable. We are considering adding an option in the next release to allow making temperature the primary field variable upon restart; enthalpies would then be recomputed and re-initialized instead.

15.3 Options for Restarts: the RESTART namelist

Namelist instances allowed: single

```
!! Example only -- not a paradigm!
&RESTART
  ignore_dt               = .true.
  ignore_Joule_heat       = .false.
  ignore_solid_mechanics  = .false.
  temperature_is_primary  = .false.
/
```

The first of these options is commonly used. It allows the user to choose for the first cycle of the restarted run a different time step from the final time step of the previous run. Particularly if other parameters have been changed, it may be prudent to restart with a smaller time step and let it grow back on subsequent cycles.

The last three options direct TRUCHAS to ignore certain field variables inherited from the previous run through the restart file. For each of them the default value is `.false.`.

- If `ignore_Joule_heat` is `.true.`, and Electromagnetics (EM) is activated, this option will force a new EM solve to recompute the Joule heating rate distribution. If it is defaulted or set to `.false.`, an EM run (heat-up) can be continued with the inherited Joule heating rate distribution.
- If `ignore_solid_mechanics` is `.true.`, and solid mechanics physics is activated, the stresses and strains will be recomputed, ignoring those in the restart file.
- If `temperature_is_primary` is `.true.`, the initial value of the enthalpy field will be calculated from the restart file temperature values. The default action is to calculate the initial value of the temperature field from the restart file enthalpy values. This option can be useful if some material properties or phase changes have been changed in the input file for the restart run. It also allows a better approximation to reality when restarting on a newly mapped mesh (Section 14.4.7).

Another namelist whose parameters are commonly changed upon restart is the `OUTPUTS` namelist. As phenomena change with the evolution of the system, one might desire more or less frequent outputs, or even start a different type of output.

Chapter 16

Caveats

In this chapter we highlight some algorithmic issues in the application of TRUCHAS. The list below is by no means exhaustive and primarily covers caveats and limitations that will be addressed in future code releases.

16.1 Use of Discrete Operators

Care must be taken in employing the LSLR nonorthogonal form of the discrete operator in the thermal conduction solution. The reasons for this are two-fold. First, the heat transfer coefficents are currently incompatible with the LSLR operator. Second, employing LSLR in thermal conduction problems with voids results in inaccurate thermal gradients, because the stencil for LSLR is large and does not account for the presence of void cells with arbritrary temperature. Hence the results will vary as the void temperature varies. Energy transfers inconsistent with real thermal gradients have also been observed in these circumstances. We note however that the use of the pseudo-ortho discrete derivative approximations produces incorrect answers in highly distorted grids. Version 2.1 of TRUCHAS introduces the Support Operator (SO) discrete approximation for calculating face gradients of temperature and pressure adjustment in the heat transfer and fluid flow calculations. This approximation is accurate and does not suffer from the stencil limitations of the LSLR method but may increase computing times.

The evaluation of face-centered velocity components is always performed as an average of the cell-centered velocities on either side of the face, even if the value of `discrete_ops_type = 'nonortho'`. This choice was made because the use of LSLR to evaluate face-centered velocity components can give inaccurate values. However the use of this simple averaging produces incorrect answers in highly distorted grids. We will continue to research this issue—suffice to say that the use of `discrete_ops_type = 'nonortho'` is not recommended for any problem that involves interfacial fluid flows with large density differences. The Support Operator approximation is a new implementation

that should give more accurate results than the ortho approximation in distorted grids.

16.2 Volume Tracking and Reconstruction

The current volume advection algorithm is a simple unsplit scheme. Such schemes are subject to errors due to overlapping flux volumes. This problem is often termed the 'corner coupling' problem, as typically it is volume fluxes near the corners that are erroneously fluxed, leading to the distortion of cell volumes. In TRUCHAS one option to alleviate this problem is to employ subcycling within the volume advection. Subcycling however leads to numerical errors that are particularly severe near the interface between liquids and voids. These errors occur because momentum and enthalpy advection is no longer consistent with volume advection. While volume advection occurs after each subcycle, the momentum and enthalpy advection occur only after a set of subcycles. To be consistent, the momentum and enthalpy flux associated with the volume flux should vary within these subcycles.

Reconstruction of fluid interfaces does not account for the velocity or acceleration of the fluid. This can result in a "stranded" piece of fluid that is located by reconstruction at an edge of a mesh cell. An example is a stranded piece of fluid located at the top of a mesh cell with a downward gravitational force applied. As the CFL stability limit keeps the product of fluid velocity and Δt below some fraction of the cell size, the result is an acceleration of the cell velocity without any associated movement of the "stranded" fluid. This leads to Δt being inversely proportional to the fluid acceleration and hence inversely proportional to time. In turn this leads to very long computational times. A new option in Version 2.1 may be used to limit the acceleration of fluid beyond physically reasonable values. The `mechanical_energy_bound` input variable in the `NUMERICS` namelist inhibits body force acceleration in cells whose mechanical (potential plus kinetic) energy exceeds its value.

16.3 Fluid Flow Algorithm

The combination of inflow velocity boundary conditions, no outflow boundary conditions, and limited void regions will result in a failure of the projection algorithm. This occurs because the solution of the projection equations does not exist and the projection technique will fail to converge. This causes TRUCHAS to terminate with an error message. Limited diagnostic information and graphical and restart data may not be produced in parallel runs. We will continue to research this issue.

Solidifying fluids can lead to rapidly shrinking time steps for flows in confined spaces. This happens when solidification occurs in cells adjacent to cells completely filled with fluid. As the liquid fraction decreases in the solidifying cells, the fluid velocity must become large to balance the flow in adjacent fluid cells. This problem is ameliorated if a porous medium drag model is enabled for the calculation, because the drag force

increases as the liquid volume decreases, keeping it from unbounded growth.

Only one void material (with zero density) can be defined with the current void collapse model. Void cells can conduct heat (assuming the conductivity of the void material is non-zero) but have no heat capacity. This leads to loss of enthalpy. We will correct this in the next code release.

The calculated fluid distribution can be sensitive to the input values for material priorities. In some cases the wrong set of priorities can lead to results that are physically unrealistic. Sometimes it is also difficult to decide what set of priorities will lead to a "good" solution. We will continue to research this issue.

There are some circumstances that generate a large series of WARNING messages concerning the motion of fluid between cells. These messages typically indicate that the flow algorithm is breaking down and the results should not be trusted. A few individual messages are generally not troublesome, particularly if the offending volume fraction is small. We will continue to research this issue.

16.4 Phase Change Algorithm

The phase diagram employed to model phase changes is very simple; a constant partition coefficient is assumed, leading to phase boundaries that are lines emanating from one point on the pure composition axis. We plan to address this issue in the next release of TRUCHAS.

The effect of density changes upon solidification is currently ignored. We treat any string of materials connected by a phase change as having the same density. In addition, the densities of all materials that are connected by a sequence of phase changes must be set equal in the input file. We plan to address this issue in the next release of TRUCHAS.

16.5 Restrictions Due to SI Units

All simulations that employ chemical reactions must be performed in SI units, unless the user inputs an appropriate consistent value of the gas constant R. The same is true for calculations involving radiative heat transfer, unless the user inputs an appropriate value for the Stefan-Boltzmann constant. Finally, the EM calculation assumes SI units. To use a different system of units, the user must specify appropriate values for the parameters `epsilon_0` and `mu_0`. All other TRUCHAS physics is insensitive to the user's chosen system of units, as long as all input values are self-consistent.

16.6 Solvers

Nonlinear solvers can be sensitive to the size of the time step in many cases. In the case of "NK failure" a good first step is to limit the time step size below that at the time of the failure. This often allows TRUCHAS to compute through the problem section. The alternative AIN nonlinear solver may also circumvent this problem.

16.7 Solid Mechanics Algorithm

The solid mechanics formulation is based on small strains and displacements. Accuracy will suffer if strains or rotations are too large. In general it is probably best to limit strains and rotations to less than 0.01. The acceptable limit depends on the application and the user's requirements.

Tetrahedral mesh cells are not recommended for solid mechanics calculations because of degraded accuracy. Tetrahedral cells have not been tested with gap elements.

Slide lines using gap elements are limited to small total displacements. For the purpose of heat transfer we assume complete alignment between cell faces as the gap element deforms. This will result in inaccurate results as the slide displacement approaches the cell dimensions. In addition, the results will not converge as the mesh resolution increases. We will continue to research this issue.

One should not employ preconditioning in problems that involve displacement constraints (`'normal_constraint'` or `'normal_displacement'`). Preconditioning is not effective and may prevent convergence. Correcting this is a high priority, because computation times can be very long without preconditioning.

When creating a mesh that will be preprocessed to add gap elements, it is important that side sets in CUBIT that will be used for gap elements have element faces from no more than two bodies. See Chapter 9 for more information about using gap elements.

Solid mechanics graphics output is currently limited to GMV format. In addition some of the solid mechanics short edit results and timing information in the '*.out' files may be incorrect.

16.8 Electromagnetics Algorithm

In this release the Joule heat computation has been parallelized. However, there is an unresolved problem in the parallel form of the preconditioner that usually results in the failure of the CG linear solve. Thus it

is still usually necessary to compute the Joule heat in serial. In a parallel run this can be done by setting `parallel` to false in the `ALTMESH` namelist, which forces the Joule heat part of the simulation (only) to be done in serial. Note that all but one processor will be idle during the Joule heat computation which may unacceptably wasteful, so it is often preferred to perform an initial Joule-heat-only run in serial and then restart in parallel with heat transfer and other physics enabled.

16.9 Operational Issues

Primary and alternate mesh files must be located in the local directory where TRUCHAS is executed or their location must be specified in full. We plan to eliminate this restriction in the next code release.

Care must be taken in specifying boundary conditions for restart simulations. This is particularly true for problems that specify boundary geometry as material interfaces. It is almost always necessary to change the boundary condition specification for a restart to continue in the same fashion as the original simulation. In many cases the boundary condition specification from the original simulation will fail to compute in restarts because of changes in the fluid configuration.

Additional considerations for TRUCHAS restarts are discussed in Section 15.2.

Appendix A

Input File: Heat Conduction

```
Heat Conduction
This problem demonstrates the heat conduction capabilities
of TRUCHAS by solving the heat transfer across a square
domain with an internally generated structured mesh.

&MESH

  ncell                       =10, 10, 1
  coord                       =3*-0.5, 3*0.5

/

&PHYSICS

  fluid_flow                  =.false.
  heat_conduction             =.true.
  phase_change                =.false.,

/

&MATERIAL

  material_number             =1
  material_name               ='solid_Cu'
  material_feature            ='background'
  priority                    =1
  density                     =8960.
  cp_constants                =385.
  conductivity_constants      =400.

/

&BODY

  material_number             =1
  surface_name                ='background'
  temperature                 =0.

/

&BC

  surface_name                ='conic'
  conic_relation              ='='
  conic_x                     =1.
```

```
  conic_constant              =0.5
  conic_tolerance             =1.e-6
  bc_variable                 ='temperature'
  bc_type                     ='dirichlet'
  bc_value                    =100.

/

&BC

  surface_name                ='conic'
  conic_relation              ='='
  conic_x                     =1.
  conic_constant              =-0.5
  conic_tolerance             =1.e-6
  bc_variable                 ='temperature'
  bc_type                     ='dirichlet'
  bc_value                    =0.

/

&BC

  surface_name                ='external material boundary'
  surface_materials           =1
  bc_variable                 ='temperature'
  bc_type                     ='hneumann'

/


&OUTPUTS

  output_t                    =0., 3.e3
  output_dt                   =300.

/

&NUMERICS

  dt_constant                 =100.
  energy_nonlinear_solution   ='nk'

/

&NONLINEAR_SOLVER

  name                        ='nk'
```

```
  method                      =’nk’
  linear_solver_name          =’ssor-gmres’
  convergence_criterion       =1.e-8
  use_damper                  =.true.
  perturbation_parameter      =1.e-6
  damper_parameters           =0.5, 2., 1., 1.

/

&LINEAR_SOLVER

  name                        =’ssor-gmres’
  method                      =’gmres’
  preconditioning_method      =’ssor’
  preconditioning_steps       =2
  relaxation_parameter        =1.4
  convergence_criterion       =1.e-5

/
```

Appendix B

Input File:Isothermal Phase Change

```
Generated by InputConvert.pl from hc_hs_ipc.inp on Wed Nov 26 14:49:23 MST 2003
-----------------
Heat Transfer + Isothermal Phase Change + Heat Source
This problem reads in an unstructured mesh from a file
and uses an internal heat source to melt a small portion
of the domain.  The phase change is isothermal.

&MESH

  Mesh_File                    =’hc_hs_ipc_mesh.txt’
  Mesh_File_Format             =’genesis’

/

&OUTPUTS

  Output_T                     =0.0, 40.0
  Output_Dt                    =10.0
  Short_Output_Dt_Multiplier   =1

/

&PHYSICS

  fluid_flow                   =.false.
  heat_conduction              =.true.

  phase_change                 =.true.,

/

&HEAT_SOURCE

  heat_source_constant         =1.0e+09
  heat_source_center           =-0.5, 0.0, 1.0
  heat_source_radius           =0.25, 0.25, 2.0
  heat_source_time             =0.0, 15.0
  travel_speed                 =0.0, 0.0, 0.0

/

&NONLINEAR_SOLVER

  name                         =’ht_nl_solver’
  method                       =’nk’
```

```
  linear_solver_name           ='gmres-ssor'
  convergence_criterion        =1.0e-08
  maximum_iterations           =100
  use_damper                   =.false.
  perturbation_parameter       =1.0e-06
  Damper_Parameters            =0.8, 1.5, 1.0, 1.0

/

&LINEAR_SOLVER

  name                         ='gmres-ssor'
  method                       ='gmres'
  preconditioning_method       ='ssor'
  preconditioning_steps        =3
  relaxation_parameter         =0.9
  maximum_iterations           =800
  convergence_criterion        =1.0e-05

/

&NUMERICS

  sfront_number                =1.0
  fourier_number               =0.85
  dt_constant                  =0.40
  cycle_max                    =200
  energy_nonlinear_solution    ='ht_nl_solver'

/

&BC

  BC_Name                      ='bottom'
  Surface_Name                 ='conic'
  Conic_Relation               ='='
  Conic_Z                      =1.0
  Conic_Constant               =1.0
  Conic_Tolerance              =1.0e-5
  BC_Variable                  ='temperature'
  BC_Type                      ='htc'
  BC_Value                     =12e04
  T0_Convection                =10.0

/

&BC
```

```
  BC_Name                   =’top’
  Surface_Name              =’conic’
  Conic_Relation            =’=’
  Conic_Z                   =1.0
  Conic_Constant            =-1.0
  Conic_Tolerance           =1.0e-5
  BC_Variable               =’temperature’
  BC_Type                   =’htc’
  BC_Value                  =12e04
  T0_Convection             =10.0

/

&BC

  BC_Name                   =’back’
  Surface_Name              =’conic’
  Conic_Relation            =’=’
  Conic_X                   =1.0
  Conic_Constant            =0.5
  Conic_Tolerance           =1.0e-5
  BC_Variable               =’temperature’
  BC_Type                   =’hneumann’

/

&BC

  BC_Name                   =’front’
  Surface_Name              =’conic’
  Conic_Relation            =’=’
  Conic_X                   =1.0
  Conic_Constant            =-0.5
  Conic_Tolerance           =1.0e-5
  BC_Variable               =’temperature’
  BC_Type                   =’htc’
  BC_Value                  =12e04
  T0_Convection             =10.0

/

&BC

  BC_Name                   =’left’
  Surface_Name              =’conic’
  Conic_Relation            =’=’
  Conic_Y                   =1.0
  Conic_Constant            =1.0
```

```
  Conic_Tolerance             =1.0e-5
  BC_Variable                 ='temperature'
  BC_Type                     ='htc'
  BC_Value                    =12e04
  T0_Convection               =10.0

/

&BC

  BC_Name                     ='right'
  Surface_Name                ='conic'
  Conic_Relation              ='='
  Conic_Y                     =1.0
  Conic_Constant              =-1.0
  Conic_Tolerance             =1.0e-5
  BC_Variable                 ='temperature'
  BC_Type                     ='htc'
  BC_Value                    =12e04
  T0_Convection               =10.0

/

&BODY

  material_number             =2
  surface_name                ='background'
  temperature                 =100.

/

&MATERIAL

  Material_Number             =1
  Material_Name               ='liquid'
  Density                     =7850.0
  cp_constants                =418.7
  conductivity_constants      =2570.0

/

&MATERIAL

  Material_Number             =2
  Material_Name               ='Solid'
  Material_Feature            ='background'
  Density                     =7850.0
  cp_constants                =418.7
```

```
  conductivity_constants      =3070.0

/

&PHASE_CHANGE_PROPERTIES

  phase_change_active         =.TRUE.
  phase_change_type           ='isothermal'
  phase_change_model          ='none'

  Hi_Temp_Phase_Material_Id   =1
  Lo_Temp_Phase_Material_Id   =2
  Latent_Heat                 =10000.0
  Melting_Temperature         =1510

/

&SENS_VARIABLE

  sens_variable_Name          ='top t0_convection'
  sens_variable_type          ='BC'
  sens_variable_identifier    ='top'
  sens_variable_component     ='t0_convection'
  sens_variable_pert          =0.1

/

&SENS_VARIABLE

  sens_variable_Name          ='top bc_value 1'
  sens_variable_type          ='BC'
  sens_variable_identifier    ='top'
  sens_variable_component     ='bc_value'
  sens_variable_pert          =120.

/

&SENS_FUNCTION

  sens_function_Name          ='T: 2 cells diagonal from BCT'
  sens_function_type          ='T'
  sens_function_rparameters   =-0.45, 0.1788230, .75

/

&SENS_FUNCTION

  sens_function_Name          ='H: 2 cells diagonal from BCT'
```

```
  sens_function_type           ='H'
  sens_function_rparameters    =-0.45, 0.1788230, .75

/
```

Appendix C

Input File: Binary Alloy Phase Change

```
Phase change in an Eutectic Alloy

&MESH

  Ncell                       =24, 16, 2
  Coord                       =0, 0, 0, 1.5, 1.0, 1.0

/

&PHYSICS

  fluid_flow                  =.false.
  heat_conduction             =.true.
  phase_change                =.true.,

/

&MATERIAL

  Material_Name               ='liquid'
  Material_Number             =1
  Conductivity_Constants      =15
  Density                     =5000
  Cp_Constants                =1000
  Priority                    =1

/



&MATERIAL

  Material_Name               ='solid'
  Material_Number             =2
  Conductivity_Constants      =15
  Density                     =5000
  Cp_Constants                =1000
  Priority                    =2

/



&MATERIAL
```

```
  Material_Name               =’mold1’
  Material_Number             =3
  Conductivity_Constants      =20
  Density                     =10000
  Cp_Constants                =2000
  Priority                    =3

/

&MATERIAL

  Material_Name               =’mold2’
  Material_Number             =4
  Material_Feature            =’background’
  Conductivity_Constants      =25
  Density                     =5000
  Cp_Constants                =1000
  Priority                    =4

/

&BODY

  Surface_Name                =’box’
  Length                      =0.5, 1, 1
  Fill                        =’inside’
  Translation_Pt              =0.25, 0.5, 0.5
  Material_Number             =1
  Concentration               =0.1
  Temperature                 =1000

/

&BODY

  Surface_Name                =’box’
  Length                      =0.5, 1, 1
  Fill                        =’inside’
  Translation_Pt              =0.75, 0.5, 0.5
  Material_Number             =3
  Temperature                 =1000

/

&BODY

  Surface_Name                =’background’
  Fill                        =’nodefault’
```

```
  Material_Number             =4
  Temperature                 =1000

/

&BC

  Surface_Name                =’conic’
  Conic_X                     =1
  Conic_Constant              =0
  Conic_Tolerance             =1e-06
  Conic_Relation              =’=’
  BC_Type                     =’dirichlet’
  BC_Variable                 =’temperature’
  BC_Value                    =100

/

&BC

  Surface_Name                =’conic’
  Conic_X                     =1
  Conic_Constant              =-1.5
  Conic_Tolerance             =1e-06
  Conic_Relation              =’=’
  BC_Type                     =’dirichlet’
  BC_Variable                 =’temperature’
  BC_Value                    =100

/

&BC

  Surface_Name                =’conic’
  Conic_Y                     =1
  Conic_Constant              =0
  Conic_Tolerance             =1e-06
  Conic_Relation              =’=’
  BC_Type                     =’hneumann’
  BC_Variable                 =’temperature’

/

&BC

  Surface_Name                =’conic’
  Conic_Y                     =1
  Conic_Constant              =-1
```

```
  Conic_Tolerance             =1e-06
  Conic_Relation              ='='
  BC_Type                     ='hneumann'
  BC_Variable                 ='temperature'

/

&BC

  Surface_Name                ='conic'
  Conic_Z                     =1
  Conic_Constant              =0
  Conic_Tolerance             =1e-06
  Conic_Relation              ='='
  BC_Type                     ='hneumann'
  BC_Variable                 ='temperature'

/

&BC

  Surface_Name                ='conic'
  Conic_Z                     =1
  Conic_Constant              =-1
  Conic_Tolerance             =1e-06
  Conic_Relation              ='='
  BC_Type                     ='hneumann'
  BC_Variable                 ='temperature'

/

&OUTPUTS

  Output_T                    =0, 5000
  Output_Dt                   =500
  Short_Output_Dt_Multiplier  =1
  Long_Output_Dt_Multiplier   =1
  Long_Edit_Bounding_Coords   =0, 1.5, 0, 1, 0, 1

/

&NUMERICS

  dt_constant                 =10
  energy_nonlinear_solution   ='nk'

/
```

```
&NONLINEAR_SOLVER

  name                        =’nk’
  method                      =’nk’
  linear_solver_name          =’nk-linear’
  convergence_criterion       =1.0e-01
  maximum_iterations          =50
  output_mode                 =’none’
  use_damper                  =.true.
  Damper_Parameters           =0.9, 1.1, 1.0, 1.0
  perturbation_parameter      =1.0e-06

/

&LINEAR_SOLVER

  name                        =’nk-linear’
  method                      =’fgmres’
  convergence_criterion       =1.0e-6
  preconditioning_method      =’ssor’
  relaxation_parameter        =0.9
  preconditioning_steps       =3

/

&PHASE_CHANGE_PROPERTIES

  phase_change_active         =.TRUE.
  phase_change_type           =’alloy’
  phase_change_model          =’scheil’
  Hi_Temp_Phase_Material_Id   =1
  Lo_Temp_Phase_Material_Id   =2
  Latent_Heat                 =1000.0
  Liquidus_Slope              =-1000.0
  Liquidus_temp               =900.0
  Melting_Temperature_Solvent =1000.0
  Partition_Coefficient_Constant=0.50
  Eutectic_temperature        =0.0

/
```

Appendix D

Input File:Chemistry

```
Chemical reaction of first order in a box of dimensions
1 x 0.5 x 0.5 split into fifty cells along the x-axis

&CHEMICAL_REACTION

  cr_m                          =1,
  cr_n                          =1,
  cr_reactants_id               =2,
  cr_products_id                =1,
  cr_Cmax                       =0.99,
  cr_ko1                        =1.0e5,
  cr_ko2                        =0.0
  cr_Ea1                        =7.0e4,
  cr_Ea2                        =0.0,
  cr_Htot                       =2000

/

&MESH

  Ncell                         =50, 1, 1
  Coord                         =0, 0, 0,
                                1, 0.5, 0.5

/

&PHYSICS

  fluid_flow                    =.false.,
  heat_conduction               =.true.,

  phase_change                  =.false.,

/

&NUMERICS

  dt_constant                   =0.1
  energy_nonlinear_solution     ='nk',

/

&OUTPUTS

  Output_T                      =0., 100
```

```
  Output_Dt                    =20
  Short_Output_Dt_Multiplier   =1,

/

&MATERIAL

  Material_Name                =’Product’,
  Material_Number              =1,
  Conductivity_Constants       =0.31,
  Density                      =30.,
  Cp_Constants                 =100.,
  Priority                     =1,

/

&MATERIAL

  Material_Name                =’Reactant’,
  Material_Number              =2,
  Material_Feature             =’background’,
  Conductivity_Constants       =0.31,
  Density                      =30.,
  Cp_Constants                 =100.,
  Priority                     =2,

/

&MATERIAL

  Material_Name                =’Mold’,
  Material_Number              =3,
  Conductivity_Constants       =0.1,
  Density                      =30.,
  Cp_Constants                 =10.,
  Priority                     =3,

/

&BODY

  Material_Number              =3,
  Surface_Name                 =’box’,
  Fill                         =’inside’,
  Temperature                  =400.,
  Translation_Pt               =0.25, 0.50, 0.50,
  Length                       =0.50, 1.0, 1.0
```

```
/

&BODY

  Surface_Name                =’background’,
  Fill                        =’nodefault’,
  Material_Number             =2,
  Temperature                 =550.

/

&BC

  Surface_Name                =’conic’,
  Conic_X                     =1,
  Conic_Constant              =0,
  Conic_Tolerance             =1e-06,
  Conic_Relation              =’=’,
  BC_Type                     =’dirichlet’,
  BC_Variable                 =’temperature’,
  BC_Value                    =100.,

/

&BC

  Surface_Name                =’external material boundary’,
  Surface_Materials           =2,
  BC_Type                     =’hneumann’,
  BC_Variable                 =’temperature’,

/

&BC

  Surface_Name                =’external material boundary’,
  Surface_Materials           =3,
  BC_Type                     =’hneumann’,
  BC_Variable                 =’temperature’,

/

&LINEAR_SOLVER

  name                        =’ssor-gmres’,
  method                      =’gmres’,
  convergence_criterion       =1.e-4
  preconditioning_method      =’ssor’,
```

```
  relaxation_parameter         =0.9,
  preconditioning_steps        =3,
  stopping_criterion           ='||r||/||r0||'

/

&NONLINEAR_SOLVER

  name                         ='nk',
  method                       ='nk',
  linear_solver_name           ='ssor-gmres',
  convergence_criterion        =1.e-8
  maximum_iterations           =15,
  output_mode                  ='none',
  use_damper                   =.true.,
  Damper_Parameters            =0.8, 1.5, 1.0, 1.0,
  perturbation_parameter       =1e-06,

/
```

Appendix E

Input File: Flow and Filling

```
Filling of a cylindrical mold.

This problem demonstrates the flow capabilities of truchas by
showing the filling of a cylindrical volume.  The domain is read
in from a mesh file in 'genesis' format and is periodic along the
z-direction.  The y-direction is the vertical direction with
positive y being the 'up' direction.


&MESH

  mesh_file                   ='flow_mesh.txt'
  mesh_file_format            ='genesis'
  coordinate_scale_factor     =100.
  partitions_total            =4

/

&PHYSICS

  fluid_flow                  =.true.
  body_force                  =0.0, -981.0, 0.0
  inviscid                    =.true.
  phase_change                =.false.,

/

&MATERIAL

  material_number             =1
  material_name               ='water'
  priority                    =1
  density                     =1.

/

&MATERIAL

  material_number             =2
  material_name               ='void'
  priority                    =2
  density                     =0.
  material_feature            ='background'

/
```

```
&MATERIAL

  Material_Name               =’mold’
  Material_Number             =3
  Priority                    =3
  Density                     =1000
  Immobile                    =.true.

/

&BODY

  surface_name                =’from mesh file’
  mesh_material_number        =1
  material_number             =2
  temperature                 =0.

/

&BODY

  surface_name                =’from mesh file’
  mesh_material_number        =2
  material_number             =3
  temperature                 =0.

/

&BC

  Surface_Name                =’conic’
  Conic_Relation              =’=’
  Conic_Y                     =1
  Conic_Constant              =-12.2
  Conic_Tolerance             =1.0e-6
  Bounding_Box                =-6.5, 6.5, 12.18, 12.22, -10, 10

  BC_Variable                 =’velocity’
  BC_Type                     =’dirichlet’
  BC_Value                    =0.0, -1000., 0.0

  Inflow_Material             =1
  Inflow_Temperature          =0

/

&OUTPUTS
```

```
  output_t                    =0., 0.04
  output_dt                   =.005
  short_output_dt_multiplier  =1
  long_output_dt_multiplier   =2

/

&NUMERICS

  dt_init                     =1.e-4
  dt_grow                     =1.05
  dt_min                      =1.e-7
  dt_max                      =1.
  discrete_ops_type           ='ortho'
  projection_linear_solution  ='projection-solver'
  courant_number              =0.4
  volume_track_interfaces     =.true.
  volume_track_brents_method  =.true.
  volume_track_iter_tol       =1.0e-12
  cutvof                      =1.0e-08

/

&LINEAR_SOLVER

  name                        ='projection-solver'
  method                      ='gmres'
  preconditioning_method      ='ssor'
  preconditioning_steps       =2
  relaxation_parameter        =1.4
  convergence_criterion       =1.e-8
  maximum_iterations          =50

/
```

Appendix F

Input File: Multiple Phase Changes + Buoyancy Driven Flow

```
Generated by InputConvert.pl from hc_mipc_flow.inp on Wed Nov 26 14:50:05 MST 2003
-----------------
Multiple Phase Changes in a Pure Material

&MESH

  Ncell                        =12, 8, 1
  Coord                        =0, 0, 0, 1.5, 1, 1

/

&OUTPUTS

  Output_T                     =0, 400
  Output_Dt                    =100
  Short_Output_Dt_Multiplier   =1
  Long_Output_Dt_Multiplier    =1
  Long_Edit_Bounding_Coords    =0, 1.5, 0, 1, 0, 1

/

&PHYSICS

  fluid_flow                   =.true.
  heat_conduction              =.true.
  body_force                   =0.0, -9.81, 0.0
  inviscid                     =.true.

  phase_change                 =.true.,

/

&NUMERICS

  dt_init                      =1.e-3
  dt_grow                      =1.05
  dt_min                       =1.e-7
  dt_max                       =2.e0
  discrete_ops_type            ='ortho'
  energy_nonlinear_solution    ='nk-energy'
  projection_linear_solution   ='projection-solver'
  courant_number               =0.4
  volume_track_interfaces      =.true.
  volume_track_brents_method   =.true.
  volume_track_iter_tol        =1.0e-12
```

```
  cutvof                      =1.0e-08

/

&MATERIAL

  Material_Name               =’liquid’
  Material_Number             =1
  Material_Feature            =’background’
  Priority                    =1
  Density                     =5000
  density_change_relation     =’temperature_polynomial’
  density_change_coefficients_T=-1.e-5
  density_change_exponents_T  =1.
  Conductivity_Constants      =250
  Cp_Constants                =10
  reference_temperature       =1000.

/

&MATERIAL

  Immobile                    =.true.
  Material_Name               =’solid1’
  Material_Number             =2
  Priority                    =2
  Density                     =5000
  Conductivity_Constants      =150
  Cp_Constants                =10

/

&MATERIAL

  Immobile                    =.true.
  Material_Name               =’solid2’
  Material_Number             =3
  Conductivity_Constants      =1000
  Density                     =5000
  Cp_Constants                =10
  Priority                    =3

/

&BODY

  Surface_Name                =’background’
  Fill                        =’nodefault’
```

```
  Material_Number             =1
  Temperature                 =1010

/

&BC

  Surface_Name                =’conic’
  Conic_X                     =1
  Conic_Constant              =0
  Conic_Tolerance             =1e-06
  Conic_Relation              =’=’
  BC_Type                     =’dirichlet’
  BC_Variable                 =’temperature’
  BC_Value                    =1200

/

&BC

  Surface_Name                =’conic’
  Conic_X                     =1
  Conic_Constant              =-1.5
  Conic_Tolerance             =1e-06
  Conic_Relation              =’=’
  BC_Type                     =’dirichlet’
  BC_Variable                 =’temperature’
  BC_Value                    =700

/

&BC

  Surface_Name                =’conic’
  Conic_Y                     =1
  Conic_Constant              =0
  Conic_Tolerance             =1e-06
  Conic_Relation              =’=’
  BC_Type                     =’hneumann’
  BC_Variable                 =’temperature’

/

&BC

  Surface_Name                =’conic’
  Conic_Y                     =1
  Conic_Constant              =-1
```

```
  Conic_Tolerance             =1e-06
  Conic_Relation              ='='
  BC_Type                     ='hneumann'
  BC_Variable                 ='temperature'

/

&BC

  Surface_Name                ='conic'
  Conic_Z                     =1
  Conic_Constant              =0
  Conic_Tolerance             =1e-06
  Conic_Relation              ='='
  BC_Type                     ='hneumann'
  BC_Variable                 ='temperature'

/

&BC

  Surface_Name                ='conic'
  Conic_Z                     =1
  Conic_Constant              =-1
  Conic_Tolerance             =1e-06
  Conic_Relation              ='='
  BC_Type                     ='hneumann'
  BC_Variable                 ='temperature'

/

&PHASE_CHANGE_PROPERTIES

  phase_change_active         =.TRUE.
  phase_change_type           ='isothermal'
  phase_change_model          ='none'

  Hi_Temp_Phase_Material_Id   =1
  Lo_Temp_Phase_Material_Id   =2
  Latent_Heat                 =1000.0
  Melting_temperature         =1000.0

/

&PHASE_CHANGE_PROPERTIES

  phase_change_active         =.TRUE.
  phase_change_type           ='isothermal'
```

```
  phase_change_model           =’none’

  Hi_Temp_Phase_Material_Id    =2
  Lo_Temp_Phase_Material_Id    =3
  Latent_Heat                  =1000.0
  Melting_temperature          =850.0

/

&NONLINEAR_SOLVER

  name                         =’nk-energy’
  method                       =’nk’
  linear_solver_name           =’nk-linear’
  convergence_criterion        =1.0e-03
  maximum_iterations           =300
  output_mode                  =’none’
  use_damper                   =.true.
  Damper_Parameters            =0.8, 1.5, 1.0, 1.0
  perturbation_parameter       =1.0e-06

/

&LINEAR_SOLVER

  name                         =’nk-linear’
  method                       =’fgmres’
  convergence_criterion        =1.0e-6
  preconditioning_method       =’ssor’
  relaxation_parameter         =0.9
  preconditioning_steps        =3
  maximum_iterations           =300

/

&LINEAR_SOLVER

  name                         =’projection-solver’
  method                       =’gmres’
  preconditioning_method       =’ssor’
  preconditioning_steps        =2
  relaxation_parameter         =1.4
  convergence_criterion        =1.e-7
  maximum_iterations           =2500

/
```

Appendix G

Input File: Elastic and ViscoPlastic Deformation

```
Viscoplastic Residual Stress Calculation
This problem demonstrates the viscoplastic capabilities of
TRUCHAS by solving the deformation of an aluminiium ring-graphite
plug assembly.


&MESH

  mesh_file                   =’vp_mesh.txt’,
  mesh_file_format            =’genesis’,

/


&PHYSICS

  fluid_flow                  =.false.,
  heat_conduction             =.true.,
  solid_mechanics             =.true.,
  phase_change                =.false.,

/

&MATERIAL

  Immobile                    =.true.
  material_number             =1,
  priority                    =1,
  material_name               =’graphite’,
  density                     =1.7e+03,
  cp_constants                =1.925e+03,
  conductivity_constants      =1.95e+02,
  Lame1_Constants             =3.4e+9,
  Lame2_Constants             =2.76e+9,
  CTE_Constants               =7.0e-06,
  Stress_Reference_Temperature =6.30e+02,
  Material_Feature            =’background’
  Viscoplastic_Model          =’elastic_only’

/

&MATERIAL

  Immobile                    =.true.
  material_number             =2,
```

```
  priority                    =2,
  material_name               =’5754 aluminum’,
  density                     =2.70e+03,
  cp_constants                =9.00e+02,
  conductivity_constants      =2.40e+02,
  Lame1_Constants             =5.20e+10,
  Lame2_Constants             =2.60e+10,
  CTE_Constants               =2.20e-05,
  Stress_Reference_Temperature =6.30e+02,
  Viscoplastic_Model          =’mts’,
  MTS_k                       =1.38e-23,
  MTS_mu_0                    =28.815e9,
  MTS_sig_a                   =10.0e6,
  MTS_d                       =3.440e9,
  MTS_temp_0                  =215.0,
  MTS_b                       =2.86e-10,
  MTS_edot_0i                 =1.0e7,
  MTS_g_0i                    =3.6,
  MTS_q_i                     =1.5,
  MTS_p_i                     =0.5,
  MTS_sig_i                   =107.2e6

/

&BODY

  material_number             =1,
  mesh_material_number        =1,
  surface_name                =’from mesh file’,
  temperature                 =4.55e+02,

/

&BODY

  material_number             =2,
  mesh_material_number        =2,
  surface_name                =’from mesh file’,
  temperature                 =4.55e+02,

/

define the boundary conditions:
  z                           =-0.00635 Heat transfer and zero z displacement on the
bottom surface

&BC
```

```
  surface_name                =’from mesh file’
  mesh_surface                =6,
  bc_variable                 =’temperature’,
  bc_type                     =’hneumann’

/

&BC

  surface_name                =’from mesh file’
  mesh_surface                =6,
  bc_variable                 =’displacement’,
  bc_type                     =’z-displacement’,
  bc_value                    =0.0e0,

/

  z                           =0.00625 heat transfer and all traction components zero (def
the top surface

&BC

  surface_name                =’from mesh file’
  mesh_surface                =5,
  bc_variable                 =’temperature’,
  bc_type                     =’htc’,
  t0_convection               =2.98e+02,
  bc_value                    =1.0e3,

/
  x                           =0.0 insulated with zero x displacement (symmetry plane)

&BC

  surface_name                =’from mesh file’
  mesh_surface                =3,
  bc_variable                 =’temperature’,
  bc_type                     =’hneumann’,

/

&BC

  surface_name                =’from mesh file’
  mesh_surface                =3,
  bc_variable                 =’displacement’,
  bc_type                     =’x-displacement’,
  bc_value                    =0.0e0,
```

```
/
6
  y                          =0.0 insulated with zero y displacement (symmetry plane)
&BC

  surface_name               =’from mesh file’
  mesh_surface               =2,
  bc_variable                =’temperature’,
  bc_type                    =’hneumann’,

/

&BC

  surface_name               =’from mesh file’
  mesh_surface               =2,
  bc_variable                =’displacement’,
  bc_type                    =’y-displacement’,
  bc_value                   =0.0e0,

/
8
  Outer radius               =0.0508 heat transfer and all tractions zero
&BC

  surface_name               =’from mesh file’
  mesh_surface               =4,
  bc_variable                =’temperature’,
  bc_type                    =’htc’,
  t0_convection              =2.98e+02,
  bc_value                   =1.0e2,

/

&OUTPUTS

  output_t                   =0.0e+00, 3.0e0,
  output_dt                  =1.0e0,
  xml_output_dt_multiplier   =1,

/

&NUMERICS

  dt_max                     =5.0e-1,
  dt_grow                    =1.2,
  dt_init                    =2.0e-2,
```

```
  dt_min                        =1.0e-8,
  strain_limit                  =1.0e-12,
  energy_nonlinear_solution     =’conduction nonlin’,
  displacement_nonlinear_solution=’displacement nonlin’,

/

&NONLINEAR_SOLVER

  name                          =’conduction nonlin’,
  method                        =’nk’,
  linear_solver_name            =’conduction nk’,
  convergence_criterion         =1e-04,
  output_mode                   =’none’,
  use_damper                    =.false.,
  perturbation_parameter        =1e-06,

/

&NONLINEAR_SOLVER

  name                          =’displacement nonlin’,
  method                        =’ain’,
  linear_solver_name            =’displacement ain’,
  convergence_criterion         =1e-8,
  AIN_max_vectors               =30,
  AIN_vector_tolerance          =0.01,
  maximum_iterations            =200

/


&LINEAR_SOLVER

  name                          =’conduction nk’,
  method                        =’fgmres’,
  preconditioning_method        =’ssor’,
  preconditioning_steps         =2,
  stopping_criterion            =’||r||/||r0||’,
  convergence_criterion         =1.0e-4

/

&LINEAR_SOLVER

  name                          =’displacement ain’,
  method                        =’none’,
  preconditioning_steps         =4,
```

```
  relaxation_parameter         =1.0,
  preconditioning_method       ='tm_ssor',

/
```

Appendix H

Input File: Internal Interface Displacements

```
Displacement Boundary Condition Problem
This problem demonstrates the use of gap elements in TRUCHAS to
model internal interfaces.

&MESH

  mesh_file                   ='disp.g',
  mesh_file_format            ='exodusII',
  gap_element_blocks          =4, 5, 6

/

&PHYSICS

  fluid_flow                  =.false.,
  heat_conduction             =.false.,
  solid_mechanics             =.true.,
  phase_change                =.false.,

/

define the mold material properties

&MATERIAL

  material_number             =1,
  priority                    =1,
  material_name               ='graphite',
  density                     =1.7e+03,
  cp_constants                =1.925e+03,
  conductivity_constants      =1.95e+02,
  Lame1_Constants             =3.4e+9,
  Lame2_Constants             =2.76e+9,
  CTE_Constants               =7.0e-06,
  Stress_Reference_Temperature =7.00e+02,
  Material_Feature            ='background'
  Viscoplastic_Model          ='elastic_only',
  Immobile                    =.true.

/

define the ring material properties

&MATERIAL
```

```
  material_number             =2,
  priority                    =2,
  material_name               =’5754 aluminum’,
  density                     =2.70e+03,
  cp_constants                =9.00e+02,
  conductivity_constants      =2.40e+02,
  Lame1_Constants             =5.20e+10,
  Lame2_Constants             =2.60e+10,
  CTE_Constants               =2.20e-05,
  Stress_Reference_Temperature =7.00e+02,
  Viscoplastic_Model          =’elastic_only’,
  Immobile                    =.true.

/

gap element material properties

&MATERIAL

  material_number             =3,
  priority                    =3,
  material_name               =’gap’,
  density                     =0.0,
  cp_constants                =0.0,
  conductivity_constants      =0.0,
  Lame1_Constants             =0.0,
  Lame2_Constants             =0.0,
  CTE_Constants               =0.0,
  Immobile                    =.true.

/

define the plug geometry and initial conditions

&BODY

  material_number             =1,
  mesh_material_number        =1,
  surface_name                =’from mesh file’,
  temperature                 =6.0e+02,

/

define the ring geometry and initial conditions

&BODY

  material_number             =1,
```

```
  mesh_material_number         =2,
  surface_name                 =’from mesh file’,
  temperature                  =6.0e+02,

/

&BODY

  material_number              =2,
  mesh_material_number         =3,
  surface_name                 =’from mesh file’,
  temperature                  =6.0e+02,

/

&BODY

  material_number              =3,
  mesh_material_number         =4,
  surface_name                 =’from mesh file’,
  temperature                  =6.0e+02,

/

&BODY

  material_number              =3,
  mesh_material_number         =5,
  surface_name                 =’from mesh file’,
  temperature                  =6.0e+02,

/

&BODY

  material_number              =3,
  mesh_material_number         =6,
  surface_name                 =’from mesh file’,
  temperature                  =6.0e+02,

/

define the boundary conditions:
Zero z displacement on the bottom surface
1
&BC

  surface_name                 =’conic’,
```

```
  conic_relation              ='=',
  conic_z                     =1.0e+00,
  conic_constant              =0.0,
  conic_tolerance             =1.0e-06,
  bc_variable                 ='displacement',
  bc_type                     ='z-displacement',
  bc_value                    =0.0e0,

/

  x                           =0.0 zero x displacement (symmetry plane)
2
&BC

  surface_name                ='conic',
  conic_relation              ='=',
  conic_x                     =1.0e+00,
  conic_constant              =0.0,
  conic_tolerance             =1.0e-06,
  bc_variable                 ='displacement',
  bc_type                     ='x-displacement',
  bc_value                    =0.0e0,

/

60 degree symmetry plane with zero normal displacement
3
&BC

  surface_name                ='from mesh file',
  mesh_surface                =1,
  bc_variable                 ='displacement',
  bc_type                     ='normal-displacement',
  bc_value                    =0.0e0,

/

Interface between plug and ring with htc
4
&BC

  surface_name                ='from mesh file'
  mesh_surface                =2,
  bc_variable                 ='displacement',
  bc_type                     ='normal-constraint',
  bc_value                    =0,

/
```

```
Interface between plug and outer ring with low htc
5
&BC

  surface_name                =’from mesh file’
  mesh_surface                =3,
  bc_variable                 =’displacement’,
  bc_type                     =’contact’,
  bc_value                    =0,

/

Interface between mold bottom and ring
6
&BC

  surface_name                =’from mesh file’
  mesh_surface                =4,
  bc_variable                 =’displacement’,
  bc_type                     =’contact’,
  bc_value                    =0,

/
7
&BC

  surface_name                =’from mesh file’
  mesh_surface                =5,
  bc_variable                 =’displacement’,
  bc_type                     =’z-traction’,
  bc_value                    =-1.0,

/


&OUTPUTS

  output_t                    =0.0e+00, 1.0e-1,
  output_dt                   =1.0e-1,
  short_output_dt_multiplier  =1,
  xml_output_dt_multiplier    =1,

/

&NUMERICS

  dt_constant                 =1.0e-1,
```

```
  displacement_nonlinear_solution='displacement nonlin',
  contact_penalty              =1.0e6,
  contact_norm_trac            =1.0e4,
  contact_distance             =1.0e-7

/

&NONLINEAR_SOLVER

  name                         ='displacement nonlin',
  method                       ='ain',
  linear_solver_name           ='displacement precon',
  convergence_criterion        =1e-8,
  AIN_max_vectors              =30,
  AIN_vector_tolerance         =0.001,
  maximum_iterations           =500

/

&LINEAR_SOLVER

  name                         ='displacement precon',
  method                       ='none',
  preconditioning_steps        =4,
  relaxation_parameter         =1.0,
  preconditioning_method       ='tm_ssor',

/
```

Appendix I

Input File: Induction Heating

```
Induction Heating

This problem demonstrates the electromagnetic capabilities of TRUCHAS by
simulating the induction heating of a graphite sphere.  The 6cm diameter
sphere is positioned at the center of a 10cm diameter, 3-turn induction
coil.  The EM fields are modeled on a cylindrical domain that encloses
the sphere and some of the surrounding free space, but which is contained
within the coil.  The problem is axisymmetric, and so we solve only in
the positive octant.

&MESH

  Mesh_File                   ='ih-hex.gen'
  Mesh_File_Format            ='ExodusII'

/

&ALTMESH

  Altmesh_File                ='ih-tet.gen'

/

&PHYSICS

  Fluid_Flow                  =.false.
  Heat_Conduction             =.true.
  Electromagnetics            =.true.
  phase_change                =.false.,

/

&MATERIAL

  Material_Number             =1
  Material_Name               ='graphite'
  Material_Feature            ='background'
  Density                     =1750.0
  Cp_Constants                =895.0
  Conductivity_Constants      =7.80
  EM_Conductivity_Constants   =5.6e4
  EM_Permittivity_Constants   =1.0
  EM_Permeability_Constants   =1.0

/
```

```
&BODY

  Surface_Name                =’from mesh file’
  Mesh_Material_Number        =1
  Material_Number             =1
  Temperature                 =300.0

/

&BC

  Surface_Name                =’from mesh file’
  Mesh_Surface                =1
  BC_Variable                 =’temperature’
  BC_Type                     =’hneumann’

/

&BC

  Surface_Name                =’from mesh file’
  Mesh_Surface                =2
  BC_Variable                 =’temperature’
  BC_Type                     =’radiation’
  BC_Value                    =0.9, 300.0

/

&OUTPUTS

  Output_T                    =0.0, 30.0
  Output_Dt                   =5.0
  XML_Output_Dt_Multiplier    =1

/

&NUMERICS

  Dt_Constant                 =0.5
  Energy_Nonlinear_Solution   =’HT NLS’
  HT_Discrete_Ops_Type        =’SO’

/

&NONLINEAR_SOLVER

  Name                        =’HT NLS’
```

```
  Method                        =’ain’
  Linear_Solver_Name            =’AIN PC’
  Convergence_Criterion         =1.0e-06

/

&LINEAR_SOLVER

  Name                          =’AIN PC’
  Method                        =’none’
  Preconditioning_Method        =’ssor’
  Preconditioning_Steps         =4,
  Relaxation_Parameter          =1.40

/

&SO_SOLVER

  Name                          =’SOFLUX’
  Convergence_Criterion         =1.0e-8
  Stopping_Criterion            =’||r||/||r0||’
  Maximum_Iterations            =100

/

&ELECTROMAGNETICS

  EM_Domain_Type                =’quarter_cylinder’
  Source_Frequency              =2000.0
  Steps_Per_Cycle               =20
  Maximum_Source_Cycles         =5
  SS_Stopping_Tolerance         =0.01
  Maximum_CG_Iterations         =100
  CG_Stopping_Tolerance         =1.0e-8
  Num_Etasq                     =1.0e-6

/

&INDUCTION_COIL

  Center                        =3*0.0
  Radius                        =0.05
  Length                        =0.04
  NTurns                        =3
  Current                       =5000.0

/
```

Appendix J

Input File: Radiation

```
SPHERICALLY SYMMETRIC ENCLOSURE RADIATION TEST PROBLEM

Concentric material shells separated by void.  Heat flux is imposed at the
inner surface of the inner shell, and the temperature is imposed at the
outer surface of the outer shell.  The two shells are coupled through
radiative heat transfer in the intervening void.

Symmetry is exploited by solving only in the positive octant.

&MESH

  mesh_file                   ='2shell8.gen'
  mesh_file_format            ='ExodusII'

/

&PHYSICS

  fluid_flow                  =.false.
  heat_conduction             =.true.
  Stefan_Boltzmann            =1.0
  phase_change                =.false.,

/

&MATERIAL

  material_number             =1
  material_name               ='foo'
  density                     =1.0
  Cp_constants                =1.0
  conductivity_constants      =4.0

/

&MATERIAL

  material_number             =2
  material_name               ='bar'
  density                     =1.0
  Cp_constants                =1.0
  conductivity_constants      =1.0

/
```

```
!!  Dummy material to satisfy Truchas need for background material.
&MATERIAL

  material_number             =3
  material_name               =’void’
  material_feature            =’background’
  density                     =0.0
  Cp_constants                =0.0
  conductivity_constants      =0.0
  void_temperature            =0.0

/

!!  Inner shell, mesh file block 1
&BODY

  surface_name                =’from mesh file’
  mesh_material_number        =1
  material_number             =1
  temperature                 =1.0

/

!!  Outer shell
&BODY

  surface_name                =’from mesh file’
  mesh_material_number        =2
  material_number             =2
  temperature                 =1.0

/

!!  Dummy body to satisfy truchas need for background material.
&BODY

  material_number             =3
  surface_name                =’background’
  temperature                 =1.0

/

!!  Imposed flux on inner surface of inner shell.
&BC

  surface_name                =’from mesh file’
  mesh_surface                =1
  BC_variable                 =’temperature’
```

```
  BC_type                       =’neumann’
  BC_value                      =16.0

/

!!  Radiation from outer surface of inner shell.
&BC

  surface_name                  =’from mesh file’
  mesh_surface                  =2
  BC_variable                   =’temperature’
  BC_type                       =’enclosure_radiation’
  BC_value                      =1, 0.8

/

!!  Radiation from inner surface of outer shell.
&BC

  surface_name                  =’from mesh file’
  mesh_surface                  =3
  BC_variable                   =’temperature’
  BC_type                       =’enclosure_radiation’
  BC_value                      =1, 0.2

/

!!  Imposed temperature on outer surface of outer shell.
&BC

  surface_name                  =’from mesh file’
  mesh_surface                  =4
  BC_variable                   =’temperature’
  BC_type                       =’dirichlet’
  BC_value                      =1.0

/

!!  No-flux conditions on symmetry planes.
&BC

  surface_name                  =’from mesh file’
  mesh_surface                  =5
  BC_variable                   =’temperature’
  BC_type                       =’hneumann’

/
```

```
&RADIATION_ENCLOSURE

  enclosureID                 =1
  partial                     =.false.
  method                      =’file’
  infile                      =’2shell8-le.vf’
  chaparral_symmetry          =’x’, ’y’, ’z’
  linearsolver                =’RAD LS’

/
  method                      =’chaparral’
  outfile                     =’2shell8.vf’

&LINEAR_SOLVER

  name                        =’RAD LS’
  method                      =’cg’
  preconditioning_method      =’none’
  stopping_criterion          =’||r||/||r0||’
  convergence_criterion       =1.0e-3

/
  stopping_criterion          =’||r||’
  convergence_criterion       =1.0e-6

&OUTPUTS

  Output_t                    =0.0, 4.0
  Output_dt                   =0.4
  XML_output_dt_multiplier    =1
  short_output_dt_multiplier  =100
  enclosure_diagnostics       =.true.

/

&NUMERICS

  dt_constant                 =4.0e-3
  discrete_ops_type           =’ortho’
  HT_discrete_ops_type        =’SO’
  energy_nonlinear_solution   =’HT NLS’
  enclosure_flux_convergence  =4.0e-5

/
  ht_discrete_ops_type        =’SO’
  enclosure_max_iterations    =300

&NONLINEAR_SOLVER
```

```
  name                        ='HT NLS'
  method                      ='ain'
  linear_solver_name          ='HT AIN PC'
  convergence_criterion       =1.0e-06
  AIN_max_vectors             =10
  AIN_vector_tolerance        =0.01
  maximum_iterations          =100

/

&LINEAR_SOLVER

  name                        ='HT AIN PC'
  method                      ='none'
  preconditioning_method      ='ssor'
  preconditioning_steps       =4
  relaxation_parameter        =1.40

/

&PROBE

  probe_name                  ='r0'
  probe_coords                =0.0, 0.0, 0.5

/

&PROBE

  probe_name                  ='r1'
  probe_coords                =0.0, 0.0, 1.0

/

&PROBE

  probe_name                  ='r2'
  probe_coords                =0.0, 0.0, 2.0

/

&PROBE

  probe_name                  ='r3'
  probe_coords                =0.0, 0.0, 4.0

/
```

Appendix K

Input File: Diffusion Solver

```
Nonlinear Diffusion Test Problem

This problem illustrates the use of the diffusion solver component of Truchas
by simulating the solid-state diffusion of a solute into a substrate material.
The quasi-2D domain is [0,3]x[-2,0] (one cell thick in the z-direction).
No-flux conditions are applied on all boundaries except for part of the top
boundary, y=0 and 0<x<1, where the solute concentration is fixed at 1.  The
solute concentration is 0 inside the domain initially.  The solute diffusivity
is a nonlinear function of its concentration.

&OUTPUTS
  Output_T  = 0.0, 1.4
  Output_Dt = 0.05
  XML_Output_Dt_Multiplier = 1
/

&MESH
  Mesh_File = 'umds1.gen'
  Mesh_File_Format = 'ExodusII'
/

&PHYSICS
  Diffusion_Solver = .true.
  Fluid_Flow       = .false.
/

&DIFFUSION_SOLVER
  System_Type  = 'species'
  Abs_Conc_Tol = 1.0e-4
  Rel_Conc_Tol = 1.0e-3
```

```
/

&NUMERICS
  Dt_Init = 1.0d-6
/

&DS_SPECIES_BC
  Name         = 'inflow boundary'
  Face_Set_IDs = 5
  BC_Type      = 'dirichlet'
  BC_Value     = 1.0
/

&DS_SPECIES_BC
  Name         = 'noflow boundaries'
  Face_Set_IDs = 1, 4, 11, 12
  BC_Type      = 'hneumann'
/

&DS_SPECIES_BC
  Name         = 'artificial boundary'
  Face_Set_IDs = 2, 3
  BC_Type      = 'dirichlet'
  BC_Value     = 0.0
/

&BODY
  Surface_Name         = 'from mesh file'
  Mesh_Material_Number = 10
  Material_Number      = 1
  Temperature          = 0.0
  Concentration        = 0.0
/

&MATERIAL
  Material_Number  = 1
  Material_Name    = 'substrate'
  Material_Feature = 'background'
  Density          = 1.0,
  SPC_Diffusivity_Relation  = 'concentration polynomial'
  SPC_Diffusivity_Constants = 0.02, 1.0
  SPC_Diffusivity_Exponents = 0, 1
/
```

Appendix L

Input File: Parallel Heat Conduction

```
Heat Conduction

This problem demonstrates the heat conduction capabilities
of TRUCHAS by solving the heat transfer across a square
domain with an internally generated structured mesh.

This input file works in parallel.

&PARALLEL_PARAMETERS

  partitioner                 =’chaco’

/

&xPARALLEL_PARAMETERS

  partitioner                 =’cartesian’
  processor_array             =2, 2, 1

/

&MESH

  ncell                       =12, 12, 4
  coord                       =-0.5, -0.5, -0.5, 0.5, 0.5, 0.5

/

&PHYSICS

  fluid_flow                  =.false.
  heat_conduction             =.true.
  phase_change                =.false.,

/

&MATERIAL

  material_number             =1
  material_name               =’solid_Cu’
  material_feature            =’background’
  priority                    =1
  density                     =8960.
  cp_constants                =385.
  conductivity_constants      =400.
```

```
/

&BODY

  material_number             =1
  surface_name                =’background’
  temperature                 =0.

/

&BC

  surface_name                =’conic’
  conic_relation              =’=’
  conic_x                     =1.
  conic_constant              =0.5
  conic_tolerance             =1.e-6
  bc_variable                 =’temperature’
  bc_type                     =’dirichlet’
  bc_value                    =100.

/

&BC

  surface_name                =’conic’
  conic_relation              =’=’
  conic_x                     =1.
  conic_constant              =-0.5
  conic_tolerance             =1.e-6
  bc_variable                 =’temperature’
  bc_type                     =’dirichlet’
  bc_value                    =0.

/

&BC

  surface_name                =’external material boundary’
  surface_materials           =1
  bc_variable                 =’temperature’
  bc_type                     =’hneumann’

/


&OUTPUTS
```

```
  output_t                    =0., 3.e3
  output_dt                   =300.
  xml_output_dt_multiplier    =1
  decomposition               =.true.

/

&NUMERICS

  dt_constant                 =100.
  energy_nonlinear_solution   ='nk'

/

&NONLINEAR_SOLVER

  name                        ='nk'
  method                      ='nk'
  linear_solver_name          ='ssor-gmres'
  convergence_criterion       =1.e-8
  use_damper                  =.true.
  perturbation_parameter      =1.e-6
  damper_parameters           =0.5, 2., 1., 1.

/

&LINEAR_SOLVER

  name                        ='ssor-gmres'
  method                      ='gmres'
  preconditioning_method      ='ssor'
  preconditioning_steps       =2
  relaxation_parameter        =1.4
  convergence_criterion       =1.e-5

/
```