truchas

Truchas Reference Manual

Version 21.01.1 — 1/12/2021

Computational Physics and Methods Group (CCS-2)
Computer, Computational, and Statistical Sciences Division
Los Alamos National Laboratory

LA-CC-15-097

Contents

Chapter 1

Introduction

1.1 Invoking Truchas

Truchas is executed in serial using a command of the form

```
truchas [-h] [-d[:n]] [-o:outdir] [-r:rstfile] infile
```

assuming `truchas` is the name of the executable. The brackets denote optional arguments that are described in Table 1.1. The only required argument is the path to the input file *`infile`*. This file name must end with the extension "`.inp`" (without the quotes). The general format of the input file is described in the next section, and the following chapters describe the various Fortran namelists that go into the input file to describe the problem to be simulated.

All of the output files are written to directory whose name is generated from the base name of the input file. For example, if the input file is `myprob.inp`, the output directory will be named `myprob_output`. The name of the output directory can be overridden using the `-o` option. The directory will be created if necessary.

The precise manner of executing Truchas in parallel depends on the MPI implementation being used. This may be as simple as prefixing the serial invocation above with "`mpirun -np n`", where `n` is the number of processes. But this varies widely and providing specific instructions is beyond the scope of this document. There is no difference in the Truchas arguments between serial and parallel, however.

Table 1.1: Truchas command line options

Option	Description
`-h`	Print a usage summary of the command line options and exit.
`-d[:`*`n`*`]`	Sets the debug output level *`n`*. The default level is 0, which produces no debug output, with levels 1 and 2 producing progressively more debug output. `-d` is equivalent to `-d:1`.
`-o:`*`outdir`*	Causes all output files to be written to the directory *`outdir`* instead of the default directory. The directory is created if necessary.
`-r:`*`rstfile`*	Executes in restart mode, restarting from the data in the file *`rstfile`*. This file is generated from the output of a previous Truchas simulation using post-processing utilities.

```
This is a comment.  Anything outside a namelist input is ignored.

&MESH
  ! Within a namelist input "!" introduces a comment.
  mesh_file = "my-big-mesh.exo" ! character string value
/

Another comment.

&PHYSICS
  heat_conduction = .true.     ! logical values are .true./.false.
  body_force = 0.0, 0.0, -9.8  ! assigning values to an array
  !This would be an equivalent method ...
  !body_force(1) =  0.0
  !body_force(2) =  0.0
  !body_force(3) = -9.8
/

Newlines in a namelist are optional.
&PHYSICAL_CONSTANTS stefan_boltzmann=0.1, absolute_zero=0.0 /
```

Figure 1.1: Fragment of a Truchas input file illustrating namelist input syntax.

1.1.1 Stopping Truchas

There are occasions where one would like to gracefully terminate a running Truchas simulation before it has reached the final simulation time given in the input file. This is easily done by sending the running process the `SIGURG` signal:

```
kill -s SIGURG pid
```

where ***pid*** is the process id. When Truchas receives this signal, it continues until it reaches the end of the current time step, where it writes the final solution and then exits normally.

1.2 Input File Format

The Truchas input file is composed of a sequence of Fortran namelist inputs. Each namelist input has the form

&*namelist-group-name*
  namelist-input-body
/

The input begins with a line where the first nonblank is the character & immediately followed by the name of the namelist group. The input continues until the / character. The body of the namelist input consists of a sequence of *name* = *value* pairs, separated by commas or newlines, that assign values to namelist variables. Namelist input is a feature of Fortran and a complete description of the syntax can be found in any Fortran reference, however the basic syntax is very intuitive and a few examples like those in Fig. 1.1 should suffice to explain its essentials.

The namelists and the variables they contain are described in the following chapters. A particular namelist will be required or optional, and it may appear only once or multiple times. Not all variables of a namelist need to be specified. Some may not be relevant in a given context, and others may have acceptable default values; only those that are used and need to be assigned a value need to be specified.

The order of the namelist inputs in the input file is not significant; they may appear in any order. Any text outside of a namelist input is ignored and can be regarded as a comment; see Fig. 1.1.

Fortran is case-insensitive when interpreting the namelist group names and variable names; they may be written in any mixture of upper and lower case. However character string *values*, which are interpreted by Truchas, are case-sensitive unless documented otherwise.

In the event of a namelist input syntax error, Truchas will report that it was unable to read the namelist, but unfortunately it is not able to provide any specific information about the error because Fortran does not make such information available. In such cases the user will need to scan the namelist input for syntax errors: look for misspelt variable names, variables that don't belong to the namelist, blank written in place of an underscore, etc.

1.3 Physical Units

Truchas does not require the use any particular system of physical units, nor does it provide a means for the user to specify the dimension of a numerical value. The user is simply expected to ensure that all input values are given using a consistent system of units. To assist in this end, the dimension for all dimensional quantities is documented using the following abstract units: mass M, length L, time T, thermodynamic temperature Θ, and electric current I. Thus mass density, for example, will be documented as having dimension M/L^3. The following derived abstract units are also used: force F ($= \mathsf{M}\ \mathsf{L}/\mathsf{T}^2$) and energy E ($= \mathsf{M}\ \mathsf{L}^2/\mathsf{T}^2$).

There are a few physical constants, like the Stefan-Boltzmann constant, that have predefined values in SI units. These constants are referenced by a few specific models, and where a model uses one of these constants this fact is noted. Use the `PHYSICAL_CONSTANTS` namelist to redefine the value of these constants where necessary.

1.4 Working With Output Files

As described earlier, Truchas writes its output files to the directory named in the `-o` option, or if omitted, to a directory whose name is generated from the base name of the input file: `myprob_output` if `myprob.inp` is the input file, for example. Two primary files are written, a `.log` file that is a copy of the terminal output, and a `.h5` HDF5 file that contains all the simulation results. HDF5 is a widely-used format for storing and managing data, and there are a great many freely-available tools for working with these files. In this release, which is the first to feature HDF5 output, we provide only a few essential tools, described below, for processing the `.h5` file. We expect to provide additional tools in future releases.

1.4.1 `write_restart`

The program `write_restart` is used to create Truchas restart files using data from an `.h5` output file. The command syntax is

`write_restart [`*`options`*`]` *`H5FILE`*

where *`H5FILE`* is the `.h5` output file and the possible options are

`-h`	Display usage help and exit.
`-l`	Print a list of the available cycles from which the restart file can be created. No restart file is written.
`-n` *`N`*	Data from cycle *`N`* is used to create the restart file; if not specified, the last cycle is used.
`-o` *`FILE`*	Write restart data to *`FILE`*. If not specified, *`FILE`* is taken to be the *`H5FILE`* name with the `.h5` suffix replaced by `.restart.`*`N`* where *`N`* is the cycle number.
`-m` *`FILE`*	Create a mapped restart file using the specified ExodusII mesh *`FILE`* as the target mesh.
`-s` *`FLOAT`*	Scale the mapped restart mesh by the factor *`FLOAT`*.

1.4.2 `write_probes`

The `write_probes` utility extracts probe data (see the `PROBE` namelist) from an `.h5` output file and writes it to the terminal (where it can be redirected as needed) in a multicolumn format suitable for many line plotting programs. The command syntax is

`write_probes { -h | -l | -n `*`N`*` }` *`H5FILE`*

where *`H5FILE`* is the `.h5` output file and the available options are

`-h`	Display usage help and exit.
`-l`	Print a list of the available probes.
`-n` *`N`*	Data for probe index *`N`* is written.

1.4.3 `truchas-gmv-parser.py`

The python script `truchas-gmv-parser.py` is used to create input files for the GMV visualization tool. Formerly distributed gratis, GMV has been commercialized (`http://www.generalmeshviewer.com`). Earlier free versions of the tool can still be found on the internet, however, and it remains available within LANL. The command syntax is

`python truchas-gmv-parser.py [`*`options`*`]` *`H5FILE`*

where *`H5FILE`* is the `.h5` output file. Use the option `-h` to get a full list of the available options.

Chapter 2

ALTMESH Namelist

Overview

The ALTMESH namelist specifies the alternate mesh used by the induction heating solver. This is a 3D tetrahedral mesh imported from an ExodusII format disk file.

ALTMESH Namelist Features

Required/Optional: Required when Electromagnetics is true.

Single/Multiple Instances: Single

Components

- Altmesh_Coordinate_Scale_Factor
- Altmesh_File
- First_Partition
- Grid_Transfer_File
- Partitioner
- Partition_File
- rotation_angles
- data_mapper_kind

Altmesh_Coordinate_Scale_Factor

Description: An optional factor by which to scale all mesh node coordinates.

Type: real

Default: 1.0

Valid values: > 0

`Altmesh_File`

Description: Specifies the path to the ExodusII mesh file. If not an absolute path, it will be interpreted relative the the Truchas input file directory.

Type: case-sensitive string

Default: none

`Grid_Transfer_File`

Description: Certain fields must be mapped between the main and alternative meshes during the course of a simulation. These mappings are accomplished using some fixed grid-mapping data that depend only on the two meshes. This optional variable specifies the path of a file containing this grid mapping data. If specified, and if the file exists, it will be read and its mapping data checked to ensure that it corresponds to the two meshes being used. If it corresponds, the data will be used for the calculation. Otherwise, the grid mapping data is computed and written to the file `altmesh_mapping_data.bin` in the output directory for use in future calculations, avoiding a needless and potentially costly recomputation of the same data.

Type: case-sensitive string

Default: none

Note: The mapping data depends on the internal ordering of the nodes and cells of each mesh, in addition to the meshes themselves. Thus it is recommended that this file be named in such a way that reflects the identity of the two meshes *and* the number of processors used to compute the mapping data; mapping data computed with one number of processors will not be usable in a calculation with a different number of processors, even when the same pair of meshes is used.

`Partitioner`

Description: The partitioning method used to generate the parallel decomposition of the EM mesh.

Type: case-insensitive string

Default: `"chaco"`

Valid values: `"chaco"`, `"metis"`, `"file"`, `"block"`

Notes: See the `MESH` namelist variable `Partitioner` for a description of the options and their associated input variables.

Partition_File

Description: Specifies the path to the EM mesh cell partition file, and is required when Partitioner is "file". If not an absolute path, it will be interpreted as a path relative to the Truchas input file directory.

Type: case-sensitive string

Default: none

Notes: See the Notes for the MESH namelist variable Partition_File.

First_Partition

Description: Specifies the number given the first partition in the numbering convention used in the partition file. Either 0-based or 1-based numbering is allowed.

Type: integer

Default: 0

Valid values: 0 or 1

rotation_angles

Description: A list of 3 angles, given in degrees, specifying the amount of counter-clockwise rotation about the x, y, and z-axes to apply to the mesh. The rotations are done sequentially in that order. A negative angle is a clockwise rotation, and a zero angle naturally implies no rotation.

Type: real 3-vector

Default: $(0.0, 0.0, 0.0)$

data_mapper_kind (Experimental)

Description: This specifies the tool that will be used to map fields between the main heat transfer mesh and this alternative mesh. If "portage" is selected, an experimental data mapper based on the Portage toolkit, https://laristra.github.io/portage, will be used. This data mapper is capable of handling main meshes containing prism and pyramid cells, but it has not yet been thoroughly vetted. Otherwise the normal data mapping tool will be used by default.

Type: string

Default: "default"

Valid values: "default", "portage"

Chapter 3

BC Namelist

Overview

The BC namelist is used to define boundary conditions for the solid mechanics model at external boundaries and internal material interfaces.
The preferred method for specifying the mesh surface where a boundary condition applies is to reference a side set from the ExodusII-format mesh. Alternatively, the mesh surface can specified using a conic surface:

$$0 = p(x, y, z) = c_0 + c_x x + c_y y + c_z z + c_{xx} x^2 + c_{yy} y^2 + c_{zz} z^2 + c_{xy} xy + c_{xz} xz + c_{yz} yz \quad (3.1)$$

A face belongs to the mesh surface whenever its centroid lies on this surface (see Conic_Tolerance). The coefficients are specified using the Conic_* variables. Another method for solid mechanics is to specify nodes. The method is selected using Surface_Name. The specified surface may also be restricted to lie within a bounding box.

BC Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Multiple

Components

- BC_Name
- BC_Table
- BC_Type
- BC_Value
- BC_Variable
- Bounding_Box
- Conic_Constant
- Conic_Tolerance

- Conic_X
- Conic_XX
- Conic_XY
- Conic_XZ
- Conic_Y
- Conic_YY
- Conic_YZ
- Conic_Z
- Conic_ZZ
- Mesh_Surface
- Node_Disp_Coords
- Surface_Name

BC_Name

Description: A name used to identify a particular instance of this namelist.

Type: case-sensitive string

Default: none

Note: This is optional and used for logging purposes only.

BC_Variable

Description: The name of the variable to which this boundary condition applies.

Type: case-insensitive string

Default: none

Valid values: `"displacement"`

BC_Type

Description: The type of boundary condition.

Type: string

Default: Depends on `BC_Variable`:

`'displacement': 'x-traction', 'y-traction', 'z-traction'`

Valid values: Depends on `BC_Variable`:

```
'displacement':  'x-traction', 'y-traction', 'z-traction',
    'x-displacement', 'y-displacement', 'z-displacement',
    'normal-displacement', 'normal-traction', 'free-interface',
    'normal-constraint', 'contact'
```

Notes:
- The solid mechanics displacement solution defaults to a traction-free surface with no displacement constraints (`'x-traction'`, `'y-traction'`, and `'z-traction'` set to zero.)
- The `'free-interface'`, `'normal-constraint'` and 'contact' types can only be specified for interfaces with gap elements.

`BC_Value`

Description: Value(s) for the constant(s) used in this BC definition. See also `BC_Table`.

Physical dimension: varies

Type: `real` (up to 24 values depending on `BC_Type`)

Default: 0.0

Notes: The meaning of the items in the BC_Value list depends on the particular boundary condition:

`BC_Variable`	`BC_Type`	**Value Description**	**Physical Dimension**	**Number of values**
`"displacement"`	`"x-displacement"`	displacement	L	1
`"displacement"`	`"y-displacement"`	displacement	L	1
`"displacement"`	`"z-displacement"`	displacement	L	1
`"displacement"`	`"x-traction"`	traction (force/area)	$\mathsf{F/L^2}$	1
`"displacement"`	`"y-traction"`	traction (force/area)	$\mathsf{F/L^2}$	1
`"displacement"`	`"z-traction"`	traction (force/area)	$\mathsf{F/L^2}$	1
`"displacement"`	`"normal-displacement"`	displacement	L	1
`"displacement"`	`"free-interface"`	not used	—	0
`"displacement"`	`"normal-constraint"`	not used	—	0
`"displacement"`	`"contact"`	not used	—	0

`Bounding_Box`

Description: The extents in each dimension of a bounding box that restricts the extent of the mesh surface where the boundary condition is applied. This does not apply in the case `Surface_Name` is `"node set"`.

Physical dimension: L

Type: A real array $(x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max})$.

Default: Unlimited in each dimension.

`Conic_Constant`

Description: Value of the coefficient c_0 in the conic polynomial (3.1).

Type: real

Default: 0.0

`Conic_Tolerance`

Description: A mesh face is considered to lie on the conic surface when the absolute value of the conic polynomial (3.1) at the face centroid is less than the value of this parameter. Only relevant when using a conic polynomial to define the boundary condition surface.

Type: real

Default: 10^{-6}

Valid values: > 0

Notes: It is important to note that this is not a tolerance on the distance of a centroid from the conic surface, but merely a tolerance on the value of the conic polynomial. Its dimension depends on that of the coefficients in the polynomial.

Most mesh generators place nodes on a bounding surface. For non-planar surfaces, this has the consequence that face centroids will not lie exactly on the surface, making the choice of this tolerance rather significant.

`Conic_X`

Description: Value of the coefficient c_x in the conic polynomial (3.1).

Type: real

Default: 0.0

`Conic_XX`

Description: Value of the coefficient c_{xx} in the conic polynomial (3.1).

Type: real

Default: 0.0

`Conic_XY`

Description: Value of the coefficient c_{xy} in the conic polynomial (3.1).

Type: real

Default: 0.0

`Conic_XZ`

Description: Value of the coefficient c_{xz} in the conic polynomial (3.1).

Type: real

Default: 0.0

`Conic_Y`

Description: Value of the coefficient c_y in the conic polynomial (3.1).

Type: real

Default: 0.0

Conic_YY

Description: Value of the coefficient c_{yy} in the conic polynomial (3.1).

Type: real

Default: 0.0

Conic_YZ

Description: Value of the coefficient c_{yz} in the conic polynomial (3.1).

Type: real

Default: 0.0

Conic_Z

Description: Value of the coefficient c_z in the conic polynomial (3.1).

Type: real

Default: 0.0

Conic_ZZ

Description: Value of the coefficient c_{zz} in the conic polynomial (3.1).

Type: real

Default: 0.0

Mesh_Surface

Description: Identifier of a side set defined in the ExodusII-format mesh. Only relevant when `Surface_Name` is `"from mesh file"`.

Type: integer

Default: none

Node_Disp_Coords

Description: List of points that identify mesh nodes where a displacement boundary condition is applied. Up to 50 points can be specified as a list of (x, y, z) coordinates. Only relevant when `Surface_Name` is `"node set"`.

Physical dimension: `L`

Type: real

Surface_Name

Description: Selects the method of specifying the mesh surface where the boundary condition will be applied.

Type: case-insensitive string

Default: none

Valid values:

Value	Associated variables
"from mesh file"	Mesh_Surface. Requires that the mesh is imported from an ExodusII-format mesh file.
"conic"	Conic_Tolerance, Conic_Constant, Conic_X, Conic_Y, Conic_Z, Conic_XX, Conic_YY, Conic_ZZ, Conic_XY, Conic_XZ, Conic_YZ
"node set"	Node_Disp_Coords

Chapter 4

BODY Namelist

Overview

The BODY namelists define initial material distributions and conditions. The BODY namelists are processed in the order they appear, and identify the specified part of the computational domain not claimed by any preceding BODY namelist. Any "background" type BODY must be listed last.
Each namelist is used to specify a geometry and initial state. The geometry is specified via the variables using an acceptable combination of surface_name, axis, fill, height, length, mesh_material_number, radius, rotation_angle, rotation_pt, and translation_pt, hereafter referred to as geometry-type parameters. The initial state is specified using material_number, velocity, phi, and temperature or temperature_function.

BODY Namelist Features

Required/Optional: Required

Single/Multiple Instances: Multiple

Components

- axis
- fill
- height
- length
- material_name
- mesh_material_number
- phi
- radius
- rotation_angle
- rotation_pt
- surface_name
- temperature
- temperature_function
- translation_pt
- velocity

`axis`

Description: The axis to be used for defining a cylinder or plane.

Type: string

Default: (none)

Valid values: `'x'`, `'y'`, `'z'`

`fill`

Description: The side of the surface to which material is to be inserted for this body.

Type: string

Default: `'inside'`

Valid values: `'inside'`, `'outside'`

`height`

Description: Height of a cylinder body.

Physical Dimension: `L`

Type: `real`

Default: (none)

Valid values: $(0.0, \infty)$

`length`

Description: Length of each side of the box body, or the coefficients of an ellipse or ellipsoid body.

Physical Dimension: `L`

Type: `real` triplet

Default: (none)

Valid values: $(0, \infty)$

`material_name`

Description: Name of the material, or material phase in the case of a multi-phase material, that occupies the volume of this body.

Type: string

Default: (none)

`mesh_material_number`

Description: List of material numbers (element block IDs) associated with the cells as defined in the mesh file. This parameter is only meaningful when surface_name = 'from mesh file'.

Type: `integer` list (16 max)

Default: (none)

Valid values: Existing material numbers in mesh file (if the mesh file is in Exodus/Genesis format, this is the mesh block number).

phi

Description: Initial value of the diffusion solver's multi-component scalar field in the material body.

Physical Dimension: varies

Type: `real` vector of `Num_Species` values

Default: 0.0

Valid values: $(-\infty, \infty)$

radius

Description: Radius of the geometric body (cylinder, sphere, ellipsoid).

Physical Dimension: `L`

Type: `real`

Default: (none)

Valid values: $(0.0, \infty)$

rotation_angle

Description: Angle (degrees) about the (x, y, z) axes this body is to be rotated. This variable is only supported for 'plane' and 'cylinder' body types.

Type: `real` triplet

Default: 0.0, 0.0, 0.0

Valid values: $(-\infty, \infty)$

rotation_pt

Description: Location of the point about which this body is to be rotated. This variable is only supported for 'plane' and 'cylinder' body types.

Physical Dimension: `L`

Type: `real` triplet

Default: 0.0, 0.0, 0.0

Valid values: $(-\infty, \infty)$

surface_name

Description: Type of surface characterizing the interface topology for this body. The available options are:

- **`"background"`** A background body will occupy all space which has not been claimed by previously listed `BODY` namelists. If provided, it must be the final `BODY` namelist provided. When specified, no other geometry-type parameters are relevant.
- **`"plane"`** A plane is specified using `axis`, `rotation_angle`, `rotation_pt`, and `fill` to define the normal direction, and `translation_pt` to provide a point on the plane surface. The normal vector is an 'outward' normal, such that the region defined is in the opposite direction of the normal vector unless `fill` = 'outside'.
- **`"box"`** A box is specified using `translation_pt` as the center, `length` for the length of x, y, and z sides respectively, and `fill` to invert the shape. This shape does not support rotation.

"sphere" A sphere is specified using translation_pt as the center, radius, and fill to invert the shape.

"ellipsoid" An ellipsoid of the form

$$\frac{(x-x_0)^2}{l_1^2} + \frac{(y-y_0)^2}{l_2^2} + \frac{(z-z_0)^2}{l_3^2} \leq 1$$

is specified using translation_pt as the center, length for l_1, l_2, and l_3, , and fill to invert the shape. This shape does not support rotation.

"ellipse" An infinitely long elliptic cylinder of the form

$$\frac{(x-x_0)^2}{l_1^2} + \frac{(y-y_0)^2}{l_2^2} \leq 1$$

is specified using translation_pt as the center, length for l_1 and l_2, , and fill to invert the shape. This shape does not support rotation, and will be aligned with the z axis.

"cylinder" A cylinder is specified using translation_pt as the center of the base, axis, rotation_angle, and rotation_pt to define the orientation, radius, height, and fill to invert the shape.

"from mesh file" This option is used to specify cells associated with element blocks in the input mesh file. mesh_material_number is used to list the desired element blocks. fill may be used to invert the selection.

Type: string

Default: (none)

temperature

Description: Initial constant temperature of the material body.

Physical dimension: Θ

Type: real

Default: none

Note: Either temperature or temperature_function must specified, but not both.

temperature_function

Description: The name of a FUNCTION namelist that defines the initial temperature function for the material body. That function is expected to be a function of (x, y, z).

Type: string

Default: none

Note: Either temperature_function or temperature must specified, but not both.

translation_pt

Description: Location to which each surface origin of this body is translated.

Physical Dimension: L

Type: real triplet

Default: 0.0, 0.0, 0.0

Valid values: $(-\infty, \infty)$

`velocity`

Description: Initial velocity of the material body.

Physical Dimension: `L/T`

Type: `real` triplet

Default: 0.0, 0.0, 0.0

Valid values: $(-\infty, \infty)$

Chapter 5

DIFFUSION_SOLVER Namelist

Overview

The DIFFUSION_SOLVER namelist sets the parameters that are specific to the heat and species transport solver. The namelist is read when either of the PHYSICS namelist options Heat_Transport or Species_Transport are enabled.

The solver has two time integration methods which are selected by the variable Stepping_Method. The default is a variable step-size, implicit second-order BDF2 method that controls the local truncation error of each step to a user-defined tolerance by adaptively adjusting the step size. The step size is chosen so that an a priori estimate of the error will be within tolerance, and steps are rejected when the actual error is too large. A failed step may be retried with successively smaller step sizes.

The other integration method is a non-adaptive, implicit first-order BDF1 method specifically designed to handle the exceptional difficulties that arise when heat transfer is coupled to a fluid flow system that includes void. In this context the heat transfer domain changes from one step to the next because of the moving void region, and mesh cells may only be partially filled with material. For this method the time step is controlled by flow or other physics models.

Both methods share a common nonlinear solver and preconditioning options.

The initial step size and upper and lower bounds for the step size are set in the NUMERICS namelist. In addition, the step size selected by the adaptive solver may be further limited by other physics models or by the NUMERICS variables Dt_Grow and Dt_Constant. When only diffusion solver physics are enabled, it is important that these variables be set appropriately so as not to unnecessarily impede the normal functioning of the diffusion solver.

DIFFUSION_SOLVER Namelist Features

Required/Optional Required when heat transport and/or species transport physics is enabled.
Single/Multiple Instances Single

Components

- Abs_Conc_Tol
- Abs_Enthalpy_Tol
- Abs_Temp_Tol
- Cond_Vfrac_Threshold
- Hypre_AMG_Debug
- Hypre_AMG_Logging_Level
- Hypre_AMG_Print_Level

- Max_NLK_Itr
- Max_NLK_Vec
- Max_Step_Tries
- NLK_Preconditioner
- NLK_Tol
- NLK_Vec_Tol
- PC_AMG_Cycles
- PC_Freq
- PC_SSOR_Relax
- PC_SSOR_Sweeps
- Rel_Conc_Tol
- Rel_Enthalpy_Tol
- Rel_Temp_Tol
- Residual_Atol
- Residual_Rtol
- Stepping_Method
- Verbose_Stepping
- Void_Temperature

Abs_Conc_Tol

Description: The tolerance ϵ for the absolute error component of the concentration error norm used by the BDF2 integrator. If $\delta\mathbf{c}$ is a concentration field increment with reference concentration field $\mathbf{c}$, then this error norm is

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

The relative error tolerance η is given by Rel_Conc_Tol. This variable is only relevant to the adaptive integrator and to diffusion systems that include concentration as a dependent variable.

Physical dimension: same as the 'concentration' variable

Type: `real`

Default: none

Valid values: ≥ 0

Notes: The error norm is dimensionless and normalized. The BDF2 integrator will accept time steps where the estimated truncation error is less than 2, and chooses the next suggested time step so that its prediction of the next truncation error is $\frac{1}{2}$.

For c_j sufficiently small the norm approximates an absolute norm with tolerance ϵ, and for c_j sufficiently large the norm approximates a relative norm with tolerance η. If $\epsilon = 0$ then the norm is a pure relative norm and the concentration must be bounded away from 0.

The same tolerance is used for all concentration components.

Abs_Enthalpy_Tol

Description: The tolerance ϵ for the absolute error component of the enthalpy error norm used by the BDF2 integrator. If $\delta\mathbf{H}$ is a enthalpy field increment with reference enthalpy field $\mathbf{H}$, then this error norm is

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

The relative error tolerance η is given by Rel_Enthalpy_Tol. This variable is only relevant to the adaptive integrator and to diffusion systems that include enthalpy as a dependent variable.

Physical dimension: $\mathsf{E}/(\Theta\,\mathsf{L}^3)$

Type: `real`

Default: none

Valid values: ≥ 0

Notes: The error norm is dimensionless and normalized. The BDF2 integrator will accept time steps where the estimated truncation error is less than 2, and chooses the next suggested time step so that its prediction of the next truncation error is $\frac{1}{2}$.

For H_j sufficiently small the norm approximates an absolute norm with tolerance ϵ, and for H_j sufficiently large the norm approximates a relative norm with tolerance η. If $\epsilon = 0$ then the norm is a pure relative norm and the enthalpy must be bounded away from 0.

Abs_Temp_Tol

Description: The tolerance ϵ for the absolute error component of the temperature error norm used by the BDF2 integrator. If $\delta\mathbf{T}$ is a temperature field increment with reference temperature field $\mathbf{T}$, then this error norm is

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

The relative error tolerance η is given by Rel_Temp_Tol. This variable is only relevant to the adaptive integrator and to diffusion systems that include temperature as a dependent variable.

Physical dimension: Θ

Type: `real`

Default: none

Valid values: ≥ 0

Notes: The error norm is dimensionless and normalized. The BDF2 integrator will accept time steps where the estimated truncation error is less than 2, and chooses the next suggested time step so that its prediction of the next truncation error is $\frac{1}{2}$.

For c_j sufficiently small the norm approximates an absolute norm with tolerance ϵ, and for c_j sufficiently large the norm approximates a relative norm with tolerance η. If $\epsilon = 0$ then the norm is a pure relative norm and the temperature must be bounded away from 0.

Cond_Vfrac_Threshold

Description: Material volume fraction threshold for inclusion in heat conduction when using the non-adaptive integrator.

Type: `real`

Default: 0.001

Valid values: $(0, 1)$

Note: Fluid flow systems that include void will result in partially filled cells, often times with only a tiny fragment of material. Including such cells in the heat conduction problem can cause severe numerical difficulties. By excluding cells with a material volume fraction less than this threshold from participation in heat conduction we can obtain a much better conditioned system. Note that we continue to track enthalpy for such cells, including enthalpy that may be advected into or out of the cell; we just do not consider diffusive transport of enthalpy.

Hypre_AMG_Debug

Description: Enable debugging output from Hypre's BoomerAMG solver. Only relevant when `NLK_Preconditioner` is set to `'Hypre_AMG'`.

Type: `logical`

Default: `.false.` (off)

Note: See `HYPRE_BoomerAMGSetDebugFlag` in the Hypre Reference Manual.

Hypre_AMG_Logging_Level

Description: Enable additional diagnostic computation by Hypre's BoomerAMG solver. Only relevant when `NLK_Preconditioner` is set to `'Hypre_AMG'`.

Type: `integer`

Default: 0 (none)

Valid values: 0, none; > 0, varying amounts. Refer to the Hypre Reference Manual description of `HYPRE_BoomerAMGSetLogging` for details.

Hypre_AMG_Print_Level

Description: The diagnostic output verbosity level of Hypre's BoomerAMG solver. Only relevant when `NLK_Preconditioner` is set to `'Hypre_AMG'`.

Type: `integer`

Default: 0

Valid values:

0	no output
1	write setup information
2	write solve information
3	write both setup and solve information

See `HYPRE_BoomerAMGSetPrintLevel` in the Hypre Reference Manual.

Max_NLK_Itr

Description: The maximum number of NLK nonlinear solver iterations allowed.

Type: `integer`

Default: 5

Valid values: ≥ 2

Notes: This variable is used by both the adaptive and non-adaptive integrators, though the appropriate values differ significantly.

For the adaptive integrator, the failure of a nonlinear iteration to converge *is not* necessarily fatal; the BDF2 integration procedure expects that this will occur, using it as an indication that the preconditioner for the nonlinear system needs to be updated. If still unsuccessful, the step may be retried with a halved time step size, perhaps repeatedly. Therefore it is important that the maximum number of iterations not be set too high, as this merely delays the recognition that some recovery strategy needs to be taken, and can result in much wasted effort.

By contrast, a nonlinear solver convergence failure *is* fatal for the non-adaptive solver. Thus the maximum number of iterations should be set to some suitably large value; if the number of iterations ever exceeds this value the simulation is terminated.

The default value is appropriate for the adaptive integrator.

`Max_NLK_Vec`

Description: The maximum number of acceleration vectors to use in the NLK nonlinear solver.

Type: `integer`

Default: `Max_NLK_Itr` -1

Valid values: > 0

Notes: The acceleration vectors are derived from the difference of successive nonlinear function iterates accumulated over the course of a nonlinear solve. Thus the maximum possible number of acceleration vectors available is one less than the maximum number of NLK iterations, and so specifying a larger number merely wastes memory. If a large number of NLK iterations is allowed (as when using the non-adaptive integrator) then it may be appropriate to use a smaller value for this parameter, otherwise the default value is fine.

`Max_Step_Tries`

Description: The maximum number of attempts to successfully take a time step before giving up. The step size is reduced between each try. This is only relevant to the adaptive solver.

Type: `integer`

Default: 10

Valid values: ≥ 1

Notes: If other physics is enabled then this variable is effectively assigned the value 1, overriding the input value. This is required for compatibility with the other physics solvers which currently have no way of recovering from a failed step.

`NLK_Preconditioner`

Description: The choice of preconditioner for the NLK iteration. There are currently two preconditioners to choose from: SSOR and HYPRE_AMG. The former is symmetric over relaxation, and the later is an algebraic multigrid preconditioner from the *hypre* library.

Type: `string`

Default: `'SSOR'`

Valid values: `'SSOR'` or `'Hypre_AMG'`

Notes: If SSOR is the chosen as the preconditioner, the user can set PC_SSOR_Relax to the over relaxation parameter and PC_SSOR_Sweeps to the number of SSOR sweeps. If Hypre_AMG is the chosen as the preconditioner, the user can set PC_AMG_Cycles to the number of AMG cycles per preconditioning step.

NLK_Tol

Description: The convergence tolerance for the NLK nonlinear solver. The nonlinear system is considered solved by the current iterate if the BDF2 integrator norm of the last solution correction is less than this value. This variable is only relevant to the adaptive integrator.

Type: `real`

Default: 0.1

Valid values: $(0, 1)$

Notes: This tolerance is relative to the dimensionless and normalized BDF2 integrator norm; see Abs_Conc_Tol, for example. The nonlinear system only needs to be solved to an accuracy equal to the acceptable local truncation error for the step, which is roughly 1. Solving to a greater accuracy is wasted effort. Using a tolerance in the range $(0.01, 0.1)$ is generally adequate to ensure a sufficently converged nonlinear iterate.

NLK_Vec_Tol

Description: The NLK vector drop tolerance. When assembling the acceleration subspace vector by vector, a vector is dropped when the sine of the angle between the vector and the subspace less than this value.

Type: `real`

Default: 0.001

Valid values: > 0

PC_AMG_Cycles

Description: The number of V-cycles to take per preconditioning step of the nonlinear iteration.

Physical Dimension: dimensionless

Type: `integer`

Default: 2

Valid values: ≥ 1

Notes: We use standard $V(1, 1)$ cycles. Parameters other than the number of V cycles cannot be controlled by the user.

PC_Freq

Description: This controls how frequently the preconditioner is updated in the adaptive BDF2 integrator. A value of N will allow a preconditioner to be used for as many as N consecutive time steps before being updated, although it may be updated more frequently based on other criteria. A value of 1 causes the preconditioner to be updated every time step. The default behavior is to not require any minimum update frequency.

Type: `integer`

Default: ∞

Valid values: ≥ 1

Notes: A basic strategy of the adaptive BDF2 integrator is to use a preconditioner for as many time steps as possible, and only update it when a nonlinear time step iteration fails to converge. This generally works quite well. But if you find that the integrator is thrashing — evidenced by the number of times a step failed with an old preconditioner and was retried (this is the `NNR` diagnostic value in the terminal output) being a significant fraction of the number of time steps — it may be more cost effective to set this value to 1, for example.

`PC_SSOR_Relax`

Description: The relaxation parameter used in the SSOR preconditioning of the nonlinear system.

Physical Dimension: dimensionless

Type: `real`

Default: 1.4

Valid values: $(0, 2)$

Notes: A value less than 1 gives under-relaxation and a value greater than 1 over-relaxation.

`PC_SSOR_Sweeps`

Description: The number of sweeps used in the SSOR preconditioning of the nonlinear system.

Type: `integer`

Default: 4

Valid values: ≥ 1

Notes: The effectiveness of the SSOR preconditioner (measured by the convergence rate of the nonlinear iteration) improves as the number of sweeps increases, though at increasing cost. For especially large systems where the effectiveness of SSOR deteriorates, a somewhat larger value than the default 4 sweeps may be required. Using fewer than 4 sweeps is generally not recommended.

`Rel_Conc_Tol`

Description: The tolerance η for the relative error component of the concentration error norm used by the BDF2 integrator. If $\delta\mathbf{c}$ is a concentration field increment with reference concentration field $\mathbf{c}$, then this error norm is

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

The absolute error tolerance ϵ is given by `Abs_Conc_Tol`. This variable is only relevant to the adaptive solver and to diffusion systems that include concentration as a dependent variable.

Physical Dimension: dimensionless

Type: `real`

Default: 0.0

Valid values: $[0, 1)$

Notes: See the notes for `Abs_Conc_Tol`.

Rel_Enthalpy_Tol

Description: The tolerance η for the relative error component of the enthalpy error norm used by the BDF2 integrator. If $\delta\mathbf{c}$ is a enthalpy field increment with reference enthalpy field $\mathbf{H}$, then this error norm is

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

The absolute error tolerance ϵ is given by Abs_Enthalpy_Tol. This variable is only relevant to the adaptive solver and to diffusion systems that include enthalpy as a dependent variable.

Physical Dimension: dimensionless

Type: `real`

Default: 0.0

Valid values: $[0, 1)$

Notes: See the notes for Abs_Enthalpy_Tol.

Rel_Temp_Tol

Description: The tolerance η for the relative error component of the temperature error norm used by the BDF2 integrator. If $\delta\mathbf{T}$ is a temperature field increment with reference temperature field $\mathbf{T}$, then this error norm is

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

The absolute error tolerance ϵ is given by Abs_Temp_Tol. This variable is only relevant to the adaptive solver and to diffusion systems that include temperature as a dependent variable.

Physical Dimension: dimensionless

Type: `real`

Default: 0.0

Valid values: $[0, 1)$

Notes: See the notes for Abs_Temp_Tol.

Residual_Atol

Description: The absolute residual tolerance ϵ_1 used by the iterative nonlinear solver of the non-adaptive integrator. If r_0 denotes the initial nonlinear residual, iteration stops when the current residual r satisfies $||r||_2 \le \max\{\epsilon_1, \epsilon_2||r_0||_2\}$.

Type: `real`

Default: 0

Valid values: ≥ 0

Note: Ideally this tolerance should be set to 0, but in some circumstances, especially at the start of a simulation, the initial residual may be so small that it is impossible to reduce it by the factor ϵ_2 due to finite precision arithmetic. In such cases it is necessary to provide this absolute tolerance. It is impossible, however, to say what a suitable value would be, as this depends on the nature of the particular nonlinear system. Some guidance can be obtained through trial-and-error by enabling Verbose_Stepping and observing the magnitude of the residual norms in the resulting diagnostic output file.

Residual_Rtol

Description: The relative residual tolerance ϵ_2 used by the iterative nonlinear solver of the non-adaptive integrator. If r_0 denotes the initial nonlinear residual, iteration stops when the current residual r satisfies $||r||_2 < \max\{\epsilon_1, \epsilon_2||r_0||_2\}$.

Type: `real`

Default: none

Valid values: $(0, 1)$

Stepping_Method

Description: The choice of time integration method.

Type: string

Default: `'Adaptive BDF2'`

Valid values: `'Adaptive BDF2'` or `'Non-adaptive BDF1'`

Note: The non-adaptive integrator must be selected when fluid flow is enabled and void material is present. Otherwise use the default adaptive integrator.

Verbose_Stepping

Description: A flag that enables the output of detailed BDF2 time stepping information. The human-readable information is written to a file with the suffix `.bdf2.out` in the output directory.

Type: `logical`

Default: `.false.`

Void_Temperature

Description: An arbitrary temperature assigned to cells that contain only the void material. The value has no effect on the simulation, and is only significant to visualization.

Type: `real`

Default: 0

Chapter 6

DS_SOURCE Namelist

The DS_SOURCE namelist is used to define external volumetric sources for species transport model.

Overview

DS_SOURCE Namelist Features

Required/Optional: Optional
Single/Multiple Instances: Multiple

Components

- Equation
- Cell_Set_IDs
- Source_Constant
- Source_Function

Equation

Description: The name of the equation this source applies to.

Type: string

Default: none

Valid values: "concentration1", "concentration2", ...

Note: Any name may be specified, but Truchas will only look for and use DS_SOURCE namelists with the indicated equation names; any others are silently ignored.

Cell_Set_IDs

Description: A list of cell set IDs that define the subdomain where the source is applied.

Type: a list of up to 32 integers

Default: none

Valid values: any valid mesh cell set ID

Note: Different instances of this namelist with a given Equation value must apply to disjoint subdomains; overlapping of source functions is not allowed.

Exodus II mesh element blocks are interpreted by Truchas as cell sets having the same IDs.

Source_Constant

Description: The constant value of the source function.

Type: real

Default: none

Note: Either Source_Constant or Source_Function must be specified, but not both.

Source_Function

Description: The name of a FUNCTION namelist that defines the source function. That function is expected to be a function of (t, x, y, z).

Type: string

Default: none

Note: Either Source_Function or Source_Constant or must be specified, but not both.

Chapter 7

ELECTROMAGNETICS Namelist

Overview

The ELECTROMAGNETICS namelist sets most of the parameters used by the electromagnetic (EM) solver to calculate the Joule heat used in induction heating simulations. Exceptions are the electrical conductivity, electric susceptibility, and magnetic susceptibility which are defined for each material phase using the MATERIAL namelist, and the induction coils hat produce an external magnetic source field, which are specified in INDUCTION_COIL namelists. The EM calculations are performed on a tetrahedral mesh specified by the ALTMESH namelist, which is generally different than the main mesh used throughout the rest of Truchas.

A Remark on Units: The EM solver assumes SI units by default. In particular, the result of the Joule heat calculation is a *power density*—W/m^3 in SI units. To use a different system of units, the user must supply appropriate values for the free-space constants Vacuum_Permittivity and Vacuum_Permeability. In any case, the user must ensure that a consistent set of units is used throughout Truchas.

ELECTROMAGNETICS Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Single

Components

- CG_Stopping_Tolerance
- EM_Domain_Type
- Graphics_Output
- Material_Change_Threshold
- Maximum_CG_Iterations
- Maximum_Source_Cycles
- Num_Etasq
- Output_Level
- Source_Frequency
- Source_Times
- SS_Stopping_Tolerance
- Steps_Per_Cycle
- Symmetry_Axis
- Uniform_Source

CG_Stopping_Tolerance

Description: Tolerance used to determine when the conjugate gradient (CG) iteration has converged. The criterion used is that $\|\mathbf{r}\|/\|\mathbf{r}_0\| <$ `CG_Stopping_Tolerance`. The electromagnetics solver uses its own special preconditioned CG linear solver.

Type: `real`

Default: 10^{-5}

Valid values: $(0, 0.1)$

Notes: The numerical characteristics of the electromagnetic system require that the linear systems be solved to significantly greater accuracy than would otherwise be required. Too loose a tolerance will manifest itself in a significant build-up of noise in the solution of the electric field over the course of the simulation. This input variable should not be greater than 10^{-4}.

EM_Domain_Type

Description: A flag specifying the type of domain geometry that is discretized by the computational mesh.

Type: string

Default: none

Valid values: `'Full_Cylinder'`, `'Half_Cylinder'`, `'Quarter_Cylinder'`

Notes: At this time there is not yet a facility for specifying general boundary conditions for the electromagnetic simulation. Consequently, the computational domain Ω is limited to the following special cases when Symmetry_Axis=`'z'`:

`'Full_Cylinder'`:	$\Omega = \{(x, y, z) \mid x^2 + y^2 \le r^2,\ z_1 \le z \le z_2\}$
`'Half_Cylinder'`:	$\Omega = \{(x, y, z) \mid x^2 + y^2 \le r^2,\ x \ge 0,\ z_1 \le z \le z_2\}$
`'Quarter_Cylinder'`:	$\Omega = \{(x, y, z) \mid x^2 + y^2 \le r^2,\ x, y \ge 0,\ z_1 \le z \le z_2\}$

The values $r > 0$, $z_1 < z_2$ are inferred from the mesh and are not specified directly. Dirichlet source field conditions are imposed on the boundaries $\{x^2 + y^2 = r^2\}$ and $\{z = z_1, z_2\}$, and symmetry conditions on the symmetry planes $\{x = 0\}$ and $\{y = 0\}$ if present. See the User Manual for more details.

The analogous definitions for the other possible symmetry axes, `'x'` and `'y'`, are obtained by the appropriate cyclic permutation of the coordinates.

For the computational mesh used in the EM simulation, see the ALTMESH namelist.

Experimental Features. The value `'Frustum'` specifies that the domain is a frustum of a right cone

$$\Omega = \{(x, y, z) \mid x^2 + y^2 \le m^2(z - z_0)^2,\ z_1 \le z \le z_2\}$$

or an angular wedge of a frustum. As with the other domain types the values $m > 0$, z_0 and $z_1 < z_2$ are inferred from the mesh and are not specified directly. For wedges of a frustum, the wedge sides can lie on any plane from the family of 30 degree increment planes that includes the $x = 0$ plane. The preceding description is for the z-axis symmetry case, but the analogous functionality for the other symmetry axes is also provided.

Graphics_Output

Description: Controls the graphics output of the electromagnetic solver.

Type: `logical`

Default: `.false.`

Valid values: `.true.`or `.false.`

Notes: If `.true.`, the electromagnetic solver will generate its own graphics data files in OpenDX format (see `http://www.opendx.org`). The files contain the material parameter fields, the averaged Joule heat field, and the time series of the electromagnetic fields. The files are identified by the suffixes `-EM.dx` and `-EM.bin`. The value of `Graphics_Output` has no impact on the normal graphics output generated by Truchas, which is determined elsewhere, and the averaged Joule heat field used in heat transport will be output there in either case.

Material_Change_Threshold

Description: Controls, at each step, whether the Joule heat is recalculated in response to temperature-induced changes in the EM material parameter values. The Joule heat is recalculated whenever the difference between the current parameter values and those when the Joule heat was last computed exceeds this threshold value. Otherwise the previously calculated Joule heat is used. The maximum relative change is used as the difference measure.

Type: `real`

Default: 0.3

Valid values: $(0.0, \infty)$

Notes: The electric conductivity and magnetic permeability are the only values whose changes are monitored. The electric permittivity only enters the equations through the displacement current term, which is exceedingly small in this quasi-magnetostatic regime and could be dropped entirely. Thus the Joule heat is essentially independent of the permittivity and so any changes in its value are ignored.

For electric conductivity, only the conducting region (where the value is positive) is considered when computing the difference. An underlying assumption is that this region remains fixed throughout the simulation.

Maximum_CG_Iterations

Description: Maximum number of conjugate gradient (CG) iterations allowed. The electromagnetics solver uses its own special preconditioned CG linear solver.

Type: `integer`

Default: 500

Valid values: $(0.0, \infty)$

Maximum_Source_Cycles

Description: The electromagnetic field equations are integrated in time toward a periodic steady state. This input variable specifies the time limit, measured in cycles of the sinusoidal source field, for the Joule heat calculation.

Type: `integer`

Default: 10

Valid values: $(0.0, \infty)$

Notes: To avoid ringing, the amplitude of the external source field is ramped and is not at full strength until after approximately two cycles have passed. Consequently this input variable should not normally be < 3. Convergence to a periodic steady state is usually attained within 5 cycles; see `SS_Stopping_Tolerance`. If convergence is not attained within the limit allowed by this input variable, the last result is returned and a warning issued, but execution continues with the rest of the physics simulation.

The electromagnetic field equations are solved on an *inner* time distinct from that of the rest of the physics; see `SS_Stopping_Tolerance`.

`Num_Etasq`

Description: This value is used for the displacement current coefficient η^2, in the low-frequency, nondimensional scaling of Maxwell's equations, when its value exceeds the physical value.

Physical dimension: dimensionless

Type: `real`

Default: 0

Valid values: $(0.0, \infty)$

Notes: The quasi-magnetostatic regime is characterized by $\eta^2 \ll 1$. Since this value can become exceedingly small (resulting in a difficult-to-solve, ill-conditioned system), it may be helpful to use a numerical value instead, say $\eta^2 = 10^{-6}$ or 10^{-8}, without having any discernable effect on the solution. However, it is generally safe to ignore this variable and let the solver use the physical value. See the *Truchas Physics and Algorithms* for more details.

`Output_Level`

Description: Controls the verbosity of the electromagnetic solver

Type: `integer`

Default: 1

Valid values: 1, 2, 3 or 4

Notes: At the default level, 1, a status message is output at the end of each source field cycle showing the progress toward steady state. Level 2 adds a summary of the CG iteration for each time step. Level 3 adds the norm of the difference between the solution and extrapolated predictor for each time step. This gives an indication of the (time) truncation error, and if noise is accumulating in the system it will be seen here; see `CG_Stopping_Tolerance`. Level 4 adds convergence info for each CG iterate. Levels 1 and 2 are typical.

`Source_Frequency`

Description: Frequency f (cycles per unit time) of the sinusoidally-varying magnetic source fields that drive the Joule heat calculation.

Physical dimension: T^{-1}

Type: `real`

Default: none

Valid values: Any single positive value, or any sequence of positive values.

Notes: A sequence of up to 32 values may be assigned to this variable in order to specify a time-dependent frequency; see `Source_Times` and Fig. 7.1.

The source fields are due to induction coils specified through `INDUCTION_COIL` namelists, and a spatially uniform source field specified by `Uniform_Source`. All operate at the common frequency specified by this variable, and a common phase. The phase value is irrelevant due to the time averaging of the calculated Joule heat.

The Joule heat calculation is coupled to the rest of the physics in a manner that assumes the time scale of the electromagnetic fields, f^{-1} is *much smaller* than the time scale of the other physics (as defined by the characteristic time step). Consequently, the frequency f must not be too small.

Source_Times

Description: A sequence of times that define the time partition of the piecewise-constant functional form used in the case of time-dependent source field parameters.

Physical dimension: T

Type: real

Default: none

Valid values: Any strictly increasing sequence of values.

Notes: If this variable is not specified, then the source field parameters are assumed to be constant in time, with a *single* value assigned to Source_Frequency, Uniform_Source, and each Current variable in any INDUCTION_COIL namelists. Otherwise, if n values are assigned to Source_Times, then $n + 1$ values must be assigned to each of those variables. See Fig. 7.1 for a description of the functional form described by these values.

At most 31 values may be specified for this variable.

SS_Stopping_Tolerance

Description: The electromagnetic field equations are integrated in time toward a periodic steady state. Convergence to this steady state is measured by comparing the computed Joule heat field averaged over the last source field cycle, q_{last}, with the result from the previous cycle, q_{prev}. When $\|q_{\text{last}} - q_{\text{prev}}\|_{\max}/\|q_{\text{last}}\|_{\max} <$ SS_Stopping_Tolerance, the Joule heat calculation is considered converged and q_{last} returned.

Type: real

Default: 10^{-2}

Valid values: $(0.0, \infty)$

Notes: Depending on the accuracy of the other physics, 10^{-2} or 10^{-3} are adequate values. If the value is taken too small (approaching machine epsilon) convergence cannot be attained.

It is assumed that the time scale of the electromagnetic fields is *much shorter* than the time scale of the other physics; see Source_Frequency. In this case it suffices to solve the electromagnetic field equations to a periodic steady state, while temporarily freezing the other physics, and averaging the rapid temporal variation in the Joule heat field over a cycle. In effect, the electromagnetic field equations are solved over an *inner* time distinct from that of the rest of the physics.

Steps_Per_Cycle

Description: The number of time steps per cycle of the external source field used to integrate the electromagnetic field equations

Type: integer

Default: 20

Valid values: $(0.0, \infty)$

Notes: Increasing the number of time steps per cycle increases the accuracy of the Joule heat calculation, while generally increasing the execution time. A reasonable range of values is $[10, 40]$; anything less than 10 is *severely* discouraged.

The electromagnetic field equations are solved on an *inner* time distinct from that of the rest of the physics; see SS_Stopping_Tolerance.

Symmetry_Axis

Description: A flag that specifies which axis is to be used as the problem symmetry axis for the Joule heat simulation.

Type: string

Default: 'z'

Valid values: { 'x', 'y', 'z' }

Notes: The value of this variable determines the orientation of the uniform magnetic source field specified by Uniform_Source, and the coils specified by the INDUCTION_COIL namelists, if any. It also determines the assumed orientation of the computational domain; see EM_Domain_Type.

Uniform_Source

Description: Amplitude of a sinusoidally-varying, uniform magnetic source field that drives the Joule heat computation. The field is directed along the problem symmetry axis as specified by Symmetry_Axis.

Physical dimension: I/L

Type: real

Default: 0

Valid values: Any single value, or any sequence of values.

Notes: A sequence of up to 32 values may be assigned to this variable in order to specify a time-dependent amplitude; see Source_Times and Fig. 7.1. If this variable is not specified, its value or values, as appropriate, are assumed to be zero.

The total external magnetic source field that drives the Joule heat computation will be the superposition of this field and the fields due to the coils specified in the INDUCTION_COIL namelists, if any.

For reference, the magnitude of the magnetic field within an infinitely-long, finely-wound coil with current *density* I is simply I. The field magnitude at the center of a circular current loop of radius r with current I is $I/2r$. In both cases the field is directed along the axis of the coil/loop.

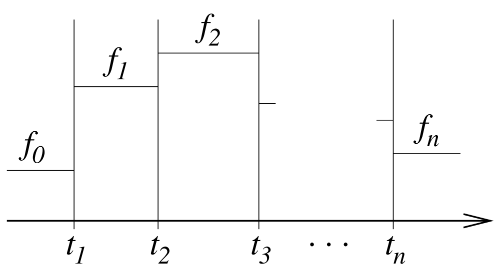


Figure 7.1: Piecewise constant function form showing the constant values $f_0, f_1, \ldots, f_n$ in relation to the time partition $t_1, t_2, \ldots, t_n$.

Chapter 8

`ENCLOSURE_RADIATION` Namelist

Overview

`ENCLOSURE_RADIATION` Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Multiple, one for each enclosure.

Components

- `Name`
- `Enclosure_File`
- `Coord_Scale_Factor`
- `Skip_Geometry_Check`
- `Ambient_Constant`
- `Ambient_Function`
- `Error_Tolerance`
- `Precon_Method`
- `Precon_Iter`
- `Precon_Coupling_Method`
- `toolpath`

`Name`

Description: A unique name for this enclosure radiation system.

Type: string (31 characters max)

Default: none

`Enclosure_File`

Description: The path to the enclosure file. This is interpreted relative to the Truchas input file unless this is an absolute path. If operating in moving enclosure mode (see **`toolpath`**) the file name is the base name for a collection of enclosure files.

Type: string (255 characters max)

Default: none

Notes: The `genre` program from the RADE tool suite can be used to generate this file.

Coord_Scale_Factor

Description: An optional factor with which to scale the node coordinates of the enclosure surface.

Type: real

Default: 1.0

Valid values: > 0.0

Notes: The faces of the enclosure surface must match faces from the Truchas mesh. If the coordinates of the mesh are being scaled, it is likely that the same scaling needs to be applied to the enclosure surface.

Ambient_Constant

Description: The constant value of the ambient environment temperature.

Physical dimension: Θ

Type: real

Default: none

Valid values: $\geq$ `Absolute_Zero`

Notes: Either `Ambient_Constant` or `Ambient_Function` must be specified, but not both. Currently this is necessary even for full enclosures, although in that case the value will not be used and any value is acceptable.

Ambient_Function

Description: The name of a `FUNCTION` namelist that defines the ambient environment temperature function. That function is expected to be a function of t alone.

Type: string

Default: none

Valid values:

Notes: Either `Ambient_Function` or `Ambient_Constant` must be specified, but not both. Currently this is necessary even for full enclosures, although in that case the value will not be used and any constant value is acceptable.

Error_Tolerance

Description: The error tolerance ϵ for the iterative solution of the linear radiosity system $Aq = b$. Iteration stops when the approximate radiosity q satisfies $||b - Aq||_2 < \epsilon||b||_2$.

Type: real

Default: `1.0e-3`

Valid values: > 0

Notes: The Chebyshev iterative method is used when solving the radiosity system in isolation with given surface temperatures. However the usual case has the radiosity system as just one component of a larger nonlinear system that is solved by a Newton-like iteration, and this condition on the radiosity component is one necessary condition of the complete stopping criterion of the iteration.

`toolpath`

Description: The name of a `TOOLPATH` namelist that defines a time-dependent motion that has been suitably partitioned. If this variable is specified it enables the moving enclosure mode of operation. This works in concert with the `genre` program.

`Precon_Method` (Expert Parameter)

Description: Preconditioning method for the radiosity system. *Use the default.*

Type: string

Default: `"jacobi"`

Valid values: `"jacobi"`, `"chebyshev"`

Notes: The preconditioner for the fully-coupled heat transfer/enclosure radiation system NLK solver is built from smaller preconditioning pieces, one of which is a preconditioner for the radiosity system alone. The least costly and seemingly most effective is Jacobi.

`Precon_Iter` (Expert Parameter)

Description: The number of iterations of the `Precon_Method` method to apply as the radiosity system preconditioner. *Use the default.*

Type: integer

Default: 1

Valid values: ≥ 1

`Precon_Coupling_Method` (Expert Parameter)

Description: Method for coupling the radiosity and heat transfer system preconditioners. *Use the default.*

Type: string

Default: `"backward GS"`

Valid values: `"jacobi"`, `"forward GS"`, `"backward GS"`, `"factorization"`

Notes: There are several methods for combining the independent preconditionings of the radiosity system and heat transfer system to obtain a preconditioner for the fully-coupled system. If we view it as a block system with the the radiosity system coming first, the first three methods correspond to block Jacobi, forward block Gauss-Seidel, and backward Gauss-Seidel updates. The factorization method is an approximate Schur complement update, that looks like block forward Gauss-Seidel followed by the second half of block backward Gauss-Seidel.

`Skip_Geometry_Check` (Expert Parameter)

Description: Normally the geometry of the enclosure surface faces are compared with boundary faces of the heat conduction mesh to ensure they actually match. Setting this variable to false will disable this check, which may be necessary in some unusual use cases.

Type: `logical`

Default: `.false.`

Chapter 9

ENCLOSURE_SURFACE Namelist

Overview

ENCLOSURE_SURFACE Namelist Features

Required/Optional: Required when ENCLOSURE_RADIATION namelists are active

Single/Multiple Instances: Multiple

Components

- Name
- Enclosure_Name
- Face_Block_IDs
- Emissivity_Constant
- Emissivity_Function

Name

Description: A unique name for this enclosure surface.

Type: string (31 characters max)

Default: none

Enclosure_Name

Description: The name of the ENCLOSURE_RADIATION namelist that defines the enclosure radiation system to which this surface belongs.

Type: string

Default: none

Face_Block_IDs

Description: A list of face block IDs that define this enclosure surface

Type: a list of up to 32 integers

Default: none

Valid values: any valid face block ID from the enclosure file

Notes: The surface faces in an enclosure file are divided into blocks. When the `genre` program from the RadE tool suite is used to create this file, these blocks are automatically generated; each block corresponds to the Exodus II mesh side set used to defined it, and the side set ID is assigned as the face block ID. In this case, then, the IDs that can be specified here will be certain side set IDs from the Truchas Exodus II mesh file.

`Emissivity_Constant`

Description: The constant emissivity value for this enclosure surface.

Physical dimensions: dimensionless

Type: `real`

Default: none

Valid values: (0.0, 1.0]

Notes: Either `Emissivity_Constant` or `Emissivity_Function` must be specified, but not both.

`Emissivity_Function`

Description: The name of a `FUNCTION` namelist that defines the emissivity function. That function is expected to be a function of (t, x, y, z).

Type: string

Default: none

Notes: Either `Emissivity_Function` or `Emissivity_Constant` must be specified, but not both.

Chapter 10

EVAPORATION Namelist (Experimental)

Overview

This namelist defines a special heat flux boundary condition that models heat loss due to the evaporation of material. Its intended use is in the simulation of additive manufacturing or welding processes where the laser heat source can produce localized surface temperatures approaching and exceeding the boiling temperature. The form of the heat flux is an Arrhenius-type function

$$f(T) = AT^{\beta}e^{-E_a/RT}, \tag{10.1}$$

where T is temperature, R the gas constant, and A, β, and E_a are model parameters defined by input. When using this model, the simulation should be using Kelvin for temperature.

EVAPORATION Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Single

Components

- Face_Set_IDs
- Prefactor
- Temp_Exponent
- Activation_Energy

Face_Set_IDs

Description: A list of face set IDs that define the subset of the boundary where the evaporation boundary condition model will be imposed.

Type: a list of up to 32 integers

Default: none

Valid values: any valid mesh face set ID

Note: Exodus II mesh side sets are interpreted by Truchas as face sets having the same IDs.

Prefactor

Description: The prefactor A.

Type: real

Default: none

Temp_Exponent

Description: The temperature exponent β.

Type: real

Default: none

Activation_Energy

Description: The activation energy E_a. *This value must be specified in Joules per mole units, regardless of the units used elsewhere in the simulation.*

Type: real

Default: none

Chapter 11

FLOW Namelist

Overview

The FLOW namelist specifies the parameters for the fluid flow model and algorithm. This namelist is read whenever the PHYSICS namelist option Flow is enabled. Parameters for the linear solvers employed by the algorithm are specified using FLOW_VISCOUS_SOLVER and FLOW_PRESSURE_SOLVER namelists. Flow boundary conditions are defined using FLOW_BC namelists.

FLOW Namelist Features

Required/Optional: Required when flow physics is enabled.
Single/Multiple Instances: Single

Components

Physics Options

- inviscid

Numerical parameters

- courant_number
- viscous_number
- viscous_implicitness
- track_interfaces
- material_priority
- vol_track_subcycles
- nested_dissection (expert)
- vol_frac_cutoff (expert)
- fischer_dim (expert)
- fluid_frac_threshold (expert)
- min_face_fraction (expert)
- void_collapse (experimental)
- void_collapse_relaxation (experimental)
- wisp_redistribution (experimental)
- wisp_cutoff (experimental)
- wisp_neighbor_cutoff (experimental)

inviscid

Description: This option omits the viscous forces from the flow equations. In this case there is no viscous system to solve and the `FLOW_VISCOUS_SOLVER` namelist is not required.

Type: `logical`

Default: `.false.`

courant_number

Description: This parameter sets an upper bound on the time step that is associated with stability of the explicit fluid advection algorithm. A value of 1 corresponds (roughly) to the stability limit, with smaller values resulting in proportionally smaller allowed time steps. Truchas uses the largest time step possible subject to this and other limits.

Type: `real`

Default: 0.5

Valid values: $(0.0, 1.0]$

Notes: The Courant number for a cell is the dimensionless value $C_i = u_i \Delta t / \Delta x_i$ where Δt is the time step, u_i the fluid velocity magnitude on the cell, and Δx_i a measure of the cell size. The time step limit is the largest Δt such that $\max\{C_i\}$ equals the value of `courant_number`.

The interpretation of u_i and Δx_i for a general cell is somewhat sticky. Currently a ratio u_f/h_f is computed for each face of a cell and the maximum taken for the value of $u_i/\Delta x_i$. Here u_f is the normal fluxing velocity on the face, and h_f is the inscribed cell height at the face; that is, the minimum normal distance between the face and cell nodes not belonging to the face.

viscous_number

Description: This parameter sets an upper bound on the time step that is associated with stability of an explicit treatment of viscous flow stress tensor. A value of 1 corresponds roughly to the stability limit, with smaller values resulting in proportionally smaller allowed time steps. Truchas uses the largest time step possible subject to this and other limits.

For an implicit treatment of the viscous flow stress tensor with `viscous_implicitness` at least $\frac{1}{2}$, which is always strongly recommended, the viscous discretization is unconditionally stable and *no time step limit is needed*. In this case, the parameter can still be used to limit the time step for *accuracy*. A value of 0 will disable this limit entirely.

Type: `real`

Default: 0

Valid values: ≥ 0

Notes: The viscous number for a cell is the dimensionless value $V_i = \nu_i \Delta t / {\Delta x_i}^2$, where Δt is the time step, ν_i the kinematic viscosity (μ/ρ) on the cell, and Δx_i a measure of the cell size. The time step limit is the largest Δt such that $\max\{V_i\}$ equals the value of `viscous_number`. Currently the measure of cell size mirrors that used in the definition of the `courant_number`, namely that Δx_i is taken as the minimum of the inscribed heights h_f for the faces of the cell.

viscous_implicitness

Description: The degree of time implicitness θ used for the velocity field in the discretization of the viscous flow stress tensor in the fluid momentum conservation equation. The velocity is given by the θ-method, $\mathbf{u}_\theta = (1-\theta)\mathbf{u}_n - \theta\mathbf{u}_{n+1}$: $\theta = 0$ gives an explicit discretization and $\theta = 1$ a fully implicit discretization. In practice only the values 1, $\frac{1}{2}$ (trapezoid method), and 0 are useful, and use of the latter explicit discretization is generally not recommended. Note that an implicit discretization,

$\theta > 0$, will require the solution of a linear system; see FLOW_VISCOUS_SOLVER. This parameter is not relevant to inviscid flow problems.

Type: real

Default: 1

Valid values: $[0, 1]$

Notes: The discretization is first order except for the trapezoid method ($\theta = \frac{1}{2}$) which is second order. However note that the flow algorithm overall is only first order irrespective of the choice of θ.

The advanced velocity $\mathbf{u}_{n+1}$ is actually the predicted velocity $\mathbf{u}^{\star}_{n+1}$ from the predictor stage of the flow algorithm.

track_interfaces

Description: This option enables the tracking of material interfaces. The default is to track interfaces whenever the problem involves more than one material. If the problem involves a single fluid and it is known a priori that there will never be any mixed material cells containing fluid, then this option can be set to false to short-circuit some unnecessary work, but otherwise the default should be used.

Type: logical

Default: .true.

Notes:

material_priority

Description: A list of material names that defines the priority order in which material interfaces are reconstructed within a cell for volume tracking. All fluid material names must be listed, and if the problem includes any non-fluid materials, this list must include the case-sensitive keyword "SOLID", which stands for all non-fluid materials lumped together. The default is the list of fluid materials in input file order, followed by "SOLID" for the lumped non-fluids.

Type: string list

Notes: Different priorities will result in somewhat different results. Unfortunately there are no hard and fast rules for selecting the priorities.

vol_track_subcycles

Description: The number of sub-time steps n taken by the volume tracker for every time step of the flow algorithm. If the flow time step size is Δt then the volume tracker will take n time steps of size $\Delta t/n$ to compute the net flux volumes and advance the volume fractions for the flow step.

Type: integer

Default: 2

Notes: With the current unsplit advection algorithm [1] it is necessary to sub-cycle the the volume tracking time integration method in order to obtain good "corner coupling" of the volume flux terms.

nested_dissection (Expert Parameter)

Description: This option enables use of the nested dissection algorithm to reconstruct material interfaces in cells containing 3 or more materials. If set false the less accurate and less expensive onion skin algorithm will be used.

Type: logical

Default: .true.

vol_frac_cutoff (Expert Parameter)

Description: The smallest material volume fraction allowed. If a material volume fraction drops below this cutoff, the material is removed entirely from the cell, and its volume fraction replaced by proportional increases to the volume fractions of the remaining materials, or if the cell contains void, by increasing the void volume fraction alone.

Type: `real`

Valid values: $(0, 1)$

Default: 10^{-8}

fischer_dim (Expert Parameter)

Description: The dimension d of the subspace used in Fischer's projection method [2] for computing an initial guess for pressure projection system based on previous solutions. Memory requirements for the method are $2(d + 1)$ cell-based vectors. Set this variable to 0 to disable use of this method.

Type: `integer`

Default: 6

fluid_frac_threshold (Expert Parameter)

Description: Cells with a total fluid volume fraction less than this threshold are ignored by the flow solver, being regarded as 'solid' cells.

Type: `real`

Valid values: $(0, 1)$

Default: 10^{-2}

min_face_fraction (Expert Parameter)

Description: The variable sets the minimum value of the fluid density associated with a face for the pressure projection system. It is specified as a fraction of the minimum fluid density (excluding void) of any fluid in the problem.

Type: `real`

Default: 10^{-3}

void_collapse (Experimental)

Description: The volume-of-fluid algorithm effectively treats small fragments of void entrained in fluid as incompressible, resulting in unphysical void "bubbles" that persist in the flow. A model that drives the collapse of these void fragments will be enabled when this variable is set to true. See `void_collapse_relaxation` for a model parameter.

Type: `logical`

Default: `.false.`

void_collapse_relaxation (Experimental)

Description: The relaxation parameter in the void collapse model. See `void_collapse`.

Type: `real`

Default: 0.1

Valid values: $[0, 1]$

Notes: The relaxation factor is roughly inversely proportional to the number of timesteps required for all the void in a cell to collapse as dictated by inertial forces. Thus a relaxation factor of 1 would allow for all the void in a cell to collapse over a single timestep. A factor of 0.1 would allow for the void in a cell to collapse over the course of 10 timesteps. Larger values tend to cause more mass loss (on the order of 0.5%), although the results do improve with increased subcycling.

`wisp_redistribution` (Experimental)

Description: A cell containing a small amount of fluid (wisps) can sometimes trigger pathological behavior, manifesting as a perpetual acceleration that drives the timestep to 0. A model that redistributes these wisps to other other fluid cells will be enabled when this variable is set to true. See `wisp_cutoff` and `wisp_neighbor_cutoff` model parameters.

Type: `logical`

Default: `.false.`

`wisp_cutoff` (Experimental)

Description: Fluid cells with a fluid volume fraction below this value may be treated as wisps and have their fluid material moved around to other fluid cells. Generally, moving fluid around the domain can have an undesirable effect on the flow physics. It is therefore advisable to keep this value as small as possible. Based on numerical experimentation, the recommended value is 0.05. Smaller values, such as, 0.01 did not result in robust simulations. See `wisp_redistribution`.

Type: `real`

Default: 0.05

`wisp_neighbor_cutoff` (Experimental)

Description: Fluid cells with a fluid volume fraction below `wisp_cutoff` can only be considered a wisp if the amount of fluid in the neighboring cells is also "small". This definition of "small" is controlled by `wisp_neighbor_cutoff`. In addition, for a a cell to receive wisp material, the fluid volume fraction of it and its neighbors must be larger than `wisp_neighbor_cutoff`. See `wisp_redistribution`.

Type: `real`

Default: 0.25

Chapter 12

FLOW_BC Namelist

Overview

The FLOW_BC namelist is used to define boundary conditions for the fluid flow model at external boundaries. At inflow boundaries it also specifies the value of certain intensive material quantities, like temperature, that may be associated with other physics models.

Each instance of the namelist defines a particular condition to impose over a subset Γ of the domain boundary. The boundary subset Γ is specified using mesh face sets. The namelist variable face_set_ids takes a list of face set IDs, and the boundary condition is imposed on all faces belonging to those face sets. Note that ExodusII mesh sides sets are imported into Truchas as face sets with the same IDs. The following common types of boundary conditions can be defined:

- *Pressure.* A pressure Dirichlet condition $p = p_b$ on Γ is defined by setting type to "pressure". The boundary value p_b is specified using either pressure for a constant value, or pressure_func for a function.
- *Velocity.* A velocity Dirichlet condition $\mathbf{u} = \mathbf{u}_b$ on Γ is defined by setting type to "velocity". The boundary value $\mathbf{u}_b$ is specified using either velocity for a constant value, or velocity_func for a function.
- *No slip.* The special velocity Dirichlet condition $\mathbf{u} = 0$ on Γ is defined by setting type to "no-slip".
- *Free slip.* A free-slip condition where fluid is not permitted to penetrate the boundary, $\hat{n} \cdot \mathbf{u} = 0$ on Γ, where $\hat{n}$ is the unit normal to Γ, but is otherwise free to slide along the boundary (no tangential traction) is defined by setting type to "free-slip".
- *Tangential surface tension.*

These boundary condition types are mutually exclusive: namely, no two types may be defined on overlapping subsets of the boundary. Any subset of the boundary not explicitly assigned a boundary condition will be implicitly assigned a free-slip condition.

Currently it is only possible to assign boundary conditions on the external mesh boundary. However in many multiphysics applications the boundary of the fluid flow domain will not coincide with the boundary of the larger problem mesh. In some cases the boundary will coincide with an internal mesh-conforming interface that separates fluid cells and solid (non-fluid) cells, where a boundary condition could conceivably be assigned. In other cases, typically those involving phase change, the boundary is only implicit, passing through mixed fluid/solid cells, and will not conform to the mesh. In either case, the flow algorithm aims to impose an effective no-slip condition for viscous flows, or a free-slip condition for inviscid flows. A possible modeling approach in the former mesh-conforming case is to define an internal mesh interface using the MESH namelist variable interface_side_sets. This effectively creates new external mesh boundary where flow boundary conditions can be assigned.

`FLOW_BC` Namelist Features

Required/Optional: Optional
Single/Multiple Instances: Multiple

Components

- `name`
- `face_set_ids`
- `type`
- `pressure`
- `pressure_func`
- `velocity`
- `velocity_func`
- `dsigma`
- `inflow_material`
- `inflow_temperature`

`name`

Description: A unique name used to identify a particular instance of this namelist
Type: string (31 characters max)
Default: none

`face_set_ids`

Description: A list of face set IDs that define the portion of the boundary where the boundary condition will be imposed.
Type: integer list (32 max)
Default: none

`type`

Description: The type of boundary condition. The available options are:

`"pressure"` Pressure is prescribed on the boundary. Use `pressure` or `pressure_func` to specify its value.

`"velocity"` Velocity is prescribed on the boundary. Use `velocity` or `velocity_func` to specify its value.

`"no-slip"` 0-velocity is imposed on the boundary. This is incompatible with inviscid flow.

`"free-slip"` No velocity normal to the boundary, but the tangential velocity is otherwise free (no traction forces).

`"marangoni"` Like "free-slip" except a tangential traction is applied that is due to temperature dependence of surface tension. Use `dsigma` to specify the value of $d\sigma/dT$. This is incompatible with inviscid flow.

Type: string
Default: none
Notes: The different boundary condition types are mutually exclusive; no two can be specified on a common portion of the boundary.

`pressure`

Description: The constant value of boundary pressure for a pressure-type boundary condition. To specify a function, use `pressure_func` instead.

Default: none

Type: real

`pressure_func`

Description: The name of a `FUNCTION` namelist defining a function that gives the boundary pressure for a pressure-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

`velocity`

Description: The constant value of boundary velocity for a velocity-type boundary condition. To specify a function, use `velocity_func` instead.

Default: none

Type: real 3-vector

`velocity_func`

Description: The name of a `VFUNCTION` namelist defining a function that gives the boundary velocity for a velocity-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

`dsigma`

Description: The constant value of $d\sigma/dT$ for the marangoni-type condition. Here $\sigma(T)$ is the temperature dependent surface tension coefficient.

Default: none

Type: real

Units: ???

`inflow_material`

Description: Velocity and pressure boundary conditions may result in fluid flow into the domain across the boundary. This parameter specifies the name of the fluid material to flux in. If not specified, materials are fluxed into a cell through a boundary face in the same proportion as the material volume fractions present in the cell.

Default: none

inflow_temperature

Description: Velocity and pressure boundary conditions may result in fluid flow into the domain across the boundary. This parameter specifies the temperature of the material fluxed in. If not specified, materials are fluxed into a call through a boundary face at the same temperature as the cell.

Default: none

Chapter 13

FLOW_PRESSURE_SOLVER and FLOW_VISCOUS_SOLVER Namelists

Overview

The flow algorithm requires the solution of two linear systems at each time step: the implicit viscous velocity update system and the pressure Poisson system. Truchas uses the hybrid solver from the HYPRE software library [3] to solve these systems.
The hybrid solver first uses a diagonally-scaled iterative Krylov solver. If it determines that convergence is too slow, the solver switches to a more expensive but more effective preconditioned Krylov solver that uses an algebraic multigrid (AMG) preconditioner (BoomerAMG).
The `FLOW_VISCOUS_SOLVER` namelist sets the HYPRE hybrid solver parameters for the solution of the implicit viscous velocity update system, and the `FLOW_PRESSURE_SOLVER` namelist sets the solver parameters for the solution of the pressure Poisson system. The same variables are used in both namelists.

FLOW_VISCOUS_SOLVER Namelist Features

Required/Optional: Required only for viscous flow with `viscous_implicitness` > 0.

Single/Multiple Instances: Single

FLOW_PRESSURE_SOLVER Namelist Features

Required/Optional: Required

Single/Multiple Instances: Single

krylov_method

Description: Selects the Krylov method used by the HYPRE hybrid solver. The options are `"cg"` (default), `"gmres"`, and `"bicgstab"`.

krylov_dim

Description: The Krylov subspace dimension for the restarted GMRES method.

Type: `integer`

Valid values: > 0

Default: `5`

conv_rate_tol

Description: The convergence rate tolerance θ where the hybrid solver switches to the more expensive AMG preconditioned Krylov solver. The average convergence rate after n iterations of the diagonally-scaled Krylov solver is $\rho_n = \left(\|r_n\|/\|r_0\|\right)^{1/n}$, where $r_n = Ax_n - b$ is the residual of the linear system, and its convergence is considered too slow when

$$\left[1 - \frac{|\rho_n - \rho_{n-1}|}{\max(\rho_n, \rho_{n-1})}\right] \rho_n > \theta$$

Type: `real`

Valid values: $(0, 1)$

Default: 0.9

abs_tol, rel_tol

Description: The absolute and relative error tolerances ϵ_1 and ϵ_2 for the solution of the linear system. The test for convergence is $\|r\| \le \max\{\epsilon_1, \epsilon_2\|b\|\}$, where $r = Ax - b$ is the residual of the linear system.

Type: `real`

Default: None

Note:

max_ds_iter

Description: The maximum number of diagonally scaled Krylov iterations allowed. If convergence is not achieved within this number of iterations the hybrid solver will switch to the preconditioned Krylov solver.

Type: `integer`

Default:

max_amg_iter

Description: The maximum number of preconditioned Krylov iterations allowed.

Type: `integer`

Default:

print_level

Description: Set this parameter to 2 to have HYPRE write diagnostic data to the terminal for each solver iteration. This is only useful in debugging situations. The default is 0, no output.

Additional HYPRE parameters (Expert)

Some additional HYPRE solver parameters and options can be set using these namelists. Nearly all of these are associated with the BoomerAMG preconditioner, and all have reasonable defaults set by HYPRE. See the *ParCSR Hybrid Solver* section in the HYPRE reference manual [4] for details. The HYPRE user's manual [5] has some additional information. The variables that can be set are listed below. Note that the variables correspond to similarly-named HYPRE library functions and not actual HYPRE variables. Also note that there are many parameters and options that cannot currently be set by the namelists.

cg_use_two_norm (logical)
amg_strong_threshold (real)
amg_max_levels (integer)
amg_coarsen_method (integer)
amg_smoothing_sweeps (integer)
amg_smoothing_method (integer)
amg_interp_method (integer)

Chapter 14

`FUNCTION` Namelist

Overview

There is often a need to specify phase properties, boundary condition data, source data, etc., as functions rather than constants. The `FUNCTION` namelist provides a means for defining functions that can be used in many situations.

The namelist can define several types of functions: a multi-variable polynomial, a continuous piecewise linear function defined by a table of values, a smooth step function, and with certain Truchas build configurations, a general user-provided function from a shared object library that is dynamically loaded at runtime. The functions are functions of m variables. The expected number of variables and what unknowns they represent (i.e., temperature, time, x-coordinate, etc.) depends on the context in which the function is used, and this will be detailed by the documentation of those namelists where these functions can be used.

Polynomial Function. This function is a polynomial in the variables $\mathbf{v} = (v_1, \ldots, v_m)$ of the form

$$f(\mathbf{v}) = \sum_{j=1}^{n} c_j \prod_{i=1}^{m} (v_i - a_i)^{e_{ij}} \tag{14.1}$$

with coefficients c_j, integer-valued exponents e_{ij}, and arbitrary reference point $\mathbf{a} = (a_1, \ldots, a_m)$. The coefficients are specified by `Poly_Coefficients`, the exponents by `Poly_Exponents`, and the reference point by `Poly_Refvars`.

Tabular Function. This is a continuous, single-variable function $y = f(x)$ interpolated from a sequence of data points (x_i, y_i), $i = 1, \ldots, n$, with $x_i < x_{i+1}$. A smooth interpolation method is available in addition to the standard linear interpolation; see `Tabular_Interp`. There are also two different methods for extrapolating on $x < x_1$ and $x > x_n$; see `Tabular_Extrap`.

Smooth Step Function. This function is a smoothed (C_1) step function in a single variable $\mathbf{v} = (x)$ of the form

$$f(x) = \begin{cases} y_0 \text{ if } x \le x_0, \\ y_0 + (y_1 - y_0)s^2(3 - 2s), s \equiv (x - x_0)/(x_1 - x_0), \text{ when } x \in (x_0, x_1), \\ y_1 \text{ if } x \ge x_1, \end{cases} \tag{14.2}$$

with parameters x_0, x_1, y_0, and y_1.

Shared Library Function. This is a function from a shared object library having a simple Fortran 77 or C compatible interface. Written in Fortran 77 the function interface must look like

```
double precision function myfun (v, p) bind(c)
  double precision v(*), p(*)
```

where `myfun` can, of course, be any name. The equivalent C interface is

```
double myfun (double v[], double p[]);
```

The vector of variables $\mathbf{v} = (v_1, \ldots, v_m)$ is passed in the argument `v` and a vector of parameter values specified by `Parameters` is passed in the argument `p`. Instructions for compiling the code and creating a shared object library can be found in the *Truchas Installation Guide*. The path to the library is given by `Library_Path` and the name of the function (`myfun`, e.g.) is given by `Library_Symbol`. Note that the `bind(c)` attribute on the function declaration inhibits the Fortran compiler from mangling the function name (by appending an underscore, for example) as it normally would.

`FUNCTION` Namelist Features

Required/Optional: Optional
Single/Multiple Instances: Multiple

Components

- `Name`
- `Type`
- `Library_Path`
- `Library_Symbol`
- `Parameters`
- `Poly_Coefficients`
- `Poly_Exponents`
- `Poly_Refvars`
- `Smooth_Step_X0`
- `Smooth_Step_X1`
- `Smooth_Step_Y0`
- `Smooth_Step_Y1`
- `Tabular_Data`
- `Tabular_Dim`
- `Tabular_Extrap`
- `Tabular_Interp`

`Name`

Description: A unique name by which this function can be referenced by other namelists.
Type: A case-sensitive string of up to 31 characters.
Default: None

`Type`

Description: The type of function defined by the namelist.
Type: case-sensitive string
Default: none
Valid values: `"polynomial"` for a polynomial function, `"tabular"` for a tabular function, `"smooth step"` for a smooth step function, or `"library"` for a function from a shared object library. The `"library"` value is not available with a Truchas executable built with the “dynamic loading” option disabled.

Library_Path

Description: The path to the shared object library that contains the function.

Type: A string of up to 128 characters.

Default: none

Library_Symbol

Description: The symbol name of the function within the shared object file.

Type: A string of up to 128 characters.

Default: none

Notes: Unless the Fortran function is declared with the `BIND(C)` attribute, which is the recommended practice, a Fortran compiler will almost always mangle the name of the function so that the symbol name is not quite the same as the name in the source code. Use the UNIX/Linux command-line utility `nm` to list the symbol names in the library file to determine the correct name to use here.

Parameters

Description: Optional parameter values to pass to the shared library function.

Type: real vector of up to 16 values

Default: None

Poly_Coefficients

Description: The coefficients c_j of the polynomial (14.1).

Type: real vector of up to 64 values

Default: None

Poly_Exponents

Description: The exponents e_{ij} of the polynomial (14.1).

Type: integer array

Default: None

Notes: Namelist array input is very flexible. The syntax

```
Poly_Exponents(i,j) = ...
```

defines the value for exponent e_{ij}. All the variable exponents for coefficient j can be defined at once by listing their values with the syntax

```
Poly_Exponents(:,j) = ...
```

In some circumstances it is possible to omit providing 0-exponents for variables that are unused. For example, if the function is expected to be a function of (t, x, y, z), but a polynomial in only t is desired, one can just define a 1-variable polynomial and entirely ignore the remaining variables. On the other hand, if a polynomial in z is desired, one must specify 0-valued exponents for all the preceding variables.

Poly_Refvars

Description: The optional reference point **a** of the polynomial (14.1).

Type: real vector

Default: 0.0

Tabular_Data

Description: The table of values (x_i, y_i) defining a tabular function $y = f(x)$. See also Tabular_Dim, Tabular_Interp, and Tabular_Extrap for additional variables that define the function.

Type: real array

Default: none

Notes: This is a $2 \times n$ array with $n \leq 100$. Namelist array input is very flexible and the values can be specified in several ways. For example, the syntax

Tabular_Data(1,:) = $x_1,\ x_2,\ \ldots,\ x_n$
Tabular_Data(2,:) = $y_1,\ y_2,\ \ldots,\ y_n$

specifies the x_i and y_i values as separate lists. Or the values can be input naturally as a table

Tabular_Data = $x_1,\quad y_1$
$x_2,\quad y_2$
$\ldots$
$x_n,\quad y_n$

The line breaks are unnecessary, of course, and are there only for readability as a table.

Tabular_Dim

Description: The dimension in the m-vector of independent variables that serves as the independent variable for the single-variable tabular function.

Type: integer

Default: 1

Notes: Functions defined by this namelist are generally functions of m variables $(v_1, v_2, \ldots, v_m)$. The number of variables and the unknowns to which they correspond depend on the context where the function is used. One of these variables needs to be selected to be the independent variable used for the tabular function. In typical use cases the desired tabular function will depend on time or temperature. Those unknowns are often the first variable, and the default value of `Tabular_Dim` is appropriate.

Tabular_Extrap

Description: Specifies the method used for extrapolating outside the range of tabular function data points.

Type: case-insensitive string

Default: `"nearest"`

Valid values: `"nearest"`, `"linear"`

Notes: Nearest extrapolation continues the y value at the first or last data point. Linear extrapolation uses the slope of the first or last data interval, or if Akima smoothing is used (see Tabular_Interp) the computed slope at the first or last data point.

`Tabular_Interp`

Description: Specifies the method used for interpolating between tabular function data points.

Type: case-insensitive string

Default: `"linear"`

Valid values: `"linear"`, `"akima"`

Notes: Akima interpolation [6] uses Hermit cubic interpolation on each data interval, with the slope at each data point computed from the linear slopes on the neighboring four intervals. The resulting function is C^1 smooth. To determine the slope at the first two and last two data points, two virtual data intervals are generated at the beginning and at the end using quadratic extrapolation.

The algorithm seeks to avoid undulations in the interpolated function where the data suggests a flat region though its choice of slopes at data points. Figure 14.1A shows the typical smooth Akima interpolation. If the first interval was expected to be flat, the exhibited undulation would likely be unacceptable. By inserting an additional data point to create successive intervals with the same slope, as in Figure 14.1BC, the algorithm identifies it as a flat region and preserves it in the interpolation. Where two flat regions with differing slopes meet, it is impossible to simultaneously retain smoothness and preserve flatness. In this case a modification to the Akima algorithm used by Matlab's `tablelookup` function is adopted, which gives preference to the region with smaller slope as shown in Figure 14.1D.

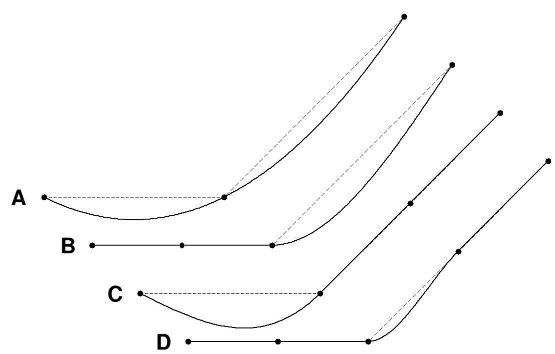


Figure 14.1: Examples of smooth Akima interpolation.

`Smooth_Step_X0`

Description: The parameter x_0 of the function (14.2).

Type: real

Default: none

Valid values: Require only $x_0 < x_1$.

`Smooth_Step_X1`

Description: The parameter x_1 of the function (14.2).

Type: real

Default: none

Valid values: Require only $x_0 < x_1$.

Smooth_Step_Y0

Description: The parameter y_0 of the function (14.2).
Type: real
Default: none

Smooth_Step_Y1

Description: The parameter y_1 of the function (14.2).
Type: real
Default: none

Chapter 15

INDUCTION_COIL Namelist

Overview

The variables in an INDUCTION_COIL namelist specify the physical characteristics of an induction coil that is to produce an external magnetic field to drive the electromagnetic Joule heat calculation. Figure 15.1 shows the idealized model of a coil that is used to analytically evaluate the driving field. The coil axis is assumed to be oriented with the problem symmetry axis as defined in the ELECTROMAGNETICS namelist. Multiple coils may be specified; the net driving field is the superposition of the fields due to the individual coils, and a spatially uniform field that can be specified in the ELECTROMAGNETICS namelist.

The coils carry a sinusoidally-varying current with a common frequency and phase. The frequency is specified in the ELECTROMAGNETICS namelist, while the phase value is irrelevant due to the time averaging of the calculated Joule heat. Each coil, however, has an independent current amplitude which is specified here. In addition, the current and frequency may be piecewise constant functions of time.

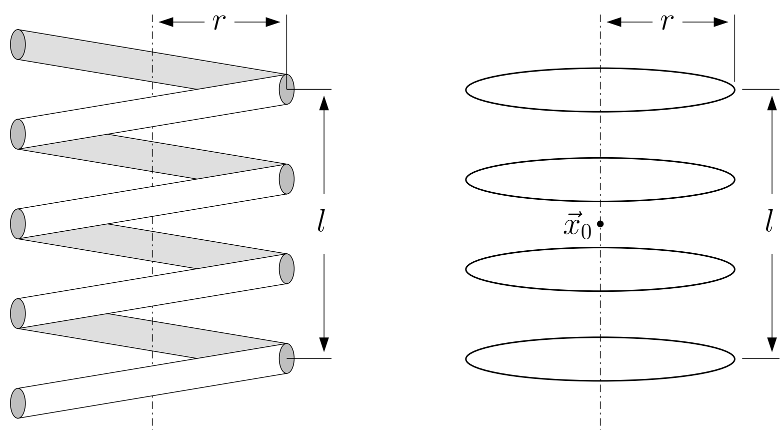


Figure 15.1: Physical 4-turn helical coil with extended wire cross section (left), and its idealized model as a stacked array of circular current loops (right).

INDUCTION_COIL Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Multiple

Components

- Center
- Current
- Length
- NTurns
- Radius

Center

Description: A 3-vector $\mathbf{x}_0$ giving the position of the center of the coil; cf. Figure 15.1.

Physical dimension: L

Type: real

Default: $(0, 0, 0)$

Valid values: any 3-vector

Current

Description: Amplitude of the sinusoidally-varying current in the coil.

Physical dimension: I

Type: real

Default: none

Valid values: Any single value, or any sequence of values.

Notes: A sequence of up to 32 values may be assigned to this variable in order to specify a time-dependent current amplitude; see Source_Times in the ELECTROMAGNETICS namelist and Fig. 7.1.

Length

Description: Length l of the coil; cf. Figure 15.1.

Physical dimension: L

Type: real

Default: none

Valid values: $(0, \infty)$

Notes: Length is not required, nor meaningful, if NTurns is 1.

NTurns

Description: Number of turns of the coil; cf. Figure 15.1.

Type: integer

Default: none

Valid values: Any positive integer.

`Radius`

Description: Radius r of the coil; cf. Figure 15.1.

Physical dimension: `L`

Type: `real`

Default: none

Valid values: $(0, \infty)$

Chapter 16

MATERIAL and PHASE Namelists

Overview

A database of materials, their properties and attributes are defined using the MATERIAL, PHASE, and PHASE_CHANGE namelists. Not all materials are necessarily included in the simulation, but only those specified by the PHYSICS namelist variable materials. This allows one to reuse material input blocks without needing to prune out unused materials which would otherwise negatively impact performance.
In Truchas usage, one or more phases comprise a material. A single-phase material is defined by a MATERIAL namelist, and the namelist specifies all properties and attributes of the material. A multi-phase material is defined by a MATERIAL namelist and an optional PHASE namelist for each of the material phases. Properties and attributes that apply to all phases can be defined in the MATERIAL namelist. Properties and attributes specific to a given phase are defined in the PHASE namelist for the phase, and these supercede any that might be defined in the MATERIAL namelist for the phase. Additional information that defines the transformation between phases is specified using the PHASE_CHANGE namelist.

The MATERIAL Namelist

The MATERIAL namelist defines a material, either single-phase or multi-phase, that is available to be used in a simulation. In addition to the property and attribute variables described below, the namelist has the following variables.

name

A unique name for the material used to reference it.

phases

A multi-phase material is defined by assigning a value to **phases**. This is a list of two or more unique phase names that comprise the material. The phases must be listed in order from low to high temperature phases. The phase names will be referenced by PHASE_CHANGE namelists and by optional PHASE namelists. Consecutive pairs of phases must be accompanied by a corresponding PHASE_CHANGE namelist.

The PHASE Namelist

The PHASE namelist defines properties and attributes specific to one phase of a multi-phase material. It is optional. Any properties or attributes defined here supersede those defined in the parent MATERIAL namelist. In addition to the property and attribute variables described below, the PHASE namelist has the following variable.

name

The name of the phase. This is one of the names assigned to the **phases** variable of the MATERIAL namelist for the parent material.

Property and Attribute Variables

The following variables specify the values of material properties and attributes. Unless otherwise noted, they may appear in both the MATERIAL and PHASE namelists. Many properties can be either constant-valued or a function. Variables ending with the suffix **_func** specify the name of a FUNCTION namelist that defines a function that computes the value of the property.

Thermodynamic Properties

The following property variables are used by the heat transport model:

- **density**
- **specific_heat**, **specific_heat_func**, **ref_temp**, and **ref_enthalpy**
- **specific_enthalpy_func**
- **conductivity**, **conductivity_func**

The material mass density (mass per volume) is specified by **density**. It is limited to constant values and all phases in a multi-phase material are currently constrained to have the same density (but see **density_delta_func** for flow and **tm_linear_cte** for solid mechanics.) Consequently this variable may only appear in the MATERIAL namelist.

There are two options for defining the temperature-dependent specific enthalpy function $h(T)$ of a material or phase. The first specifies the specific heat $C_p(T)$ (energy per unit mass per degree temperature) using **specific_heat** for a constant or **specific_heat_func** for a function of temperature. In the latter case it must either be a tabular function or a polynomial without a T^{-1} term. An analytic antiderivative of the specific heat will be generated by Truchas and used for $h(T)$. For a single-phase material or the lowest-temperature phase of a multi-phase material, there is an arbitrary constant of integration. By default it is chosen such that $h(0) = 0$. It can be chosen instead such that $h(T_{\text{ref}}) = h_{\text{ref}}$ by specifying values for the optional variables **ref_temp** and **ref_enthalpy**, which default to 0. These latter two variables may only appear in the MATERIAL namelist.

The second option is to specify the specific enthalpy function $h(T)$ (energy per unit mass) directly using **specific_enthalpy_func**. The function must be strictly increasing, and when used for a phase of a multi-phase material, it must incorporate the latent heat associated with the transformation from the adjacent lower-temperature phase (when there is one).

The thermal conductivity (power per unit length per degree temperature) of a material or phase is specified by **conductivity** for a constant or **conductivity_func** for a function. The function is assumed to be a function of temperature T, or $(T, \phi_1, \ldots, \phi_n)$ when coupled with solutal species transport.

Fluid Flow Properties

The following attribute and property variables are used by the fluid flow model:

- **is_fluid**
- **density**
- **density_delta_func**
- **viscosity**, **viscosity_func**

The boolean attribute **is_fluid** is used to indicate whether or not the material or phase is a fluid. Its default value is F (or .false.) Materials and phases marked as fluid are included in the fluid flow model.

The constant reference density ρ_0 of a fluid material or phase is specified by **density**. This is the same density value used for heat transport. A temperature-dependent fluid density $\rho(T)$ can be defined by giving its deviation from the reference density, $\delta\rho(T) = \rho(T) - \rho_0$ using the variable **density_delta_func**. This

is used only to compute the buoyancy body force of the Boussinesq approximation in the flow model. If not specified, no deviation from the reference density is assumed.
The dynamic viscosity (mass per length per time) of a material or phase is specified by **viscosity** for a constant or **viscosity_func** for a function of temperature. This is required for viscous flow problems (FLOW namelist variable `inviscid = F`).

Electromagnetic Properties

The following property variables are used by the induction heating model:

- **electrical_conductivity**, **electrical_conductivity_func**
- **electric_susceptibility**, **electric_susceptibility_func**
- **magnetic_susceptibility**, **magnetic_susceptibility_func**

The electrical conductivity of a material or phase is specified using either **electrical_conductivity** for a constant, or **electrical_conductivity_func** for a function of temperature. The default value is 0. The electromagnetics solver assumes SI units by default and the units of electrical conductivity are Siemens per meter. To use different units, the values for the PHYSICAL_CONSTANTS namelist variables **vacuum_permeability** and **vacuum_permittivity** must be redefined appropriately.
The electric susceptibility χ_e of a material or phase is specified using either **electric_susceptibility** for a constant, or **electric_susceptibility_func** for a function of temperature. The relative permittivity is $1 + \chi_e$. This property has a default value of zero, which is appropriate in most cases.
The magnetic susceptibility χ_m of a material or phase is specified using either **magnetic_susceptibility** for a constant, or **magnetic_susceptibility_func** for a function of temperature. The relative permeability is $1 + \chi_m$. This property has a default value of zero, which is appropriate in most cases.

Thermomechanical Properties

The following properties are used by the solid mechanics model, and are only relevant to non-fluid materials and phases. Additional viscoplasticity parameters may be defined using the VISCOPLASTIC_MODEL namelist.

- **tm_ref_density**
- **tm_ref_temp**
- **tm_linear_cte**, **tm_linear_cte_func**
- **tm_lame1**, **tm_lame1_func**
- **tm_lame2**, **tm_lame2_func**

The temperature at which a material or phase is stress-free is specified by **tm_ref_temp** and its density (mass per volume) at that temperature specified by **tm_ref_density**. Its linear coefficient of thermal expansion (inverse time) is specified by **tm_linear_cte** for a constant or **tm_linear_cte_func** for a function of temperature.
The first and second Lamé constants λ and G (force per area) for a material or phase is specified by **tm_lame1** and **tm_lame2** for constants or **tm_lame1_func** and **tm_lame2_func** for functions of temperature.

Species Transport Properties

The following property variables are relevant to the solutal species transport model. The model allows for an arbitrary number of species with concentrations $\{\phi_i\}_{i=1}^n$, and the variables are arrays of length n with each element being the property for the corresponding species.

- **diffusivity**, **diffusivity_func**
- **soret**, **soret_func**

The diffusivity (area per time) of a species component in a material or phase is specified by the corresponding element of **`diffusivity`** for a constant or **`diffusivity_func`** for a function. A function is expected to be a function of all the species concentrations $(\phi_1, \ldots, \phi_n)$, or when coupled with heat transfer, a function of temperature and concentrations $(T, \phi_1, \ldots, \phi_n)$.
When coupled with heat transfer, the Soret coefficient (inverse temperature) of the thermodiffusion term for a species component in a material or phase is specified by the corresponding element of **`soret`** for a constant or **`soret_func`** for a function. A function is expected to be a function of temperature and concentrations $(T, \phi_1, \ldots, \phi_n)$. This is optional. If not specified, thermodiffusion of the species component is not included in the model, but if defined for one phase of a multi-phase material, it must be defined for all its phases.

Chapter 17

MESH Namelist

Overview

The MESH namelist specifies the common mesh used by all physics models other than the induction heating model, which uses a separate tetrahedral mesh specified by the ALTMESH namelist. For simple demonstration problems, a rectilinear hexahedral mesh of a brick domain can be defined, but for most applications the mesh will need to be generated beforehand by some third party tool or tools and saved as a file that Truchas will read. At this time Exodus II [7] is the only supported mesh format (also sometimes known as Genesis). This well-known format is used by some mesh generation tools (Cubit [8], for example) and utilities exist for translating from other formats to Exodus II. The unstructured 3D mesh may be a general mixed-element mesh consisting of non-degenerate hexehedral, tetrahedral, pyramid, and wedge/prism elements. The Exodus II format supports a partitioning of the elements into *element blocks* and also supports the definition of *side sets*, which are collections of oriented element faces that describe mesh surfaces, either internal or boundary. Extensive use is made of this additional mesh metadata in assigning materials, initial conditions, boundary conditions, etc., to the mesh.

MESH Namelist Features

Required/Optional: Required

Single/Multiple Instances: Single

Components

External mesh file

- mesh_file
- interface_side_sets
- gap_element_blocks
- exodus_block_modulus

Internally generated mesh

- x_axis
- y_axis
- z_axis
- noise_factor

Common parameters

- coordinate_scale_factor
- rotation_angles
- partitioner
- partition_file
- first_partition

External Mesh File

In typical usage, the mesh will be read from a specified Exodus II mesh file. Other input variables that follow specify optional modifications that can be made to the mesh after it is read.

mesh_file

Description: Specifies the path to the Exodus II mesh file. If not an absolute path, it will be interpreted as a path relative to the Truchas input file directory.

Type: case-sensitive string

Default: none

interface_side_sets

Description: A list of side set IDs from the ExodusII mesh identifying internal mesh surfaces that will be treated specially by the heat/species transport solver.

Type: integer list

Default: An empty list of side set IDs.

Valid values: Any side set ID whose faces are internal to the mesh.

Notes: The heat/species transport solver requires that boundary conditions are imposed along the specified surface. Typically these will be interface conditions defined by THERMAL_BC namelists, but in unusual use cases they could also be external boundary conditions defined by the same namelists. In the latter case it is necessary to understand that the solver views the mesh as having been sliced open along the specified internal surfaces creating matching pairs of additional external boundary and, where interface conditions are not imposed, boundary conditions must be imposed on *both* sides of the interface.

gap_element_blocks (deprecated)

Description: A list of element block IDs from an Exodus II mesh that are to be treated as gap elements.

Type: integer list

Default: An empty list of element block IDs.

Valid values: Any element block ID.

Notes: Any element block ID in the mesh file can be specified, but elements that are not connected such that they can function as gap elements or are not consistent with side set definitions will almost certainly result in incorrect behavior. The code does not check for these inconsistencies.

The heat/species transport solver drops these elements from its view of the mesh and treats them instead as an internal interface; see the notes to interface_side_sets. The block IDs specified here can be used as values for face_set_ids from the THERMAL_BC namelist.

exodus_block_modulus

Description: When importing an Exodus II mesh, the element block IDs are replaced by their value modulo this parameter. Set the parameter to 0 to disable this procedure.

Type: integer

Default: 10000

Valid values: ≥ 0

Notes: This parameter helps solve a problem posed by mixed-element meshes created by Cubit and Trelis. In those tools a user may define an element block comprising multiple element types. But when exported in the Exodus II format, which doesn't support blocks with mixed element types, the element block will be written as multiple Exodus II blocks, one for each type of element. One of the blocks will retain the user-specified ID of the original block. The IDs of the others will be that ID plus an offset specific to the element type. For example, if the original block ID was 1, hexahedra in the block will be written to a block with ID 1, tetrahedra to a block with ID 10001, pyramids to a block with ID 100001, and wedges to a block with ID 200001. These are the default offset values, and they can be set in Cubit/Trelis; see their documentation for details on how the IDs are generated. It is important to note that this reorganization of element blocks occurs silently and so the user may be unaware that it has happened. In order to reduce the potential for input errors, Truchas will by default convert the block IDs to congruent values modulo N in the interval $[1, N-1]$ where N is the value of this parameter. The default value 10000 is appropriate for the default configuration of Cubit/Trellis, and restores the original user-specified block IDs. Note that this effectively limits the range of element block IDs to $[1, N-1]$.

The element block IDs are modified immediately after reading the file. Any input parameters that refer to block IDs must refer to the modified IDs.

Internally Generated Mesh

A rectilinear hexahedral mesh for a brick domain $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$ can be generated internally as part of a Truchas simulation using the following input variables. The mesh is the tensor product of 1D grids in each of the coordinate directions. Each coordinate grid is defined by a coarse grid whose intervals are subdivided into subintervals, optionally with biased sizes. The generated Exodus II mesh consists of a single element block with ID 1, and a side set is defined for each of the six sides of the domain with IDs 1 through 6 for the $x = x_{\min}$, $x = x_{\max}$, $y = y_{\min}$, $y = y_{\max}$, $z = z_{\min}$, and $z = z_{\max}$ sides, respectively. Note that while the mesh is formally structured, it is represented internally as a general unstructured mesh.

x_axis, y_axis, z_axis

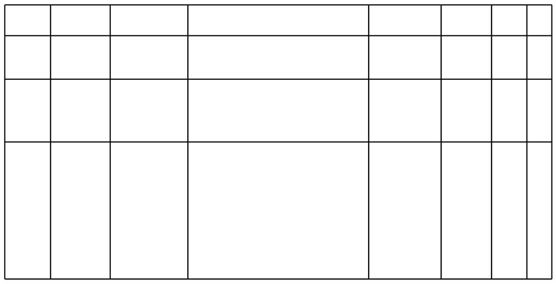
Data that describes the grid in each of the coordinate directions. The tensor product of these grids define the nodes of the 3D mesh. The data for each coordinate grid consists of these three component arrays:

`%coarse_grid`: A strictly increasing list of two or more real values that define the points of the coarse grid for the coordinate direction. The first and last values define the extent of the domain in this direction.

`%intervals`: A list of postive integers defining the number of subintervals into which each corresponding coarse grid interval should be subdivided. The number of values must be one less than the number of coarse grid points.

`%ratio`: An optional list of positive real values that define the ratio of the lengths of successive subintervals for each coarse grid interval. The default is to subdivide into equal length subintervals. If specified, the number of values must be one less than the number of coarse grid points.

See Figure 17 for an example.



```
x_axis%coarse_grid = 0.0, 0.67, 1.33, 2.0
x_axis%intervals   = 3, 1, 4
x_axis%ratio       = 1.3, 1.0, 0.7
y_axis%coarse_grid = 0.0, 0.5, 1.0
y_axis%intervals   = 1, 3
y_axis%ratio       = 0.7
z_axis%coarse_grid = 0.0, 1.0
z_axis%intervals   = 1
```

Figure 17.1: Top xy surface of the rectilinear mesh generated by the example input shown.

`noise_factor` (expert)

Description: If specified with a positive value, the coordinates of each mesh node will be perturbed by uniformly distributed random amount whose magnitude will not exceed this value times the local cell size at the node. Nodes on the boundary are not perturbed in directions normal to the boundary. This is only useful for testing.

Valid values: $\in [0, 0.3]$

Default: 0

Common Variables

The following variables apply to both types of meshes.

`coordinate_scale_factor`

Description: An optional factor by which to scale all mesh node coordinates.

Type: real

Default: 1.0

Valid values: > 0

`rotation_angles`

Description: A list of 3 angles, given in degrees, specifying the amount of counter-clockwise rotation about the x, y, and z-axes to apply to the mesh. The rotations are done sequentially in that order. A negative angle is a clockwise rotation, and a zero angle naturally implies no rotation.

Type: real 3-vector

Default: $(0.0, 0.0, 0.0)$

`partitioner`

Description: The partitioning method used to generate the parallel decomposition of the mesh.

Type: case-insensitive string

Default: `"chaco"`

Valid values: `"chaco"`, `"metis"`, `"file"`, `"block"`

Notes:

"chaco" uses a graph partitioning method from the Chaco library [9] to compute the mesh decomposition at run time. This is the standard method long used by Truchas.

"metis" uses the well-known METIS library [10] to partition the dual graph of the mesh at run time. This method has a number of options which are described below.

"file" reads the partitioning of the mesh cells from a disk file; see `partition_file`.

"block" partitions the mesh cells into nearly equal-sized blocks of consecutively numbered cells according their numbering in the mesh file. The quality of this naive decomposition entirely depends on the given ordering of mesh cells, and thus this option is not generally recommended.

partition_file

Description: Specifies the path to the mesh cell partition file, and is required when `partitioner` is `"file"`. If not an absolute path, it will be interpreted as a path relative to the Truchas input file directory.

Type: case-sensitive string

Default: none

Notes: The format of this text file consists of a sequence of integer values, one value or multiple values per line. The first value is the partition number of the first cell, the second value the partition number of the second cell, and so forth. The number of values must equal the number of mesh cells. The file may use either a 0-based or 1-based numbering convention for the partitions. Popular mesh partitioning tools typically use 0-based partition numbering, and so the default is to assume 0-based numbering; use `first_partition` to specify 1-based numbering.

first_partition

Description: Specifies the number given the first partition in the numbering convention used in the partition file. Either 0-based or 1-based numbering is allowed.

Type: integer

Default: 0

Valid values: 0 or 1

METIS Mesh Partitioning

When `"metis"` is specified for **`partitioner`**, a graph partitioning procedure from the METIS library is used to partition the dual graph of the mesh. This is the graph whose nodes are the mesh cells and edges are the faces shared by cells. The partitioning procedures have the following integer-valued options that may be specified, though all have reasonable defaults so that none must be specified. See the METIS documentation [10] for more details on these options.

metis_ptype specifies the partitioning method. Possible values are:

- 0: Multilevel recursive bisection (default)
- 1: Multilevel k-way partitioning

metis_iptype specifies the algorithm used during initial partitioning (recursive bisection only). Possible values are:

- 0: Grows a bisection using a greedy strategy (default)
- 1: Computes a bisection at random followed by a refinement

`metis_ctype` specifies the matching scheme to be used during coarsening. Possible values are:

0: Random matching
1: Sorted heavy-edge matching (default)

`metis_ncuts` specifies the number of different partitionings that will be computed. The final partitioning will be the one that achieves the best edgecut. The default is 1.

`metis_niter` specifies the number of iterations of the refinement algorithm at each stage of the uncoarsening process. The default is 10.

`metis_ufactor` specifies the maximum allowed load imbalance among the partitions. A value of n indicates that the allowed load imbalance is $(1+n)/1000$. The default is 1 for recursive bisection (i.e., an imbalance of 1.001) and the default value is 30 for k-way partitioning (i.e., an imbalance of 1.03).

`metis_minconn` specifies whether the partitioning procedure should seek to minimize the maximum degree of the subdomain graph (1); or not (0, default). The subdomain graph is the graph in which each partition is a node, and edges connect subdomains with a shared interface.

`metis_contig` specifies whether the partitioning procedure should produce partitions that are contiguous (1); or not (0, default). If the dual graph of the mesh is not connected this option is ignored.

`metis_seed` specifies the seed for the random number generator.

`metis_dbglvl` specifies the amount and type of diagnostic information that will be written to stderr by the partitioning procedure. The default is 0, no output. Use 1 to write some basic information. Refer to the METIS documentation for the many other possible values and the output they generate.

Chapter 18

NUMERICS Namelist

Overview

The NUMERICS namelist specifies general numerical parameters not specific to any particular physics, especially those controlling the overall time stepping of the Truchas model.

NUMERICS Namelist Features

Required/Optional: Required

Single/Multiple Instances: Single

Components

- Alittle
- Cutvof
- Cycle_Max
- Cycle_Number
- Discrete_Ops_Type
- Dt_Constant
- Dt_Grow
- Dt_Init
- Dt_Max
- Dt_Min
- t

Alittle

Description: A small, positive real number (relative to unity) used to avoid division by zero or to compare against other numbers to deduce relative significance.

Physical dimension: dimensionless

Type: real

Default: `EPSILON`(x), where x is of type `real`. If the precision of x is double, `EPSILON`(x) returns 1.0^{-16} for most combinations of software (Fortran 90 compiler) and hardware platforms tested.

Valid values: (0.0, 0.001]

`Cutvof`

Description: The value of a material cell volume fraction below which that material is ignored. If any material has a cell volume fraction less than `Cutvof`, then that material is deleted from that cell, with all other materials present receiving a proportional increase in volume fraction. The only exception is the "background" material, which if present receives the entire allocation (equal to the volume fraction deleted).

Physical dimension: dimensionless

Type: real

Default: 10^{-8}

Valid values: $(0.0, 1.0)$

Notes: Relative to most other volume-fraction-based algorithms, `Cutvof` is defaulted and used at a *much* lower value. If a prototypical value of `Cutvof` equal to 10^{-4} were used, as is the case in most commercial software, local and global mass conservation would suffer, and lack of algorithmic robustness would be masked. The default 10^{-8} value for `Cutvof` yields good results, hence setting it higher is generally not necessary.

`Cycle_Max`

Description: The maximum cycle number allowed in the given simulation, where one "cycle" corresponds to one integration time step over all physical model equations. The simulation will terminate gracefully when the cycle number reaches `Cycle_Max`.

Type: integer

Default: 1000000

Valid values: $(0, \infty)$

Notes: A simulation will also terminate gracefully if the last entry in the `Output_T` input variable (`real`) array (in the `OUTPUTS` namelist) is exceeded by the current simulation time `t`.

`Cycle_Number`

Description: The cycle number to be used just prior to the first actual cycle (time step) taken in the given simulation. The first simulation cycle number is then taken to be `cycle_number` + 1.

Type: integer

Default: 0

Valid values: $[1, \infty)$

Notes: The default value of 0 results in the first computational cycle taken to be 1. This input variable is most useful when starting simulations from a restart file that already contains a restart cycle number > 0.

`Discrete_Ops_Type`

Description: Flag for choosing the numerical reconstruction method used in estimating face-centered (located at cell face centroids) spatial gradients and values of discrete cell-centered data. Specifically, face pressure gradients, face velocity values, and face velocity gradients are all controlled by this flag.

Type: string

Default: `"default"`

Valid values: `"default"`, `"ortho"`, `"nonortho"`

Notes: Face-centered pressure gradients are needed for cell-centered estimates of the pressure Laplacian (used in the pressure Poisson solve) and for enforcement of solenoidal face-centered velocities. Face-centered velocity gradients are needed for cell-centered estimates of the stress tensor, and face-centered velocity values are needed for estimation of fluxing velocities. If this variable does not appear in the `NUMERICS` namelist, or if it appears and is set to `'default'`, the type of discrete operators to use is chosen by testing the orthogonality of the mesh cells. When *all* mesh cells are found to be orthogonal (to roundoff error) and thus hexahedral, then `Discrete_Ops_Type` defaults to an `'ortho'` method. Otherwise, `Discrete_Ops_Type` defaults to a `'nonortho'` method, namely the Least Squares Linear Reconstruction (LSLR) method. For a detailed discussion of discrete operators in Truchas, consult the *Truchas Physics and Algorithms*. If `'ortho'` or `'nonortho'` is set, the chosen operator type is used, whatever the mesh.

The user can set an overall discrete operator type as above and choose an alternative discrete operator type for the projection step in fluid flow (FF). To override the overall default or explicit overall setting, set the variable `FF_Discrete_Ops_Type` to the desired value. Setting this variable leaves the overall setting `Discrete_Ops_Type` unmodified for all remaining parts of the code.

Dt_Constant

Description: A *constant* integration time step value to be used for all time steps in the simulation

Physical dimension: T

Type: real

Default: none

Valid values: $(0, \infty)$

Notes: This `Dt_Constant` input variable should be used with *extreme caution*, as its specification overrules all other time step choices that are controlled by linear stability and accuracy considerations. In particular, `NUMERICS` namelist input variables `Dt_Grow`, `Dt_Init`, `Dt_Max`, and `Dt_Min` are all ignored if `Dt_Constant` is specified. Use of `Dt_Constant` is therefore advised only for controlled numerical algorithm experiments, and not for simulations intended for applications analysis and validation.

Dt_Grow

Description: A factor to multiply the current integration time step (δt) for the purpose of estimating the time step used during the next cycle ($\delta t_g = \texttt{Dt\_Grow} * \delta t$). This candidate time step (δt_g) is chosen only if all other currently active time step restrictions have not already limited its value below δt_g.

Physical dimension: dimensionless

Type: real

Default: 1.05

Valid values: $[1, \infty)$

Notes: `Dt_Grow` is *ignored* if `Dt_Constant` is specified.

Dt_Init

Description: Integration time step value used for the first computational cycle

Physical dimension: T

Type: real

Default: 10^{-6}

Valid values: $(0, \infty)$

Notes: The default value of `Dt_Init` is completely arbitrary, as it is *always* problem dependent. Unless a constant time step is desired (via specification of `Dt_Constant`), then, a value for `Dt_Init` should be specified (instead of relying on the default) that is consistent with the time scales of interest for the simulation at hand. A general rule of thumb is to set `Dt_Init` smaller than the time step ultimately desired, thereby allowing the time step to grow gradually and steadily (say, over 10-20 cycles) to the desired time step value.

For restart calculations, the initial time step size is extracted from the restart file and used instead of `Dt_Init`, unless `Ignore_Dt` in the `RESTART` namelist has been set to true.

`Dt_Init` is *ignored* if `Dt_Constant` is specified.

Dt_Max

Description: Maximum allowable value for the time step

Physical dimension: T

Type: real

Default: 10

Valid values: $(0, \infty)$

Notes: The time step is *not* allowed to exceed this value, even if other accuracy- or stability-based criteria would permit it. The default value of `Dt_Max` is completely arbitrary, as it is *always* problem dependent. Unless a constant time step is desired (via specification of `Dt_Constant`), then, a value for `Dt_Max` should be specified (instead of relying on the default) that is consistent with the time scales of interest for the simulation at hand.

`Dt_Max` is *ignored* if `Dt_Constant` is specified.

Dt_Min

Description: Minimum allowable value for the time step

Physical dimension: T

Type: real

Default: 10^{-6}

Valid values: $(0, \infty)$

Notes: If the time step falls below this value, the user is informed as such and the simulation terminates gracefully. Termination occurs at this condition because is it very probable that either numerical algorithm or physical model problems have occurred and the simulation is unable to recover. The default value of `Dt_Min` is completely arbitrary, as it is *always* problem dependent. Unless a constant time step is desired (via specification of `Dt_Constant`), then, a value for `Dt_Min` should be specified (instead of relying on the default) that is consistent with the time scales of interest for the simulation at hand. A good rule of thumb to use for `Dt_Min` is to use a value several orders of magnitude below the time scale of interest.

`Dt_Min` is *ignored* if `Dt_Constant` is specified.

t

Description: The simulation time to be used at the beginning of the first computational cycle.

Physical dimension: T

Type: real

Default: 0

Chapter 19

`OUTPUTS` Namelist

Overview

The `OUTPUTS` namelist defines the problem end time and various output options.

`OUTPUTS` Namelist Features

Required/Optional: Required
Single/Multiple Instances: Single

Components

- `Int_Output_Dt_Multiplier`
- `Output_Dt`
- `Output_Dt_Multiplier`
- `Output_T`
- `Probe_Output_Cycle_Multiplier`
- `Short_Output_Dt_Multiplier`
- `Move_Block_IDs`
- `Move_Toolpath_Name`

`Int_Output_Dt_Multiplier`

Description: Factor multiplying `Output_Dt` for time interval to write interface output data.
Type: integer array
Default: none
Valid values: ≥ 0

`Output_Dt`

Description: Output time interval for each output time span.
Physical dimension: T
Type: real array
Default: none
Valid values: > 0

Output_T

Description: A sequence of time values for defining time spans that have distinct output time intervals. The last time is the problem end time.

Physical dimension: T

Type: real array

Default: none

Valid values: strictly increasing sequence of two or more values

Probe_Output_Cycle_Multiplier

Description: Factor multiplying truchas cycle to determine frequency of writing probe output.

Type: integer

Default: 1

Valid values: > 0

Short_Output_Dt_Multiplier

Description: Factor multiplying `Output_Dt` for time interval to write short edits

Type: integer array

Default: 0

Valid values: ≥ 0

Output_Dt_Multiplier

Description: Factor multiplying `Output_Dt` for time interval to write output.

Type: integer array

Default: 0

Valid values: ≥ 0

Move_Block_IDs

Description: A list of element block IDs that are associated with a translation written to the output file. Use `Move_Toolpath_Name` to specify the translation.

Type: a list of up to 32 integers

Default: none

Notes: Use of this feature does not alter the mesh data that is written to the HDF5 output file. It merely adds some additional data that associates a time-dependent translation with element blocks. Use of the data, if any, is left to users of the file. At this time the Paraview Truchas output reader (post version 5.2) uses this information to translate the mesh blocks for visualization.

Move_Toolpath_Name

Description: The name of a `TOOLPATH` namelist that defines the translation to apply to the element blocks given by `Move_Block_IDs`.

Type: string

Default: none

Chapter 20

PHASE_CHANGE Namelist

Overview

The PHASE_CHANGE namelist defines the phase change model used for the transformation between two phases of a multi-phase material. An instance of this namelist is required for each consecutive pair of phases specified by the phases variable of a MATERIAL namelist.

Truchas models the transformation from low temperature phase ("solid") to high temperature phase ("liquid") using a "solid fraction" function $f_{\text{sol}}(T)$ that gives the fraction of low temperature phase as a function of temperature. There are two temperatures $T_{\text{sol}} < T_{\text{liq}}$ such that $f_{\text{sol}} = 1$ for $T \le T_{\text{sol}}$, $f_{\text{sol}} = 0$ for $T \ge T_{\text{liq}}$, and $0 < f_{\text{sol}} < 1$ for $T_{\text{sol}} < T < T_{\text{liq}}$. There are two alternative methods for specifying this function.

The first simple, but non-physical, method uses a smooth Hermite cubic polynomial to interpolate f_{sol} between T_{sol} and T_{liq}. This is pictured below. The second method uses a table of (T, f_{sol}) values to define $f_{\text{sol}}(T)$. This allows physics-based phase change models like lever and Scheil to be defined, and data from CALPHAD-type tools to be used directly.

Note that while described in terms of solid and liquid, these phase change models also apply to solid-solid transformations with the obvious reinterpretation of notation.

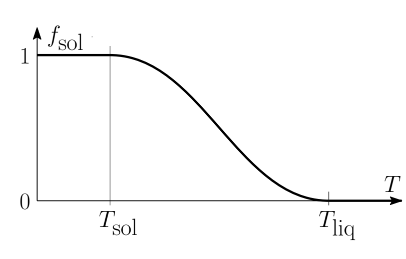
Namelist Variables

low_temp_phase, **high_temp_phase**

These are the names of the two material phases.

solidus_temp, **liquidus_temp**

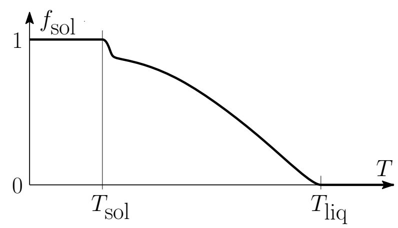
The solidus and liquidus temperatures, T_{sol} and T_{liq}. If these variables are defined, the simple solid fraction model is used, which interpolates the solid fraction over the interval $[T_{\text{sol}}, T_{\text{liq}}]$ using a smooth Hermite cubic polynomial. These variables are incompatible with **solid_frac_table**.



solid_frac_table

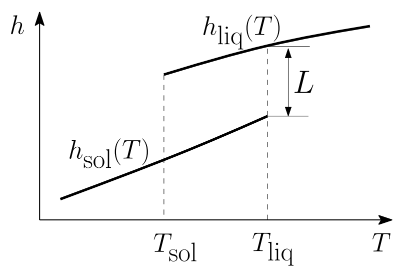
A table of temperature-solid fraction values. The format is the same as described for the FUNCTION namelist Tabular_Data variable. However the table may be given in either increasing or decreasing temperature order, and the solid fraction values must be strictly monotone. The table endpoints must specify the 1 and 0 solid fractions, and the corresponding temperatures are taken to be T_{sol} and T_{liq}, respectively. The solid fraction function is interpolated from this table using Akima smoothing; see the FUNCTION namelist Tabular_Interp. Additional data points outside the interval $[T_{\mathrm{sol}}, T_{\mathrm{liq}}]$ are automatically added to ensure that the solid fraction is constant outside the transformation interval. This variable is incompatible with **solidus_temp** and **liquidus_temp**.

```
solid_frac_table = 930.0 1.00
                   931.0 0.95
                   ...
                   940.0 0.00
```



latent_heat

The energy per unit mass L absorbed during the transformation of the material from the low temperature phase to the high temperature phase at the liquidus temperature T_{liq}. This property is required when the specific enthalpy of the high temperature phase is defined from its specific heat. Otherwise it is ignored.



Chapter 21

PHYSICAL_CONSTANTS Namelist

Overview

The values of physical constants used in Truchas' physics models are set through this namelist. The default for all these constants is their value in SI units. If a different system of units is used, these may need to be assigned the appropriate values.

PHYSICAL_CONSTANTS Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Single

Components

- Absolute_Zero
- Stefan_Boltzmann
- Vacuum_Permeability
- Vacuum_Permittivity

Absolute_Zero

Description: The value of absolute-zero in the temperature scale used. The default value is 0 for Kelvin. Centigrade would use the value -273.15, for example. This constant is used by thermal radiation boundary conditions and enclosure (view factor) radiation.

Physical dimension: Θ

Type: `real`

Default: 0 K

Valid values: any value

Stefan_Boltzmann

Description: Stefan-Boltzmann constant for thermal radiation. The default is its value in SI units. This constant is used by thermal radiation boundary condititions and enclosure (view factor) radiation.

Physical dimension: $\mathsf{E}/(\mathsf{T}\ \mathsf{L}^2\ \Theta^4)$

Type: `real`

Default: 5.67×10^{-8} W/(m^2 K^4)

Valid values: any positive value

Vacuum_Permeability

Description: The magnetic permeability of free space. The default is its value in SI units. This parameter is used by the electromagnetics solver.

Physical dimension: $\mathsf{M}\ \mathsf{L}\ \mathsf{T}^{-2}\ \mathsf{I}^{-2}$

Type: `real`

Default: $4\pi \times 10^{-7}$ H/m

Valid values: any positive value

Vacuum_Permittivity

Description: The electric permittivity of free space. The default is its value in SI units. This parameter is used by the electromagnetics solver.

Physical dimension: $\mathsf{M}^{-1}\ \mathsf{L}^{-3}\ \mathsf{T}^{4}\ \mathsf{I}^{2}$

Type: `real`

Default: 8.854188×10^{-12} F/m

Valid values: any positive value

Chapter 22

`PHYSICS` Namelist

Overview

The `PHYSICS` namelist specifies which physics models are active in the simulation. The models are implemented by the four primary physics kernels — fluid flow, heat/species transport, induction heating, and solid mechanics — which are weakly coupled using time splitting. A brief overview of the physics kernels follows; see *Truchas Physics and Algorithms* for more details.

Fluid Flow. The fluid flow physics model simulates multi-material, incompressible flow with interface tracking. A gravitational body force is defined using the `Body_Force_Density` variable. See the `MATERIAL` namelist for a description of the material properties required by the fluid flow model.

Heat and Species Transport. The heat and species transport physics kernel models both heat conduction with thermal (view factor) radiation, and solutal species diffusion and thermodiffusion. These (primarily) diffusive transport processes are fully coupled; advection of enthalpy and solutal species are handled by the fluid flow physics kernel and incorporated as loosely-coupled source terms.
Heat transport is enabled using the `Heat_Transport` flag, and solves the heat equation

$$\frac{\partial H}{\partial t} = \nabla \cdot K \nabla T + Q + Q_{\text{joule}} + Q_{\text{adv}}, \tag{22.1}$$

with dependent variables temperature T and enthalpy density H. The enthalpy density is algebraically related to temperature as $H = f(T)$ where $f'(T) = \rho\, c_p$ is the volumetric heat capacity. See the `MATERIAL` namelist for a description of the material properties required by the heat equation. The optional volumetric heat source Q is defined through the `DS_SOURCE` namelist using `"temperature"` as the equation name. The Joule heating source Q_{joule} is computed by the induction heating kernel, and the advected heat Q_{adv} by the flow kernel. The boundary conditions on T are defined through the `THERMAL_BC` namelists. The initial value of T are defined through the `Temperature` variable of the `BODY` namelists. View factor radiation systems which couple to the heat equation are defined using `ENCLOSURE_RADIATION` namelists.
Solutal species transport is enabled using the `Species_Transport` flag, which solves the n coupled equations

$$\frac{\partial \phi_i}{\partial t} = \nabla \cdot D_i (\nabla \phi_i\, [+ S_i \nabla T]) + Q_i + Q_{i,\text{adv}}, \quad i = 1, \ldots, n, \tag{22.2}$$

for species concentrations ϕ_i. The number of components n is defined by `Number_of_Species`. The thermodiffusion term in $[\,\cdot\,]$ is only included when coupled with heat transport. See the `MATERIAL` namelist for defining the diffusivities D_i and Soret coefficients S_i. The optional volumetric source Q_i is defined through the `DS_SOURCE` namelist using `"concentration`*i*`"` as the equation name. The advected species source $Q_{i,\text{adv}}$ is computed by the flow kernel. Boundary conditions on ϕ_i are defined through the `SPECIES_BC` namelists. The initial value of the ϕ_i are defined through the `Phi` variable of the `BODY` namelists.

Induction Heating. The induction heating physics kernel solves for the Joule heat that is used as a source in heat transport. It is enabled using the `Electromagnetics` flag. See the `MATERIAL` namelist for a description of the material properties required by the electromagnetics solver. The `ELECTROMAGNETICS` namelist is used to describe the induction heating problem.

Solid Mechanics. The solid mechanics physics kernel models small strain elastic and plastic deformation of solid material phases, including deformations induced by temperature changes and solid state phase changes. It is enabled using the `Solid_Mechanics` flag. See the `MATERIAL` namelist for a description of the material properties required by the solid mechanics kernel. Parameters which define the plasticity model are defined using the `VISCOPLASTIC_MODEL` namelist. Displacement and traction boundary conditions are defined using the `BC` namelist. The effect of the gravitational body force defined by `Body_Force_Density` can be included by enabling the `Solid_Mechanics_Body_Force` flag.

`PHYSICS` Namelist Features

Required/Optional: Required

Single/Multiple Instances: Single

Components

- `Body_Force_Density`
- `Electromagnetics`
- `Flow`
- `Heat_Transport`
- `Materials`
- `Number_of_Species`
- `Solid_Mechanics`
- `Species_Transport`

`Materials`

Description: A list of materials to include in the simulation. These are material names defined in `MATERIAL` namelists. The list must include all materials assigned to a region in a `BODY` namelist, or specified as an `inflow_material` in a fluid flow boundary condition, but it need not include all materials defined in the input file. Use the reserved name `"VOID"` to refer to the built-in void pseudo-material.

Type: string list

`Flow`

Description: Enables the simulation of fluid flow.

Type: logical

Default: false

Body_Force_Density

Description: A constant force per unit mass, **g**, that acts throughout material volumes. The net force on a volume is the integral of its density times **g** over the volume. Typically **g** is the gravitational acceleration.

Physical dimension: L/T^2

Type: real 3-vector

Default: (0, 0, 0)

Note: The fluid flow model always includes this body force.

The solid mechanics model has the option of including this body force or not; see `Solid_Mechanics_Body_Force`.

Heat_Transport

Description: Enables the calculation of heat conduction, advection, and radiation using the heat/species transport physics kernel.

Type: logical

Default: false

Species_Transport

Description: Enables the calculation of species diffusion and advection using the heat/species transport physics kernel. The number of species components must be specified using `Number_of_Species`.

Type: logical

Default: false

Number_of_Species

Description: The number of species components. Required when `Species_Transport` is enabled.

Type: integer

Default: 0

Valid values: > 0

Solid_Mechanics

Description: Enables the calculation of solid material stresses and strains.

Type: logical

Default: false

Electromagnetics

Description: Enables the calculation of Joule heating.

Type: logical

Default: false

Chapter 23

PROBE Namelist

Overview

The PROBE namelist is used to define the location in the computational domain where the value of specific solution quantities will be recorded at every time step. The data is written to a standard multi-column text file specified by the namelist. The solution time is written to the first column and the specified solution quantities to the remaining columns. Useful metadata about the probe is written to the first few lines of the file. These lines begin with a # character and would be treated as comment lines by many post-processors. As many probes as desired may be defined, but each must write to a different file.

Probe Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Multiple

Components

- `coord`
- `coord_scale_factor`
- `data`
- `data_file`
- `description`
- `digits`

coord

Description: The spatial coordinates of the location of the probe. These coordinates may be further scaled by an optional scaling factor; see `coord_scale_factor`.

Type: real 3-vector

Default: none

Notes: For cell-centered quantities, data will be taken from the cell whose centroid is nearest this location, and for node-centered quantities data will be taken from the nearest node.

coord_scale_factor

Description: A multiplicative scaling factor applied to the coordinates of the probe.

Type: real

Default: 1.0

data

Description: The data quantity whose value will be recorded. The available options are:

- **"temperature"** Cell-centered temperature
- **"pressure"** Cell-centered fluid pressure
- **"velocity"** Cell-centered fluid velocity

Type: string

Default: none

data_file

Description: The name of the probe output file. The file will be created in the output directory, and any existing file will be overwritten. Each probe must write to a different output file.

Type: string

Default: none

description

Description: An optional text string that will be written to the header of the output file.

Type: string

Default: none

digits

Description: The number of significant digits in the output data.

Type: integer

Default: 6

Chapter 24

RESTART Namelist

Overview

Truchas is able to use data from a previous calculation to initialize a new calculation. Such a *restart* calculation is invoked by using the '-r' commandline argument to the executable with the path name of the restart data file. By default, all appropriate data from the restart file is used. This optional namelist provides variables to limit the restart data that will be used.

RESTART Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Single

Components

- Ignore_T
- Ignore_Dt
- Ignore_Joule_Heat
- Ignore_Solid_Mechanics

Ignore_T

Description: When restarting, the initial time and starting cycle count are normally extracted from the restart file. If this flag is true, then those values are ignored and the first value of the Output_T array is used as the initial time and the cycle count starts at 0, as happens with a non-restart run.

Type: logical

Default: false

Ignore_Dt

Description: When restarting, the initial time step size is normally extracted from the restart file. If this flag is true, then that value is ignored and the value specified by Dt_Init from the NUMERICS namelist is used instead. Note that if Dt_Constant in the NUMERICS namelist is specified then its value is used regardless.

Type: logical

Default: false

Ignore_Joule_Heat

Description: If this flag is true, the Joule heat data in the restart file (if any) will be ignored when initializing the code. This variable is only relevant for restart calculations with `Electromagnetics` enabled in the `PHYSICS` namelist.

Type: logical

Default: false

Ignore_Solid_Mechanics

Description: If this flag is true, the solid mechanics data in the restart file (if any) will be ingored when initializing the code. This variable is only relevant for restart calculations with `Solid_Mechanics` enabled in the `PHYSICS` namelist.

Type: logical

Default: false

Chapter 25

SIMULATION_CONTROL Namelist (Experimental)

Overview

There may be points in time during a simulation when something changes abruptly; a boundary condition or source turns on/off, or a physics model is enabled/disabled, for example. In such circumstances it is best to hit these times precisely with a time step and then continue from that point with a reduced step size appropriate to resolving the time transients that result from the impulsive forcing of the model—in essence, to split the simulation seamlessly into a sequence of phases where each phase is a new simulation whose initial state is the final state of the preceding phase. This experimental namelist provides a means for achieving this. The start time of each additional phase subsequent to the initial phase is specified using the `Phase_Start_Times` array, and the initial step size using either the `Phase_Init_Dt` or `Phase_Init_Dt_Factor` variables. Truchas will hit those times precisely with a time step, smoothly adjusting the step size in advance to avoid abrupt step size changes, and then effectively "restart" the time stepping. Currently this only effects the second-order diffusion solver which maintains a (smooth) history of states at recent time steps. When restarting that history is deleted and time stepping begins fresh using only the current state.

SIMULATION_CONTROL Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Single

Components

- `Event_Lookahead`
- `Phase_Init_Dt`
- `Phase_Init_Dt_Factor`
- `Phase_Start_Times`

Event_Lookahead

Description: When approaching an event time, such as an output time, the time step sizes are gradually adjusted to hit the time precisely. This variable specifies the number of steps over which this occurs.

Type: integer

Default: 5

Valid values: ≥ 2

Note: A value of 1 would mean no step size adjustment until a step would go beyond the event time. This can result in needing to take an arbitrarily small time step and one smaller than the minimum allowed, and thus this is not allowed. The greater the lookahead, the more gradually the time step will be reduced.

Phase_Init_Dt

Description: The initial time step size to use for each of the simulation phases. This is the analog of `Dt_Init`, which is used for the initial (and default) phase of the simulation. Either `Phase_Init_Dt` or `Phase_Init_Dt_Factor` must be specified, but not both.

Type: real

Default: none

Phase_Init_Dt_Factor

Description: The initial time step size used for each of the simulation phases is this factor times the last step size of the preceding phase. Either `Phase_Init_Dt_Factor` or `Phase_Init_Dt` must be specified, but not both.

Type: real

Default: none

Phase_Start_Times

Description: The list of starting times of each of the phases.

Type: real array

Default: none

Note: The initial simulation phase, which is otherwise the only phase, need not be included in this list, though it may be. The provided list of times is sorted, and the first time greater than the initial time is taken as the start of the first phase following the default initial phase; earlier times are ignored.

Chapter 26

SOLID_MECHANICS Namelist

Overview

The SOLID_MECHANICS namelist sets parameters that are specific to the solid mechanics model and algorithm. This namelist is read whenever the PHYSICS namelist option Solid_Mechanics is enabled. Parameters for the nonlinear solver used by the algorithm and its preconditioner are specified in NONLINEAR_SOLVER and LINEAR_SOLVER namelists. Optional material viscoplasticity models are defined in VISCOPLASTIC_MODEL namelists.

SOLID_MECHANICS Namelist Features

Required/Optional Required when solid mechanics physics is enabled.

Single/Multiple Instances Single

Components

- Contact_Distance
- Contact_Norm_Trac
- Contact_Penalty
- Displacement_Nonlinear_Solution
- Solid_Mechanics_Body_Force
- Stress_Reduced_Integration
- Strain_Limit
- Convergence_Criterion
- Maximum_Iterations
- NLK_Vector_Tolerance
- NLK_Max_Vectors

Contact_Distance

Description: A length scale parameter β for the contact function

$$\lambda = \lambda_s * \lambda_\tau$$

where

$$\lambda_s = \begin{cases} 1 & \text{if } s < 0 \\ 0 & \text{if } s > \beta \\ 2(\frac{s}{\beta} - 1)^3 + 3(\frac{s}{\beta} - 1)^2 & \text{if } 0 < s < \beta \end{cases}$$

and

$$\lambda_\tau = \begin{cases} 1 & \text{if } \tau_n < 0 \\ 0 & \text{if } \tau_n > \tau^* \\ 2(\frac{\tau_n}{\tau^*} - 1)^3 + 3(\frac{\tau_n}{\tau^*} - 1)^2 & \text{if } 0 < \tau_n < \tau^* \end{cases}$$

$s = \hat{n} \cdot (u_k - u_j)$

Physical dimension: L

Type: real

Default: 1.0e-7

Valid values: $(0, \infty]$

Notes: The default value is usually a good value for mesh cell sizes in the 1 - 10 mm size range.

Contact_Norm_Trac

Description: A parameter τ^* for the contact function

$$\lambda = \lambda_s * \lambda_\tau$$

where

$$\lambda_s = \begin{cases} 1 & \text{if } s < 0 \\ 0 & \text{if } s > \beta \\ 2(\frac{s}{\beta} - 1)^3 + 3(\frac{s}{\beta} - 1)^2 & \text{if } 0 < s < \beta \end{cases}$$

and

$$\lambda_\tau = \begin{cases} 1 & \text{if } \tau_n < 0 \\ 0 & \text{if } \tau_n > \tau^* \\ 2(\frac{\tau_n}{\tau^*} - 1)^3 + 3(\frac{\tau_n}{\tau^*} - 1)^2 & \text{if } 0 < \tau_n < \tau^* \end{cases}$$

τ_n is the normal traction at the interface where a positive value corresponds to a tensile force normal to the surface.

Physical dimension: $\mathsf{F/L^2}$

Type: real

Default: 1.0e4

Valid values: $[0, \infty]$

Notes: The default value is probably appropriate for materials with elastic constants in the range 10^9 - 10^{11}. This parameter should probably be scaled proportionately for elastic constants that differ from this range.

Contact_Penalty

Description: A penalty factor for the penetration constraint in the contact algorithm. Changing this is probably not a good idea in the current version.

Physical dimension: dimensionless

Type: real

Default: 1.0e3

Valid values: $[0, \infty]$

Notes:

Displacement_Nonlinear_Solution

Description: A character string pointer to the nonlinear solution algorithm parameters to be used in a Newton-Krylov solution of the nonlinear thermo-elastic viscoplastic equations. This string "points" to a particular `NONLINEAR_SOLVER` namelist if it matches the `Name` input variable string in the `NONLINEAR_SOLVER` namelist.

Type: string

Default: `"default"`

Valid values: arbitrary string

Notes: If this string does not match a `Name` input variable string specified in a `NONLINEAR_SOLVER` namelist, then the default set of nonlinear solution algorithm parameters is used for the thermo-elastic viscoplastic equations.

Solid_Mechanics_Body_Force

Description: Body forces will be included in the solid mechanics calculation.

Physical dimension:

Type: `logical`

Default: `.false.`

Strain_Limit

Description: This parameter controls the use of the ODE integrator in the plastic strain calculation. It should be set to the minimum significant value of the plastic strain increment for a time step. If convergence seems poor when a viscoplastic material model is used, it may help to reduce the value.

Physical dimension: `L/L`

Type: real

Default: 1.0e-10

Valid values: ≥ 0

Notes: This parameter can not be currently used to control the time step. It may be used for such purposes in future releases.

Convergence_Criterion

Description: Tolerance used to determine when nonlinear convergence has been reached.

Type: real

Default: 1e-12

Valid values: $(0, 0.1)$

Note: We refer to the input value of `Convergence_Criterion` as ϵ. $F(x) = 0$ is the nonlinear system being solved.

The nonlinear iteration is stopped when *either* of the following two conditions are met:

- reduction in the 2-norm of the nonlinear residual meets the criterion, i.e.:

$$\frac{\|F(x_{k+1})\|_2}{\|F(x_0)\|_2} < \gamma$$

- the relative change in the max-norm of the solution meets the criterion, i.e.:

$$\frac{\|\delta x\|_\infty}{\|x_{k+1}\|_\infty} < \gamma$$

where γ is the input desired tolerance, modified using an estimate of the convergence rate, i.e.:

$$\gamma = (1 - \rho)\epsilon$$

and:

$$\rho = \frac{\|x_{k+1} - x_k\|_\infty / \|x_{k+1}\|_\infty}{\|x_k - x_{k-1}\|_\infty / \|x_k\|_\infty}$$

This is an attempt to prevent false convergence if the solution stagnates, but allow iteration to stop if the solution is acceptable.

`Maximum_Iterations`

Description: Maximum allowed number of iterations of the nonlinear solver.

Type: `integer`

Default: 100

Valid values: $[0, \infty)$

`NLK_Vector_Tolerance`

Description: The vector drop tolerance for the NLK method. When assembling the acceleration subspace vector by vector, a vector is dropped when the sine of the angle between the vector and the subspace less than this value.

Type: `real`

Default: 0.01

Valid values: $(0, 1)$

`NLK_Max_Vectors`

Description: For the NLK method, the maximum number of acceleration vectors to be used.

Type: `integer`

Default: 20

Valid values: $[0, \infty)$

Chapter 27

SPECIES_BC Namelist

Overview

The SPECIES_BC namelist is used to define boundary conditions for the species diffusion model at external boundaries. Each instance of the namelist defines a particular condition to impose on a species component over a subset of the domain boundary. The boundary subset Γ is specified using mesh face sets. The namelist variable face_set_ids takes a list of face set IDs, and the boundary condition is imposed on all faces belonging to those face sets. Note that ExodusII mesh side sets are imported into Truchas as face sets with the same IDs. The species component is specified using the comp namelist variable.
The following types of boundary conditions can be defined. The outward unit normal to the boundary Γ is denoted $\hat{n}$.

- *Concentration.* A concentration Dirichlet condition for species component j

  $$\phi_j = c \text{ on } \Gamma \tag{27.1}$$

  is defined by setting type to "concentration". The boundary value c is specified using either conc for a constant value, or conc_func for a function.
- *Flux.* A total concentration flux condition for species component j

  $$-D_j \nabla\phi_j \cdot \hat{n} = q \text{ on } \Gamma, \tag{27.2}$$

  when the system does not include temperature as a dependent variable, or

  $$-D_j(\nabla\phi_j + S_j \nabla T)\cdot\hat{n} = q \text{ on } \Gamma \tag{27.3}$$

  when it does, is defined by setting type to "flux". The concentration flux q is specified using either flux for a constant value, or flux_func for a function.

The specified species concentration boundary conditions are not allowed to overlap, and they must completely cover the computational boundary.

SPECIES_BC Namelist Features

Required/Optional: Required
Single/Multiple Instances: Multiple

Components

- name

- face_set_ids
- comp
- type
- conc
- conc_func
- flux
- flux_func

name

Description: A unique name used to identify a particular instance of this namelist.

Type: string (31 characters max)

Default: none

comp

Description: The species component this boundary condition applies to.

Type: integer

Default: 1

face_set_ids

Description: A list of face set IDs that define the portion of the boundary where the boundary condition will be imposed.

Type: integer list (32 max)

Default: none

type

Description: The type of boundary condition. The available options are:

"concentration" Concentration is prescribed on the boundary. Use conc or conc_func to specify its value.

"flux" Total outward concentration flux is prescribed on the boundary. Use flux or flux_func to specify its value.

Type: string

Default: none

conc

Description: The constant value of boundary concentration for a concentration-type boundary condition. To specify a function, use conc_func instead.

Default: none

Type: real

conc_func

Description: The name of a FUNCTION namelist defining a function that gives the boundary concentration for a concentration-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

flux

Description: The constant value of the total outward boundary concentration flux for a flux-type boundary condition. To specify a function, use flux_func instead.

Default: none

Type: real

flux_func

Description: The name of a FUNCTION namelist defining a function that gives the total outward boundary concentration flux for a flux-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

Chapter 28

THERMAL_BC Namelist

Overview

The THERMAL_BC namelist is used to define boundary conditions for the heat transfer model at external boundaries and internal interfaces. Each instance of the namelist defines a particular condition to impose over a subset of the domain boundary. The boundary subset Γ is specified using mesh face sets. The namelist variable face_set_ids takes a list of face set IDs, and the boundary condition is imposed on all faces belonging to those face sets. Note that ExodusII mesh side sets are imported into Truchas as face sets with the same IDs.

External boundaries

The following types of external boundary conditions can be defined. The outward unit normal to the boundary Γ is denoted $\hat{n}$.

- *Temperature.* A temperature Dirichlet condition

  $$T = T_b \text{ on } \Gamma \tag{28.1}$$

  is defined by setting type to "temperature". The boundary value T_b is specified using either temp for a constant value, or temp_func for a function.

- *Total Flux.* A heat flux condition

  $$-\kappa\nabla T \cdot \hat{n} = q_b \text{ on } \Gamma \tag{28.2}$$

  is defined by setting type to "flux". The heat flux q_b is specified using either flux for a constant value, or flux_func for a function.

- *Heat Transfer.* An external heat transfer flux condition

  $$-\kappa\nabla T \cdot \hat{n} = \alpha(T - T_\infty) \text{ on } \Gamma \tag{28.3}$$

  is defined by setting type to "htc". The heat transfer coefficient α is specified using either htc for a constant value, or htc_func for a function, and the ambient temperature T_∞ is specified using either ambient_temp for a constant value, or ambient_temp_func for a function.

- *Ambient Radiation.* A simple ambient thermal radiation condition

  $$-\kappa\nabla T \cdot \hat{n} = \epsilon\sigma\big((T - T_0)^4 - (T_\infty - T_0)^4\big) \text{ on } \Gamma \tag{28.4}$$

  is defined by setting type to "radiation". The emissivity ϵ is specified using either emissivity for a constant value or emissivity_func for a function, and the temperature of the ambient environment T_∞ is specified using either ambient_temp for a constant value, or ambient_temp_func for a function. Here σ is the Stefan-Boltzmann constant and T_0 is the absolute-zero temperature, both of which can be redefined if the problem units differ from the default SI units using the Stefan_Boltzmann and Absolute_Zero components of the PHYSICAL_CONSTANTS namelist.

The specified boundary conditions are not generally allowed to overlap. It is not permitted, for example, to imposed both a temperature and a flux condition on the same part of boundary. The one exception is that heat transfer and ambient radiation conditions can be superimposed; the net flux in this case will be the sum of the heat transfer and radiation fluxes.

It is also generally required that the specified boundary conditions completely cover the computational boundary. However, when enclosure radiation systems are present no boundary condition need be imposed on any part of the boundary that belongs to an enclosure. Either temperature or heat transfer conditions may still be imposed there, and in the latter case the net heat flux is the sum of the heat transfer and radiative (from enclosure radiation) fluxes.

Internal interfaces

Internal interfaces are merely coincident pairs of conforming external mesh boundaries. These are modifications to the mesh created by Truchas and are defined using the `Interface_Side_Sets` parameter from the `MESH` namelist. Only the face set IDs referenced there can be used in the definition of the following interface conditions. The following types of internal interface conditions can be defined.

- *Interface Heat Transfer.* An interface heat transfer condition models heat transfer across an imperfect contact between two bodies or across a thin subscale material layer lying along an interface Γ. It imposes continuity of the heat flux $-\kappa\nabla T\cdot\hat{n}$ across the interface Γ and gives this flux as

  $$-\kappa\nabla T\cdot\hat{n} = -\alpha[T] \text{ on } \Gamma, \tag{28.5}$$

  where $[T]$ is the jump in T across Γ in the direction $\hat{n}$. It is defined by setting `type` to `"interface-htc"`. The heat transfer coefficient α is specified using either `htc` for a constant value, or `htc_func` for a function.
- *Gap Radiation.* A gap radiation condition models radiative heat transfer across a thin open gap lying along an interface Γ. It imposes continuity of the heat flux $-\kappa\nabla T\cdot\hat{n}$ across Γ and gives the flux as

  $$-\kappa\nabla T\cdot\hat{n} = \epsilon_\Gamma\sigma\big((T_- - T_0)^4 - (T_+ - T_0)^4\big) \text{ on } \Gamma, \tag{28.6}$$

  where T_- and T_+ denote the values of T on the inside and outside gap surfaces with respect to the normal $\hat{n}$ to Γ. It is defined by setting `type` to `"gap-radiation"`. The gap emissivity ϵ_Γ is specified using either `emissivity` for a constant value, or `emissivity_func` for a function. The effective gap emissivity ϵ_Γ depends on the emissivities ϵ_- and ϵ_+ of the surfaces on either side of the gap and is given by

  $$\epsilon_\Gamma = \frac{\epsilon_-\epsilon_+}{\epsilon_- + \epsilon_+ - \epsilon_-\epsilon_+}. \tag{28.7}$$

  The value of the Stefan-Boltzmann constant σ and the absolute-zero temperature T_0 can be redefined if the problem units differ from the default SI units using the `Stefan_Boltzmann` and `Absolute_Zero` components of the `PHYSICAL_CONSTANTS` namelist.

`THERMAL_BC` Namelist Features

Required/Optional: Required

Single/Multiple Instances: Multiple

Components

- `name`
- `face_set_ids`
- `type`

- `temp`
- `temp_func`
- `flux`
- `flux_func`
- `htc`
- `htc_func`
- `ambient_temp`
- `ambient_temp_func`
- `emissivity`
- `emissivity_func`

`name`

Description: A unique name used to identify a particular instance of this namelist.
Type: string (31 characters max)
Default: none

`face_set_ids`

Description: A list of face set IDs that define the portion of the boundary where the boundary condition will be imposed.
Type: integer list (32 max)
Default: none

`type`

Description: The type of boundary condition. The available options are:

`"temperature"` Temperature is prescribed on the boundary. Use `temp` or `temp_func` to specify its value.

`"flux"` Outward heat flux is prescribed on the boundary. Use `flux` or `flux_func` to set its value.

`"htc"` External heat transfer condition. Use `htc` or `htc_func` to set the heat transfer coefficient, and `ambient_temp` or `ambient_temp_func` to set the ambient temperature.

`"radiation"` A simple ambient thermal radiation condition. Use `emissivity` or `emissivity_func` to set the emissivity, and `ambient_temp` or `ambient_temp_func` to set the temperature of the ambient environment.

`"interface-htc"` An internal interface heat transfer condition. Use `htc` or `htc_func` to set the heat transfer coefficient.

`"gap-radiation"` A gap thermal radiation condition. Use `emissivity` or `emissivity_func` to set the emissivity.

Type: string
Default: none

`temp`

Description: The constant value of boundary temperature for a temperature-type boundary condition. To specify a function, use `temp_func` instead.
Default: none
Type: real

temp_func

Description: The name of a FUNCTION namelist defining a function that gives the boundary temperature for a temperature-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

flux

Description: The constant value of the outward boundary heat flux for a flux-type boundary condition. To specify a function, use flux_func instead.

Default: none

Type: real

flux_func

Description: The name of a FUNCTION namelist defining a function that gives the outward boundary heat flux for a flux-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

htc

Description: The constant value of the heat transfer coefficient for either an external or interface heat transfer-type boundary condition. To specify a function, use htc_func instead.

Default: none

Type: real

htc_func

Description: The name of a FUNCTION namelist defining a function that gives the heat transfer coefficient for either an external or interface heat transfer-type boundary condition. The function is expected to be a function of (t, x, y, z) for an external heat transfer-type boundary condition, and a function of (T, t, x, y, z) for an interface heat transfer-type boundary condition. In the latter case T is taken to be the maximum of the two temperatures on either side of the interface.

Default: none

Type: string

ambient_temp

Description: The constant value of the ambient temperature for external heat transfer or radiation-type boundary condition. To specify a function, use ambient_temp_func instead.

Default: none

Type: real

ambient_temp_func

Description: The name of a `FUNCTION` namelist defining a function that gives the ambient temperature for external heat transfer or radiation-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

emissivity

Description: The constant value of emissivity for a radiation-type boundary condition. To specify a function, use `emissivity_func` instead.

Default: none

Type: real

emissivity_func

Description: The name of a `FUNCTION` namelist defining a function that gives the emissivity for a radiation-type boundary condition. The function is expected to be a function of (t, x, y, z).

Default: none

Type: string

Chapter 29

THERMAL_SOURCE Namelist

Overview

The THERMAL_SOURCE namelist is used to define external volumetric heat sources (power per unit volume). This source is in addition to any other sources coming from other physics, such as a Joule heat source. Each instance of this namelist defines a source, and the final source is the sum of all such sources (subject to some limitations).

Two forms of sources q can be defined:

(1) $q(t, \mathbf{x}) = f(t, \mathbf{x})\, \chi_S(\mathbf{x})$, where f is a user-defined function and S is a subdomain corresponding to one or more user-specified mesh cell sets. Here χ_S is the characteristic function on S: $\chi_S = 1$ for $\mathbf{x} \in S$ and $\chi_S = 0$ for $\mathbf{x} \notin S$. Sources of this form can be summed as long as the interiors of their subdomains do not intersect.

(2) $q(t, \mathbf{x}) = A(t) \sum_j q_j\, \chi_j(\mathbf{x})$, where A is a user-defined time-dependent prefactor, q_j is a constant source on mesh cell j, and χ_j is the characteristic function on cell j. The collection of values $\{q_j\}$ is read from a data file.

THERMAL_SOURCE Namelist Features

Required/Optional: Optional
Single/Multiple Instances: Multiple

Components

- name
- cell_set_ids
- data_file
- prefactor
- prefactor_func
- source
- source_func

name

Description: A unique name used to identify a particular instance of this namelist.
Type: string (31 characters max)
Default: none

`cell_set_ids`

Description: A list of cell set IDs that define the subdomain where the source is applied.

Type: a list of up to 32 integers

Default: none

Valid values: any valid mesh cell set ID

Note: Different instances of this namelist must apply to disjoint subdomains; overlapping of source functions of this form is not supported.

Exodus II mesh element blocks are interpreted by Truchas as cell sets having the same IDs.

`source`

Description: The constant value of the heat source. To specify a function, use **`source_func`** instead.

Physical dimension: $\mathsf{E/T\,L^3}$

Type: `real`

Default: none

`source_func`

Description: The name of a `FUNCTION` namelist that defines the source function. That function is expected to be a function of (t, x, y, z).

Type: string

Default: none

`data_file`

Description: The path to the data file. It is expected to be a raw binary file consisting of a sequence of 8-byte floating point values, the number of which equals the number of mesh cells. The order of the values is assumed to correspond to the external ordering of the mesh cells. The file can be created with most any programming language. In Fortran use an unformatted stream access file.

`prefactor`

Description: The constant value of the prefactor A. For a function use **`prefactor_func`**.

`prefactor_func`

Description: The name of a `FUNCTION` namelist that defines the function that computes the value of the time dependent prefactor $A(t)$.

Chapter 30

TOOLPATH Namelist (Experimental)

Overview

The TOOLPATH namelist defines a path through space, especially that taken by a machine tool, such as a laser or build platform, in the course of a manufacturing process. Its use is not limited to such cases, however. The path is specified using a simple command language that is adapted to common CNC machine languages. The current implementation is limited to a path through Cartesian 3-space (3-axis). The command language is described at the end of the chapter.

TOOLPATH Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Multiple

Components

- Name
- Command_String
- Command_File
- Start_Time
- Start_Coord
- Time_Scale_Factor
- Coord_Scale_Factor
- Write_Plotfile
- Plotfile_Dt
- Partition_Ds

Name

Description: A unique name used to identify a particular instance of this namelist. Clients will reference the toolpath using this name.

Type: case sensitive string (31 characters max)

Default: none

`Command_String`

Description: A string from which to read the commands that define the toolpath. This is only suitable for relatively simple paths; use `Command_File` instead for more complex paths.

Type: string (1000 characters max)

Default: none

Notes: Use single quotes to delimit the string to avoid conflicts with double quotes used within the commands.

`Command_File`

Description: The path to a file from which to read the commands that define the toolpath. If not an absolute path, it will be interpreted as a path relative to the Truchas input file directory.

Type: string

Default: none

Notes: C++ style comments may used in the file; all text from the '`//`' to the end of the line is ignored.

`Start_Time`

Description: The starting time of the toolpath.

Type: real

Default: 0

`Start_Coord`

Description: The starting coordinates of the toolpath.

Type: real 3-vector

Default: $(0, 0, 0)$

`Time_Scale_Factor`

Description: An optional multiplicative factor by which to scale all time values. This applies to all namelist variables as well as toolpath commands.

Type: real

Default: 1

Valid values: > 0

Notes: This is applied appropriately to speeds and accelerations in the toolpath commands.

Coord_Scale_Factor

Description: An optional multiplicative factor by which to scale all coordinate values. This applies to all namelist variables as well as toolpath commands.

Type: real

Default: 1

Valid values: > 0

Notes: This is applied appropriately to speeds and accelerations in the toolpath commands.

Write_Plotfile

Description: Enable this flag to have a discrete version of the toolpath written to a disk file. The file will be located in the Truchas output directory and be named `toolpath-`*name*`.dat`, where *name* is the name assigned to the namelist. If enabled, `Plotfile_Dt` must be specified.

Type: logical

Default: `.false.`

Notes: The file is a multi-column text file where each line gives the toolpath data at a specific time. The columns are, in order, the segment index, the time, the three position coordinates, and the flag settings (0 for clear and 1 for set). Not all flags are written, only those that were set at some point. The initial comment line starting with a '`#`' labels the columns. Data is written at the end point times of each path segment, and at zero or more equally spaced times within each segment interval. The latter frequency is determined by `Plotfile_Dt`.

Plotfile_Dt

Description: Output time frequency used when writing the toolpath to a disk file. See `Write_Plotfile`.

Type: real

Default: none

Valid values: > 0

Partition_Ds

Description: Assign a value to this parameter to generate an additional discrete version of the toolpath that is required by some clients. The value specifies the desired spacing in path length. Refer to the client's documentation on whether this is needed and for further information.

Type: real

Default: none

Valid values: > 0

Notes: Some clients require a discete version of the toolpath that consists of a time-ordered sequence of (time, coordinate) pairs. The sequence includes the end points of the segments. In addition each segment is partitioned into one or more parts of equal path length approximately equal to, but no greater than `Partition_Ds`.

The Toolpath Command Language

A toolpath is represented as a sequence of $n+2$ continuous path segments defined on time intervals $(-\infty, t_0]$, $[t_0, t_1]$, $[t_1, t_2]$, ..., $[t_n, \infty)$. The individual segments are simple paths (e.g., no motion or linear motion) that are defined by path commands. A set of flags (0 through 31) is associated with each path segment. Clients of a toolpath can use the setting of a flag (set or clear) for a variety of purposes; for example, to indicate that a device is on or off for the duration of the segment.

The toolpath command language is expressed using JSON text. The specification of a toolpath takes the form

`[` *command*`,` *command*`,` ... `]`

where *command* is one of the following:

`["dwell",`*dt*`]`
: Remain at the current position for the time interval *dt*.

`["moverel",[`*dx*`,`*dy*`,`*dz*`],`*s*`,`*a*`,`*d*`]`
: Linear displacement from the current position. Motion accelerates from rest to a constant speed, and then decelerates to rest at the position (dx, dy, dz) relative to the current position. The linear speed, accleration, and deceleration are s, a, and d, respectively. If d is omitted, its value is taken to be a. If both a and d are omitted then instantaneous acceleration/deceleration to speed s is assumed.

`["setflag",`n_1`,`n_2`,`...`]`

`["clrflag",`n_1`,`n_2`,`...`]`
: Sets or clears the listed flags. 32 flags (0 through 31) are available. The setting of a flag holds for all subsequent motions (above) until changed.

Except for the integer flag numbers, the real numeric values may be entered as integer or floating point numbers; that is, "1" is an acceptable alternative to "1.0".

The initial and final unbounded path segments are automatically generated and are not specified. The initial segment is a dwell at the position given by `Start_Coord` that ends at time t_0 given by `Start_Time`. Likewise the final segment is a dwell starting at a time and at a position determined by the preceding sequence of path segments. All flags start clear in the initial segment.

Chapter 31

TURBULENCE Namelist

Overview

The presence of the TURBULENCE namelist enables a simple algebraic turbulence model for viscous flow problems. The turbulent kinetic viscosity ν_t ($= \mu_t/\rho$) is taken to be

$$\nu_t = c_\mu k^{\frac{1}{2}} l \tag{31.1}$$

where c_μ is a proportionality constant, l is a length scale corresponding to the eddy size, and

$$k = f \cdot \tfrac{1}{2} u^2 \tag{31.2}$$

is the local turbulent kinetic energy per unit mass, modeled as a fraction f of the mean kinetic energy. The namelist variables give values for the model parameters. See *Truchas Physics and Algorithms* for more details.

TURBULENCE Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Single

Components

- CMU
- KE_fraction
- Length

Turbulence_CMU

Description: Value of the parameter c_μ in (31.1).

Type: real

Default: 0.05

Valid values: > 0

Notes: The default value is appropriate in most situations.

Turbulence_KE_Fraction

Description: Value of the parameter f in (31.2).

Type: real

Default: 0.1

Valid values: $(0, 1)$

Notes: The default value is appropriate in most situations.

Turbulence_Length

Description: Value of the length scale parameter l in (31.1).

Physical dimension: L

Type: real

Default: none

Valid values: > 0

Notes: The value should correspond to the size of the turbulent eddies. In turbulent pipe flow, for example, this would be one third the pipe radius.

Chapter 32

VFUNCTION Namelist

Overview

Similar to the FUNCTION namelist, the VFUNCTION namelist is used to define a vector-valued function that can be used in some situations where vector-valued data is needed, such as the specification of the boundary velocity in a flow boundary condition.

These are general vector-valued functions of one or more variables, $\mathbf{y} = (y_1, \ldots, y_m) = \mathbf{f}(x_1, \ldots, x_n)$. The dimension of the value, the number of variables, and the unknowns they represent (i.e., time, position, temperature, etc.) all depend on the context in which the function is used, and that is described in the documentation of those namelists where these functions can be used. Currently only a single function type can be defined:

Tabular Function. This is a continuous, single-variable function $\mathbf{y} = \mathbf{f}(s)$ linearly interpolated from a sequence of data points $(s_i, \mathbf{y}_i)$, $i = 1, \ldots, p$, with $s_i < s_{i+1}$. The variable x_d that is identified with s is specified by tabular_dim.

FUNCTION Namelist Features

Required/Optional: Optional

Single/Multiple Instances: Multiple

Components

- Name
- Type
- Tabular_Data
- Tabular_Dim

Name

Description: A unique name by which this vector function can be referenced by other namelists.

Type: A case-sensitive string of up to 31 characters.

Default: None

`Type`

Description: The type of function defined by the namelist.

Type: case-sensitive string

Default: none

Valid values: `"tabular"` for a tabular function

`Tabular_Data`

Description: The table of values $(s_i, \mathbf{y}_i)$ defining a tabular function $\mathbf{y} = \mathbf{f}(s)$. Use `Tabular_Dim` to set the variable identified with s.

Type: real array

Default: none

Notes: This is a $(m+1) \times p$ array that is most easily specified in the following manner

$$
\begin{array}{l}
\texttt{Tabular\_Data(:,1) = } s_1,\ y_{11},\ \ldots,\ y_{m1} \\
\texttt{Tabular\_Data(:,2) = } s_2,\ y_{12},\ \ldots,\ y_{m2} \\
\qquad\qquad\qquad\qquad \vdots \\
\texttt{Tabular\_Data(:,p) = } s_p,\ y_{1p},\ \ldots,\ y_{mp}
\end{array}
$$

`Tabular_Dim`

Description: The dimension in the m-vector of independent variables that serves as the independent variable for the single-variable tabular function.

Type: integer

Default: 1

Chapter 33

VISCOPLASTIC_MODEL Namelist

Overview

The VISCOPLASTIC_MODEL namelist defines the viscoplastic model to be used for a particular solid material phase in material stress-strain calculations. Two viscoplastic models are available: a mechanical threshold stress (MTS) model and a power law model. When no model is given for a solid material phase, it is modeled as a purely elastic material. The models specify a relation for the effective plastic strain rate $\dot{\epsilon}$ as a function of temperature T and von Mises stress σ. For more details on the models see the *Truchas Physics and Algorithms* manual. Briefly:
Power Law Model. In the simple power law model, the strain rate relation is

$$\dot{\epsilon} = A \exp(-Q/RT)\, \sigma^n, \tag{33.1}$$

where A, n, Q, and R are parameters given by this namelist.
MTS Model. The MTS model uses the strain rate relation

$$\dot{\epsilon} = \dot{\epsilon}_{0i} \exp\left[-\frac{\mu b^3 g_{0i}}{kT}\left(1 - \left(\frac{\mu_0}{\mu\,\sigma_i}(\sigma - \sigma_a)\right)^{p_i}\right)^{q_i}\right], \qquad \mu = \mu_0 - \frac{D}{\exp(T_0/T) - 1} \tag{33.2}$$

where $\dot{\epsilon}_{0i}$, g_{0i}, b, k, D, μ_0, T_0, σ_i, σ_a, p_i, and q_i are parameters given by this namelist. When $\sigma < \sigma_a$ we instead use $\dot{\epsilon} = K\,\sigma^5$, where K is chosen to give continuity with the previous relation at $\sigma = \sigma_a$. And when $\sigma - \sigma_a > \mu\,\sigma_i/\mu_0$ we take $\dot{\epsilon} = \dot{\epsilon}_{0i}$.

SURFACE_TENSION Namelist Features

Required/Optional: Optional; only relevant when Solid_Mechanics is true.

Single/Multiple Instances: Multiple; at most one per solid material phase.

Components

- Phase
- Model
- MTS_b
- MTS_d
- MTS_edot_0i
- MTS_g_0i
- MTS_k
- MTS_mu_0

- MTS_p_i
- MTS_q_i
- MTS_sig_a
- MTS_sig_i
- MTS_temp_0
- Pwr_Law_A
- Pwr_Law_N
- Pwr_Law_Q
- Pwr_Law_R

Phase

Description: The name of the material PHASE to which this viscoplastic model applies.

Type: case-sensitive string

Default: none

Model

Description: The type of viscoplastic strain rate model.

Type: case-insensitive string

Default: none

Valid values: "MTS", "power law", "elastic"

Notes: The effect of the "elastic" option is equivalent to not specifying a viscoplastic model at all; it is provided as a convenience.

MTS_b

Description: Burger's vector length b in (33.2).

Physical dimension: L

Type: real

Default: none

Valid values: > 0

MTS_d

Description: Constant D used in (33.2).

Physical dimension: F/L^2

Type: real

Default: none

MTS_edot_0i

Description: Reference strain rate $\dot{\epsilon}_{0i}$ used in (33.2).

Physical dimension: T^{-1}

Type: real

Default: none

Valid values: > 0

MTS_g_0i

Description: Material constant g_{0i} used in (33.2).

Type: real

Default: none

Valid values: > 0

MTS_k

Description: Boltzmann's constant k used in (33.2).

Physical dimension: E/Θ

Type: real

Default: none

Valid values: > 0

Note: Temperature should be expressed in Kelvin, or other temperature scale where 0 corresponds to absolute zero. If SI units are being used, k should be 1.38×10^{-23}. Use a value appropriate to the units used in (33.2).

MTS_mu_0

Description: Reference value μ_0 for the temperature dependent shear modulus used in (33.2).

Physical dimension: F/L^2

Type: real

Default: none

Valid values: > 0

MTS_p_i

Description: Exponent term p_i used in (33.2).

Type: real

Default: none

Valid values: > 0

MTS_q_i

Description: Exponent term q_i used in (33.2).

Type: real

Default: none

Valid values: > 0

MTS_sig_a

Description: The athermal stress term σ_a in (33.2).

Physical dimension: F/L^2

Type: real

Default: none

Valid values: ≥ 0

MTS_sig_i

Description: A stress term σ_i related to obstacles to dislocation motion in (33.2).

Physical dimension: F/L^2

Type: real

Default: none

Valid values: > 0

MTS_temp_0

Description: Constant T_0 used in the temperature dependent shear modulus equation in (33.2).

Physical dimension: Θ

Type: real

Default: none

Valid values: > 0

Pwr_Law_A

Description: Constant term A in (33.1).

Physical dimension: F/L^2

Type: real

Default: none

Valid values: ≥ 0

Pwr_Law_N

Description: Stress exponent term n in (33.1).

Type: real

Default: none

Valid values: > 0

Pwr_Law_Q

Description: Activation energy Q in (33.1).

Physical dimension: E/mol

Type: real

Default: none

Valid values: ≥ 0

Pwr_Law_R

Description: Gas constant R in (33.1).

Physical dimension: E/(Θ mol)

Type: real

Default: none

Valid values: > 0

Note: Temperature should be expressed in Kelvin, or other temperature scale where 0 corresponds to absolute zero. Use the value for R appropriate to the units used in (33.1).

Bibliography

[1] W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 141:112–152, 1998.

[2] Paul F Fischer. Projection techniques for iterative solution of ax= b with successive right-hand sides. *Computer methods in applied mechanics and engineering*, 163(1-4):193–204, 1998.

[3] HYPRE: High performance preconditioners. `http://www.llnl.gov/CASC/hypre/`.

[4] HYPRE Reference Manual. Available at `http://www.llnl.gov/CASC/hypre/`.

[5] HYPRE User's Manual. Available at `http://www.llnl.gov/CASC/hypre/`.

[6] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM*, 17(4):589–602, 1970.

[7] G. D. Sjaardema, L. A. Schoof, and V. R. Yarberry. EXODUS II: A finite element data model. Technical Report SAND92-2137 (revised), Sandia National Laboratories, 2006. Library source code: `http://sourceforge.net/projects/exodusii/`.

[8] Sandia National Laboratories. The CUBIT tool suite. `http://cubit.sandia.gov`.

[9] B. Hendrickson and R. Leland. The Chaco user's guide version 2.0. Technical Report SAND95-2344, Sandia National Laboratories, 1995.

[10] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. METIS website: `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview/`.

Index