Truchas Reference Manual

Version 20.07 — 7/27/2020

The Telluride Project

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*.

Need a discussion of what mapped restarts are and the limitations.

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.

Need documentation for the RadE tool suite.

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.

Need a discussion of the EM computational domain that must be meshed. See EM_Domain_-Type and the discussions in the P&A and User manuals.

Need a discussion of how this mesh must relate to the primary mesh insofar as grid mapping is concerned; things like the choice of element blocks.

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

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"`, `"file"`, `"block"`

Notes: See the `MESH` namelist variable `Partitioner` for a description of the options.

`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

Chapter 3

BC Namelist

Overview

The BC namelist is used to define boundary conditions for the flow and solid mechanics models at external boundaries and internal material interfaces. It also specifies the temperature and species concentrations at any material inflow boundaries. The default boundary condition for flow is free-slip.

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. Other methods are to specify the adjacent materials, and for solid mechanics, 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
- Inflow_Material
- Inflow_Temperature
- Mesh_Surface
- Node_Disp_Coords
- Surface_Materials
- 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: `"velocity"`, `"pressure"`, `"displacement"`

BC_Type

Description: The type of boundary condition.

Type: string

Default: Depends on `BC_Variable`:

```
'velocity': 'free-slip'
```

'pressure': (none)

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

Valid values: Depends on BC_Variable:

'velocity': 'free-slip', 'no-slip', 'dirichlet'

'pressure': 'dirichlet'

'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.
- It is not permitted to specify a 'pressure' 'dirichlet' condition and any type of 'velocity' boundary condition on the same face.

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
"velocity"	"dirichlet"	velocity	L/T	3
"velocity"	"free-slip"	not used	—	0
"velocity"	"no-slip"	not used	—	0
"pressure"	"dirichlet"	pressure	F/L^2	1
"displacement"	"x-displacement"	displacement	L	1
"displacement"	"y-displacement"	displacement	L	1
"displacement"	"z-displacement"	displacement	L	1
"displacement"	"x-traction"	traction (force/area)	F/L^2	1
"displacement"	"y-traction"	traction (force/area)	F/L^2	1
"displacement"	"z-traction"	traction (force/area)	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

BC_Table

Description: Table of values that describes time-dependent velocity boundary data. This is an alternative to `BC_Value`, but it can only be used for Dirichlet velocity boundary conditions at present.

Type: list of real values

Default: none

Notes: The list of values assigned to `BC_Table` take the form

$$\texttt{BC\_Table =} \quad \begin{array}{llll} t_1, & u_1, & v_1, & w_1, \\ t_2, & u_2, & v_2, & w_2, \\ \ldots, & & & \\ t_n, & u_n, & v_n, & w_n \end{array}$$

which specifies the velocity (u_j, v_j, w_j) at time t_j, $j = 1, 2, \ldots, n$. The times must be in ascending order, and linear interpolation is used between points. For $t < t_1$, (u, v, w) is taken equal to (u_1, v_1, w_1), and for $t > t_n$, (u, v, w) is taken equal to (u_n, v_n, w_n); that is, the velocity is continued as a constant outside the interval $[t_1, t_n]$. As many as 16 points may be specified. Note that it is not necessary to arrange the values one point per line, as illustrated.

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.

Change the criterion

`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

`Inflow_Material`

Description: Material number of the fluid flowing into the computational domain.

Type: integer

Default: none

Notes: If not specified, the materials fluxed into a cell through a boundary face will be in proportion to the material volume fractions present in the cell.

`Inflow_Temperature`

Description: Temperature of the fluid flowing into the computational domain.

Physical Dimension: Θ

Type: real

Default: none

`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_Materials

Description: Material number(s) of the material(s) adjacent to the BC surface, which can be internal or external to the computational domain.

Type: integer or integer pair

Default: none

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`
`"material boundary"`	`Surface_Materials` (two values)
`"external material boundary"`	`Surface_Materials` (one value)
`"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_number
- 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_number`

Description: Material number of material occupying the volume of this body.

Type: `integer`

Default: (none)

Valid values: Existing material number.

`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_SSOR_Relax
- PC_SSOR_Sweeps
- Rel_Conc_Tol
- Rel_Enthalpy_Tol
- Rel_Temp_Tol
- Residual_Atol
- Residual_Rtol
- Stepping_Method
- Verbose_Stepping

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: E/(Θ 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: 1

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_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 $||r||_2 \leq \max\{\epsilon_1, \epsilon_2||r_0||_2\}$.

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.`

Chapter 6

DS_SOURCE Namelist

The DS_SOURCE namelist is used to define external volumetric sources for the heat/species transport model. This source is in addition to any other sources coming from other physics, such as a Joule heat source.

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:

With Heat_Transport enabled:	"temperature"
With Species_Transport enabled:	"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 PHASE 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.

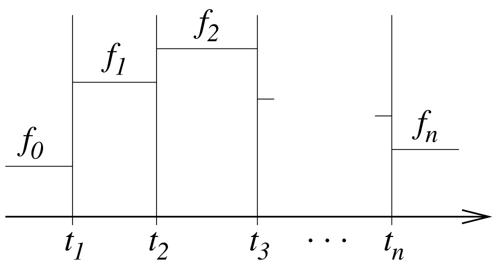


Figure 7.1: Piecewise constant function form showing the constant values $f_0, f_1, \ldots, f_n$ in relation to the time partition $t_1, t_2, \ldots, t_n$.

Chapter 8

ENCLOSURE_RADIATION Namelist

Overview

Need an overview of the namelist.

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

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.

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.

`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

Need an overview of the namelist.

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.

Need a discussion of origin of the model and a reference.

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)

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:

Say something about diffuse advection for multi-materials.

material_priority

Description: A list of material numbers that defines the priority order in which material interfaces are reconstructed within a cell for volume tracking. All fluid material numbers must be listed, and if the problem includes any solid materials, this list must include a -1, which stands for all solid materials lumped together. The default is the list of fluid materials in input file order, followed by a -1 for the lumped solids.

Type: integer list

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

We should try to give some guidance here, and/or give some indication of the issues involved.

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.

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 material number of the fluid 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 $\|r\| \leq \max\{\epsilon_1, \epsilon_2\|b\|\}$, where $r = Ax - b$ is the residual of the linear system.

Type: `real`

Default: None

Note:

Any guidance here? Pressure vs viscous solve?

`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_pcg_iter`

Description: The maximum number of preconditioned Krylov iterations allowed.

What happens if convergence is not finally achieved within this number of iterations?

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

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

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.

Add instructions for building a shared library? To Installation Guide?

`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 $\mathbf{a}$ 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, ..., x_n
Tabular_Data(2,:) = y_1, y_2, ..., 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, y_1
               x_2, y_2
                 ...
               x_n, 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.

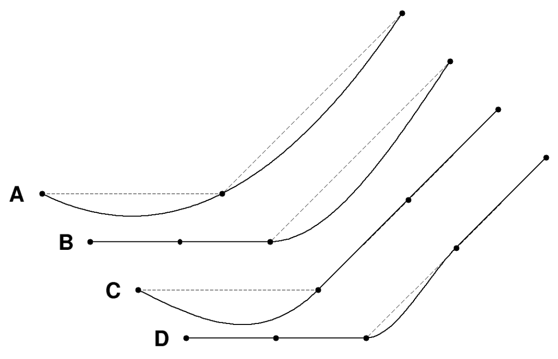


Figure 14.1: Examples of smooth Akima interpolation.

`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.

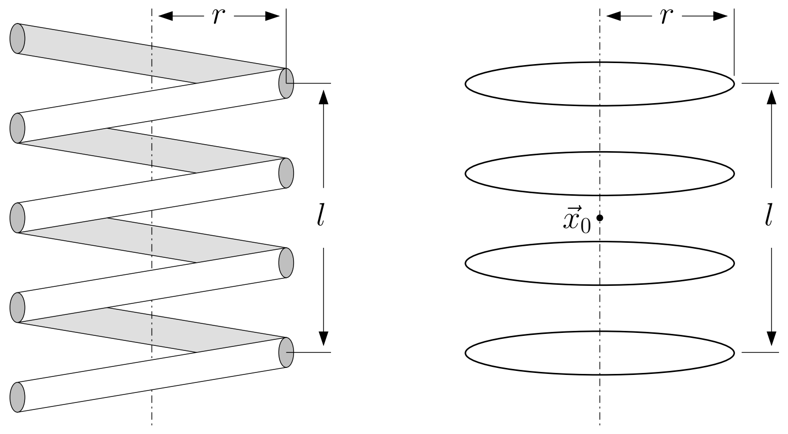


Figure 15.1: Physical 4-turn helical coil with extended wire cross section (left), and its idealized model as a stacked array of circular current loops (right).

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

LEGACY_FLOW Namelist

Overview

The LEGACY_FLOW namelist specifies the parameters for the original fluid flow model and algorithm. This namelist is read whenever the PHYSICS namelist option Legacy_Flow is enabled. Parameters for the linear solvers employed by the algorithm are specified using LINEAR_SOLVER namelists. Flow boundary conditions are defined using BC namelists.

Several variations of the basic fluid flow model are supported by this solver:

- Fluid viscosity is enabled by setting `Inviscid = .false.` and defining the dynamic viscosity μ of each fluid material phase using the PHASE property `"viscosity"`.
- Stokes flow is enabled using the Stokes flag. This presumes viscous flow is enabled.
- A simple algebraic turbulence model is enabled when input is provided for the optional TURBULENCE namelist. This also presumes viscous flow is enabled.
- A model for fluid drag in mushy zones is enabled using the Porous_Flow flag.
- Surface tension along fluid material interfaces is enabled using the Surface_Tension flag.

NB: This legacy flow model is deprecated and slated for eventual removal. Users should begin migrating to the new flow model.

LEGACY_FLOW Namelist Features

Required/Optional: Required when legacy flow physics is enabled.

Single/Multiple Instances: Single

Components

Physics Options

- Inviscid
- Porous_Flow
- Stokes
- Surface_Tension

Numerical parameters

- advection_order_energy
- advection_order_momentum

- Body_Force_Face_Method
- Body_Force_Implicitness
- CSF_Mollify_Passes
- Courant_Number
- FF_Discrete_Ops_Type
- Flux_Vol_Iter_Max
- Interface_Area
- Interface_Geometry
- Interface_Smoothing_Length
- Interface_Topology_Model
- Limiter_Type
- Mass_Limiter
- Mass_Limiter_Cutoff
- Mechanical_Energy_Bound
- MinFaceFraction
- Momentum_Solidify_Implicitness
- Porous_Implicitness
- Projection_Linear_Solution
- Surften_Number
- Viscous_Implicitness
- Viscous_Linear_Solution
- Viscous_Number
- Void_Pressure
- Volume_Track_Brents_Method
- Volume_Track_Interfaces
- Volume_Track_Iter_Max
- Volume_Track_Iter_Tol
- Volume_Track_Subcycles

Inviscid

Description: Disables the calculation of the viscous term of the flow equations. The inviscid option is incompatible with the Stokes option.

Type: logical

Default: true

Porous_Flow

Description: Adds fluid drag in mushy zones, according to a porous media flow model of Voller and Prakash [7]. If enabled, Permeability_Constant must be assigned a value for the solid phase.

Type: logical

Default: false

Stokes

Description: Disables the calculation of momentum advection.

Type: logical

Default: false

Surface_Tension

Description: Enables the calculation of surface tension effects along fluid material interfaces. If enabled, the
SURFACE_TENSION namelist is needed to specify the parameters associated with this physics model.

Type: logical

Default: false

advection_order_energy

Description: Order of accuracy of the algorithm for the advection of enthalpy between mesh cells. The first order approximation uses a donor cell method. The second order approximation uses the flux limited method.

Type: integer

Default: 1

Valid values: $\{1, 2\}$

advection_order_momentum

Description: Order of accuracy of the algorithm for the advection of momentum between mesh cells. The first order approximation uses a donor cell method. The second order approximation uses the flux limited method.

Type: integer

Default: 1

Valid values: $\{1, 2\}$

Body_Force_Face_Method

Description: Allows to select a cell-centered or face-centered formulation for the body force treatment. A cell-centered method might be required to achieve a steady-state solution for certain multi-fluid situations that should remain stable. By default, a face-centered formulation for the body-force is used.

Type: logical

Default: true

Body_Force_Implicitness

Description: Degree of time implicitness used for the body-forces in the discrete form of the momentum equations. If θ is the body-force implicitness, then the body-force at time level θ is given by $f^\theta = (1-\theta)f^n + \theta f^{n+1}$. When θ is zero, a fully explicit treatment is specified, and when θ is unity a fully implicit treatment results.

Type: real

Default: 0.5

Valid values: $[0, 1]$

Notes: The selection of an explicit treatment of the body force does not affect the stability of the flow algorithm, but does reduce the order of accuracy in time. For problems involving fluid and void, the explicit treatment may result in noisy velocities near the interface.

`CSF_Mollify_Passes`

Description: Number of times to sweep the mesh while smoothing the cell-centered interface "color" function prior to computing mean interface curvature. The smoothed (mollified) color function, also located at cell centers, is derived from an interpolation scheme that reduces to tri-linear interpolation for orthogonal, hexahedral cells.

Type: integer

Default: 1

Valid values: $[0, 10]$

Notes: The color function prior to smoothing is initialized as the volume fraction of the appropriate material. The mollified color function is only used for curvature estimates, *not* for actual interface tracking (mass advection). Too many smoothing passes will broaden the mollified color function interface to many cell widths, resulting in curvature estimates that can be highly erroneous. One to two passes are usually sufficient for curvature estimates.

`Courant_Number`

Description: The *Courant Number* C is defined as $V\delta t/h$, where δt is the time step, h is some measure of the cell size, and V is the fluid velocity magnitude. Given the current explicit treatment of advection terms in all conservation equations, linear stability theory states that time integration of these terms is stable only when $C \leq 1.0$. Given C, V, and h, a stable timestep δt_c results when $\delta t_c < Ch/V$. A specification of C is equivalent to specifying the maximum allowed advection time step δt_c. Smaller values of C result in a smaller allowed δt_c.

Physical dimension: dimensionless

Type: real

Default: 0.5

Valid values: $(0.0, 1.0]$

Notes: For general hexahedral cells (which can be degenerate), estimation of h and V can be somewhat tricky. Currently h is computed as the distance between opposite face centroids, giving two values in 2D and three in 3D. The velocity magnitude V is derived from face-centered fluxing velocities, computed as the sum of opposite face outward-directed velocities (inward velocities are assumed zero in this estimate). If the δt_c estimate gives total outward flux volumes that exceed the given cell volume, then δt_c is further scaled back to ensure outward flux volumes do not exceed cell volumes.

`FF_Discrete_Ops_Type`

Description: Flag for choosing the numerical reconstruction method used in estimating face-centered (located at cell face centroids) spatial gradients from values of discrete cell-centered data *specifically* for the projection step in the fluid flow module. Face pressure gradient estimations (or flux estimates) are controlled by this flag.

Type: string

Default: `"default"`

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

Notes: Setting `FF_Discrete_Ops_Type` to be any value other than `'default'` overrides, for fluid flow, the value defined by `Discrete_Ops_Type`. A default value for this flag ensures that the choice made by `Discrete_Ops_Type` is used for fliud flow, if `FF_Discrete_Ops_Type` is not set.

Flux_Vol_Iter_Max

Description: Maximum number of iterations allowed in the iterative search for interior face flux volume coordinates in the volume tracking algorithm

Type: integer

Default: 10

Valid values: > 0

Notes: For the volume tracking algorithm to compute the flux volumes of each material crossing each cell face, it must compute the volume of truncation formed by the material interface plane cutting through the total face flux volume. The total face flux volume is bounded on one side by the respective face and on the opposite by a mirror image of that face set back (into the cell) a distance roughly equal to the product of the normal face flux velocity and the time step. An iteration is required to locate the points of intersection of the mirror face with the four appropriate cell edges. Input variable `Flux_Vol_Iter_Max` is the maximum allowed number of iterations used in the process of iterative search for these vertices. See the Physics and Algorithm Manual for further details.

Interface_Area

Description: Flag for activating the computation of interface areas in the volume tracking algorithm, taken to be the polygonal area of the planar interfaces in each interface cell

Type: logical

Default: false

Notes: Input variable `Interface_Area` is not activated by default because it can involve an expensive set of computations, e.g., sorting and ordering points, computing dot products, etc. When computed, however, the result is an accurate estimate of the actual interfacial surface area which is written out to the interface output (`.int`) file. This information can also be used in other interfacial models such as those for surface tension, mushy zone flow, and surface delta functions.

Interface_Geometry

Description: Flag for choosing the interface geometry approximation in the volume tracking algorithm. Interfaces are approximated as a piecewise collection of planes, one per interface cell, with the interface plane equation in each cell given by $\hat{\mathbf{n}} \cdot \mathbf{x} - \rho = 0$. This flag allows the user to specify either a first order (piecewise constant) or second order (piecewise linear) computation of the interface normal $\hat{\mathbf{n}}$ used in each interface plane equation.

Type: string

Default: `"piecewise linear"`

Valid values: `"piecewise linear"`, `"linear"`, `"plic"`, `"piecewise constant"`, `"constant"`, `"slic"`

Notes: A piecewise linear approximation for the interface results in a standard gradient computation for interface normal: $\hat{\mathbf{n}} = \nabla f_k$, where f_k is the volume fraction of material k. A truly second order implementation would be "planarity preserving", i.e., an interface plane translated or rotated through any mesh should remain planar. This algorithm is planarity preserving only in a limited set of circumstances, hence it is not in general second order. See the Physics and Algorithm Manual for details on how the discrete ∇ operator is estimated.

A piecewise constant interface approximation results in a first order approximation for the interface normal, which starts with the piecewise linear approximation outlined above. This interface normal $\hat{\mathbf{n}}$, however, is further restricted by the following rule: if $\hat{n}_x > \hat{n}_y, \hat{n}_z$, then $\hat{n}_y, \hat{n}_z = 0.0$ (and similarly for other directions). The result is an interface plane that is orthogonal to one of the coordinate axes, depending upon the relative magnitudes of the original second order estimates for normal $\hat{\mathbf{n}}$. This approximation is *always* less accurate than the piecewise linear approximation, hence is only recommended for illustrative and comparative purposes. Commercial software typically employ the piecewise constant approximation, so two identical simulations, each with different choices of `Interface_Geometry`, can illustrate the effect and impact of this algorithmic choice.

Interface_Smoothing_Length

Description: Radius of support of the convolution kernel used in estimating the interface normal $\hat{\mathbf{n}}$ by convolving the kernel gradient with the volume fraction field. This normal $\hat{\mathbf{n}}$ is used in the cell interface plane equation $\hat{\mathbf{n}} \cdot \mathbf{x} - \rho = 0$ needed by the volume tracking algorithm.

Physical dimension: `L`

Type: real

Default: 0.15

Valid values: > 0

Notes: The default value of `Interface_Smoothing_Length` is completely arbitrary, as it is *always* problem dependent. The user should therefore select a value appropriate for the simulation. A good rule of thumb for the radius of support is a length that at least encompasses cells that are nearest neighbors to the reference cell. For many cases, with a proper selection of `Interface_Smoothing_Length`, the interface normal resulting from this algorithm often yields higher accuracy (is more planarity preserving) than the `'piecewise linear'` choice for the `Interface_Geometry` input variable. See the Physics and Algorithm Manual for further details.

Interface_Topology_Model

Description: Flag for choosing the method used in estimating the interface normal $\hat{\mathbf{n}}$ in the cell interface plane equation $\hat{\mathbf{n}} \cdot \mathbf{x} - \rho = 0$ required by the volume tracking algorithm. Either a convolution method or a more traditional discrete gradient computation for $\hat{\mathbf{n}} = \nabla f_k$ (f_k is the material k volume fraction) can be used.

Type: string

Default: `"least squares model"`

Valid values: `"least squares model"`, `"convolution model"`, `"convolve fast"`

Notes: See the Physics and Algorithm Manual for further details on the algorithms used for estimating interface topology.

Limiter_Type

Description: Second (and higher) order discretizations of conservation law advection terms typically require a method for enforcing monotonicity of the advected data. One method for preserving monotonicity is a “slope limiter”, of which there are numerous types. This input variable is a flag for choosing the manner by which the slope limiter is computed.

Type: string

Default: `"Barth"`

Valid values: `"Barth"`, `"Venkat"`

Notes: Advected data ϕ is defined as *monotonic* if ϕ does not possess any *new* extrema after it has been integrated forward in time with a standard advection equation $\partial\phi/\partial t + \nabla\cdot(\mathbf{u}\phi) = 0$. The advection discretization is weakly monotonic if the new time level data (ϕ^{n+1}) is monotonic relative to the old data (ϕ^n). A certain class of upwind, slope-limiting numerical schemes have been found useful for enforcing weak monotonicity. Two such examples for finite volume, unstructured mesh algorithms are those devised by V. Venkatakrishnan [8] and T. Barth [9]. The Barth slope limiter belongs to a general class of van Leer limiters [10,11], while the Venkatakrishnan limiter allows a minor amount of oscillatory behavior and therefore may not enforce monotonicity in some cases.

See the Physics and Algorithm Manual for further details on the algorithms used for higher-order discretization of the conservation equation advection terms. *This input variable is currently not active owing to the fact that second order discretization of all advection terms has been disabled.*

`Mass_Limiter`

Description: Activate the use of mass limiter to avoid the generation of excessively large velocities in partially-filled fluid-void cells where the fluid mass approaches the cutoff volume-fraction, `cutvof`. This option may be required in multiphysics flow-solidification problems when heat transfer rates are high.

Type: logical

Default: false

`Mass_Limiter_Cutoff`

Description: The mass limiter relies on the value of `Cutvof` to specify the the value of a material cell volume fraction below which mass is ignored. The `Mass_Limiter` applies an exponential filter over the range of volume fractions from `Cutvof` to the value of the `Mass_Limiter_Cutoff`. The mass limiter has a maximum damping effect at a volume fraction corresponding to `Cutvof`, and no effect at `Mass_Limiter_Cutoff`.

Physical dimension: dimensionless

Type: real

Default: 10^{-5}

Valid values: $(\texttt{Cutvof}, 1)$

Notes: The use of the `Mass_Limiter` is recommended for filling problems with heat-transfer and phase-change where solidification is occuring and the heat-transfer rates are high. Excessively large values of `Mass_Limiter_Cutoff` can lead to overly damped numerical solutions.

If the input value of `Cutvof` is larger than the default or input value of `Mass_Limiter_Cutoff`, then the value of `Mass_Limiter_Cutoff` is set to a value of 10^{-3} smaller than `Cutvof`. Similarly, if `Cutvof` is larger than 10^{-3}, then `Mass_Limiter_Cutoff` is set to a value of 1.0.

`Mechanical_Energy_Bound`

Description: Control option for applying the body force at cell faces during the velocity correction step. If the sum of the kinetic energy (per unit mass) and the potential energy (also per unit mass) exceeds `Mechanical_Energy_Bound` the body force is not added at the face. The proper value of this parameter (if the model is to be introduced) is the sum of the kinetic and potential energy (per unit mass) at the inlet to the problem, increased by perhaps 10% to allow for uncertainties in the solution. This model is used to limit the acceleration of isolated droplets of material that occur as a result of volume tracking errors.

Physical dimension: E/L^3

Type: real

Default: ∞

MinFaceFraction

Description: This option sets the minimum value of the fluid density associated with any face for the projection solution of the fluid flow equations. It is specified as a fraction of the minimum fluid density (excluding void) of any fluid material specified in the input for the calculation.

Type: real

Default: 10^{-3}

Valid values: ≥ 0

Momentum_Solidify_Implicitness

Description: Degree of time implicitness used for the velocity field in the treatment of the momentum deposition associated with solidification. If θ is the time-weighting for the momentum solidification implicitness, then the velocity $\mathbf{u}$ at time level θ is given by $\mathbf{u}^\theta = (1-\theta)\mathbf{u}^n + \theta\mathbf{u}^{n+1}$. When θ is zero, a fully explicit treatment is specified, and when θ is unity a fully implicit treatment results.

Type: real

Default: 0

Valid values: $[0, 1]$

Porous_Implicitness

Description: Degree of time implicitness used for the velocity field in the treatment of the porous drag term in the fluid momentum conservation equation. If θ is the porous implicitness, then the velocity $\mathbf{u}$ at time level θ is given by $\mathbf{u}^\theta = (1-\theta)\mathbf{u}^n + \theta\mathbf{u}^{n+1}$. When θ is zero, a fully explicit treatment is specified, and when θ is unity a fully implicit treatment results.

Type: real

Default: 0

Valid values: $[0, 1]$

Projection_Linear_Solution

Description: A character string pointer to the linear solution algorithm parameters to be used in a Krylov solution of the linear pressure Poisson equation for the projection phase of the incompressible flow algorithm. This string "points" to a particular `LINEAR_SOLVER` namelist if the string matches the `Name` input variable string in the `LINEAR_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 `LINEAR_SOLVER` namelist, then the default set of linear solution algorithm parameters is used for the pressure Poisson equation.

Surften_Number

Description: The *Surface Tension Number* S is defined as $\delta t(2\pi\sigma/\rho h^3)^{1/2}$, where δt is the time step, h is some measure of the cell size, σ is the surface tension coefficient and ρ is the fluid density. It is used to restrict the time step to avoid capillary instability. Given σ, ρ, and h, a stable time step δt_σ results when $\delta t_\sigma < S(\rho h^3/2\pi\sigma)^{1/2}$. Smaller values of S result in a smaller allowed δt_σ.

Physical dimension: dimensionless

Type: real

Default: 1

Valid values: $(0, 1]$

Notes: For general hexahedral cells (which can be degenerate), estimation of h can be somewhat tricky. Currently h is computed as the distance between opposite face centroids, giving two values in 2D and three in 3D. The fluid density ρ is computed from averaging the fluid density in the cell.

`Viscous_Implicitness`

Description: Degree of time implicitness used for the velocity field in the treatment of the Newtonian viscous stress term in the fluid momentum conservation equation. If θ is the viscous implicitness, then the velocity $\mathbf{u}$ at time level θ is given by $\mathbf{u}^\theta = (1-\theta)\mathbf{u}^n + \theta\mathbf{u}^{n+1}$. When θ is zero, a fully explicit treatment is specified, and when θ is unity a fully implicit treatment results.

Type: real

Default: 0

Valid values: $[0, 1]$

`Viscous_Linear_Solution`

Description: A character string pointer to the linear solution algorithm parameters to be used in a Krylov solution of the linear velocity equation to incorporate the implicit formulation of the viscous stress tensor. This string “points” to a particular `LINEAR_SOLVER` namelist if it matches the `Name` input variable string in `LINEAR_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 `LINEAR_SOLVER` namelist, then the default set of linear solution algorithm parameters is used for this equation.

`Viscous_Number`

Description: The *Viscous Number* V_μ is defined as $\nu\delta t/h^2$, where δt is the time step, h is some measure of the cell size, and ν is the average fluid dynamic viscosity (μ/ρ, where μ is the kinematic viscosity). Given an implicit treatment of the Newtonian stress tensor in the fluid momentum conservation equation, linear stability theory states that the time integration of this term is unconditionally stable for all diffusion time steps δt_μ (and hence all values of V_μ). For an explicit treatment, however, stable timestep δt_μ results only when $\delta t_\mu < V_\mu h^2/\nu$, where the V_μ depends upon the problem dimension and cell geometry (e.g., V_μ is 1/2, 1/4, and 1/6 in 1-D, 2-D, and 3-D, respectively, for uniform rectilinear cells and constant μ). A specification of V_μ is equivalent to specifying the maximum allowed viscous diffusion time step δt_μ. Smaller values of V_μ result in a smaller allowed δt_μ.

To promote a stable solution, the permitted value of Viscous_Number is limited to the theoretical stability limit (described in the Default section below). That is, values above the stability limit are reduced to it during input processing.

Physical dimension: dimensionless

Type: real

Default: depends upon the value of Viscous_Implicitness. Set θ to be Viscous_Implicitness. With $\theta = 0$ (explicit simulations) *Viscous Number*$=1/2, 1/4, 1/6$ for 1-D, 2-D and 3-D respectively. For $0.0 < \theta < 1.0$, *Viscous Number* $= \frac{1/2}{1-\theta}, \frac{1/4}{1-\theta}, \frac{1/6}{1-\theta}$ for 1-D, 2-D and 3-D respectively.

Valid values: $[0.0, \infty)$

Notes: *Viscous Number = 0* is treated as a special case indicating that no limitation should be placed upon the time step due to viscous effects.

For general hexahedral cells (which can be degenerate), estimation of h^2 can be somewhat tricky. Currently h^2 is computed as the squared distance between opposite face centroids, giving two values in 2-D and three in 3-D. Multiple candidate values of $\delta t_\mu < V_\mu h^2/\nu$ are then computed (one for each pair of cell faces), with the final time step constraint being taken as the minimum of the candidate values δt_μ.

Void_Pressure

Description: The uniform pressure of all void (zero density) regions.

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

Type: real

Default: 0

Volume_Track_Brents_Method

Description: Logical flag for choosing the iterative method used in finding the constant ρ in the interface plane equation $\hat{\mathbf{n}} \cdot \mathbf{x} - \rho = 0$ required by the volume tracking algorithm. For each interface cell, a separate plane equation describes the interface, with the constant ρ being needed to properly locate the plane within the cell so as to ensure volume (mass) conservation. A value of `.true.`for this flag selects Brent's method [12] for the nonlinear iterative algorithm, otherwise a standard Newton method is used.

Type: logical

Default: true

Notes: This flag defaults to `.true.`because Brent's method, while requiring 1-2 more iterations to converge relative to Newton's method, generally requires less overall CPU time because it requires one less function evaluation per iteration than Newton's method.

Volume_Track_Interfaces

Description: Logical flag for selecting a volume tracking method to represent the kinematics (movement) of *fluid* interfaces through the computational mesh. This option *must* be selected as true when multiple fluid materials having different densities are defined in a problem, as a volume tracking algorithm is currently the *only* method by which mass advection is approximated.

Type: logical

Default: False if only a single material is defined; otherwise true.

Notes: Be advised that merely adding an unused material to a single-material problem may result in slightly different simulation results. This is due to advection subcycling being enabled when the default value for this option changes.

In future versions of the code the input variable `Volume_Track_Interfaces` is likely to be replaced with a more general mass advection flag that will allow the user to select from numerous algorithms for approximating mass advection. Examples are interface tracking schemes (of which volume tracking is one of many) as well as standard continuum advection schemes.

`Volume_Track_Iter_Max`

Description: Maximum number of iterations allowed in the nonlinear iterative algorithm used to find the constant ρ in the interface plane equation $\hat{\mathbf{n}} \cdot \mathbf{x} - \rho = 0$ required by the volume tracking algorithm. For each interface cell, a separate plane equation describes the interface, with the constant ρ being needed to properly locate the plane within the cell so as to ensure volume (mass) conservation. This input variable sets a limit on the maximum number of iterations used in finding ρ.

Type: integer

Default: 20

Valid values: $(0, 100]$

Notes: Whether Brent's or Newton's method is used for the plane constant (ρ) iteration, convergence rarely requires more than $4 - 6$ iterations, hence the default of 20 should virtually never be reached in the iteration loop. Something else is likely to have gone wrong if this is the case.

`Volume_Track_Iter_Tol`

Description: Volume fraction tolerance used in declaring convergence in the nonlinear iterative algorithm employed to find the constant ρ in the interface plane equation $\hat{\mathbf{n}} \cdot \mathbf{x} - \rho = 0$ required by the volume tracking algorithm. For each interface cell, a separate plane equation describes the interface, with the constant ρ being needed to properly locate the plane within the cell so as to ensure volume (mass) conservation. Convergence is attained in each interface cell if the volume fraction truncated by the interface plane is within `Volume_Track_Iter_Tol` of the actual cell volume fraction.

Type: real

Default: 10^{-8}

Valid values: $(0, 10^{-3}]$

Notes: The default value of `Volume_Track_Iter_Tol` is much smaller than values often used in most other volume tracking algorithms, and much smaller than perhaps necessary to ensure good local and global mass conservation.

`Volume_Track_Subcycles`

Description: Volume tracking time step (δt_{vt}) specification, expressed as a multiplicative factor relative to the time step δt. If `Volume_Track_Subcycles` is 4, then $\delta t_{\mathrm{vt}} = \delta t / 4$, or 4 volume tracking time steps are required before the full δt integration is achieved.

Type: integer

Default: 2

Valid values: $(0, 20]$

Notes: Given the current naive unsplit advection algorithm [1] used for time integration, adequate "corner coupling" of the volume flux terms in the volume tracking algorithm can only be achieved by subcycling the volume tracking time integration method. In this scheme, time level n volume fractions f^n are updated after one subcycle to time level $*$ volume fractions f^*, then to f^{**} the next subcycle and so on. This process of using updated volume fractions for the next subcycle helps to ensure that volume fractions are adequately "corner coupled". A value for `Volume_Track_Subcycles` of at least 4 is recommended.

Chapter 17

LINEAR_SOLVER Namelist

Overview

The LINEAR_SOLVER namelist sets parameters used in the solution of linear systems of equations as defined in the NUMERICS namelist.

LINEAR_SOLVER Namelist Features

Required/Optional: Optional
Single/Multiple Instances: Multiple

Components

- Convergence_Criterion
- Krylov_Vectors
- Maximum_Iterations
- Method
- Name
- Output_Mode
- Preconditioning_Method
- Preconditioning_Scope
- Preconditioning_Steps
- Relaxation_Parameter
- Status_Frequency
- Stopping_Criterion

In the sections that follow we refer to the linear system being solved as $Ax = b$, where:

- A is the coefficient matrix,
- x is an estimate of the solution vector,
- b is the source vector,
- N is the number of unknowns,
- r is an estimate of the residual, $r = b - Ax$,
- x_0 is the initial guess for the solution vector, and
- r_0 is the initial estimate of the residual, $r_0 = b - Ax_0$.

`Convergence_Criterion`

Description: Value of error estimate used to determine when convergence has been reached. Meaning depends on `Stopping_Criterion`.

Type: `real`

Default: 10^{-8}

Valid values: $(0, 0.1)$

`Krylov_Vectors`

Description: Number of vectors used by GMRES and FGMRES in orthogonalization. Also determines frequency of restart in GMRES and FGMRES.

Type: `integer`

Default: max(10, min(100,N_{cells}))

Valid Values: $(0, N_{cells})$

Notes:

- relevant only when `Method` = `'gmres'` or `Method` = `'fgmres'`
- In general, as the size of `Krylov_Vectors` increases, the likelihood of successful convergence increases, but so does cost/iteration and memory usage. Start with the default value and increase only if convergence difficulties are encountered.
- To insist that GMRES and FGMRES never restart, set `Krylov_Vectors` to a value greater than or equal to `Maximum_Iterations`.

`Maximum_Iterations`

Description: Maximum number of iterations allowed.

Type: `integer`

Default: max(20, min(1000, $2 * \sqrt{N_{cells}}$))

Valid values: > 0

Notes: If a solution is not found by the time this number of iterations has been performed, execution stops, and an message is written to the `.err` file. In addition, a file containing the residuals at the last successful iteration is written. The filename will be of the form `prob.name.00000`, where `prob` is the basename of the problem input file and `name` is the name of the linear solver that failed, i.e. `Name`. The format of the file is appropriate for use with the visualization tool GMV.

When comparing values found in the graphics dump of the residuals and the values reported in the `.err` file keep in mind that the values in the error file are norms (and scaled norms) of residuals rather than the residuals themselves.

`Method`

Description: Algorithm used for solution of linear systems.

Type: string

Default: `'fgmres'`

Valid values: **`'cg'`** - preconditioned Conjugate Gradients (CG)

In the absence of roundoff, CG is guaranteed to converge in N iterations for symmetric positive definite N by N systems. Although the guarantee is lost in the real world, with effective preconditioning it can require significantly fewer than N iterations. Also, it can successfully solve slightly non-symmetric systems.

`'gmres'` - left-preconditioned Generalized Minimal Residuals (GMRES)

Effective on non-symmetric systems, but more expensive and somewhat more difficult to use due to dependence on the Krylov subspace size, `Krylov_Vectors`.

`'fgmres'` - preconditioned Flexible GMRES (FGMRES)

Extension of GMRES that implicitly applies right-preconditioning and allows the preconditioner to vary at each iteration. As with GMRES, convergence is dependent on choice of `Krylov_Vectors`.

`'tfqmr'` - preconditioned Transpose-Free Quasi-minimal Residuals (TFQMR)

An alternative to GMRES for non-symmetric systems. Can converge more quickly than GMRES and requires no additional parameters, but requires two matrix-vector products and two applications of the preconditioner per iteration and can break down (i.e. fail).

`'bcgstab'` - preconditioned stabilized Bi-Conjugate Gradients (Bi-CGSTAB)

Another alternative to GMRES for non-symmetric systems, with the same advantages and disadvantages over GMRES as TFQMR but different convergence behavior.

`'none'` - only call preconditioner

In this case, only the preconditioner is called—there is no linear solution which calls the preconditioner, and no testing for convergence. In this case, we ignore the following `LINEAR_SOLVER` namelist entries: `Convergence_Criterion`, `Krylov_Vectors`, `Maximum_Iterations`, and `Stopping_Criterion`. Choosing 'none' is suitable for the case when the `LINEAR_SOLVER` namelist is pointed to by a `NONLINEAR_SOLVER` namelist with `Method` `'ain'`.

`Name`

Description: Arbitrary but unique string to identify a set of linear solver settings

Type: string

Default: `'default'`

Valid values: arbitrary string

Notes: If `Projection_Linear_Solution` in the `NUMERICS` namelist, or `Linear_Solver_Name` in the `NONLINEAR_SOLVER` namelist, is set to a string other than `'default'`, it should match the `Name` string in one and only one `LINEAR_SOLVER` namelist.

`Output_Mode`

Description: Controls verbosity of linear solver.

Type: string

Default: `'none'`

Valid values: `'none'` - quiet

`'errors'` - errors only

`'errors+warnings'` - errors and warnings

`'warnings+errors'` - same as above

`'summary'` - errors, warnings, plus a one-line summary consisting of the iteration number, the norms of the calculated residual and error estimate, and the norms of the true residual and error estimate (if calculated) each time convergence is checked

`'iterates'` - same as `'summary'`, plus the coefficient, preconditioner (if there is one), source, initial guess and converged solution, plus the current iterate and true residual (if computed) at each iteration

`'full'` - same as `'iterates'`, plus more intermediate values computed during the course of each iteration (note that this can generate large amounts of output, and is primarily for debugging)

`Preconditioning_Method`

Description: Algorithm used to precondition the Krylov iteration

Type: string

Default: `'none'`

Valid values for `Projection_Linear_Solution`: **`'none'`** - no preconditioning

- **`'diagonal'`** - multistep weighted diagonal
- **`'jacobi'`** - multistep weighted Jacobi
- **`'ssor'`** - multistep weighted Symmetric Gauss-Seidel, a.k.a. Symmetric Successive Over-Relaxation
- **`'ilu0'`** - incomplete LU factorization (ILU) with no fill-in
- **`'lu'`** - LU decomposition

Valid values for `Viscous_Linear_Solution`: **`'none'`** - no preconditioning

- **`'diagonal'`** - multistep weighted diagonal

Note: relevant when `Method` = `'cg'`, `'fgmres'`, `'gmres'`, `'tfqmr'`, or `'bcgstab'`

`Preconditioning_Steps`

Description: Number of passes to be performed in Jacobi or SSOR preconditioning

Type: `integer`

Default: 1

Valid values: > 0

Note: Relevant only when `Preconditioning_Method` is set to `'jacobi'` or `'ssor'`.

`Preconditioning_Scope`

Description: For parallel runs, determines whether preconditioner is global (requiring communication each time the preconditioner is applied) or local (requiring no communication).

Type: string

Default: `'global'`

Valid values: `'local'`

`'global'`

`Relaxation_Parameter`

Description: Relaxation parameter used in Jacobi and SSOR preconditioning

Type: `real`

Default: 0.90

Valid values: $(0, 2)$

Note: Relevant only when `Preconditioning_Method` is set to `'jacobi'` or `'ssor'`.

`Status_Frequency`

Description: Frequency of linear solver status reports to the tty

Type: `integer`

Default: 0

Valid values: ≥ 0

Note: If set to a positive integer value n, a one-line status report will be sent to the tty every n iterations containing the iteration number as well as $\|r\|$, the relative change in $\|r\|$, and the current error estimate for that iteration.

Note that this output will occur each cycle (but only for the linear solver for which `Status_Frequency` is set).

`Stopping_Criterion`

Description: Test used to estimate error and determine when convergence has been reached. Value is set by `Convergence_Criterion`.

Type: string

Default: `'||r||'`

Valid values: **`'||r||/||b||'`** - good for most situations, but can lead to difficulties when $\|A\|\|x\| >> \|b\|$ - in that case, use the following criterion

`'||r||/(||A||*||x||+||b||)'` - useful in cases mentioned above, but note that it requires $\|A\|$ (or some estimate), which may not be available - in addition, it increases the cost of GMRES somewhat

`'||r||/||r0||'` - useful, but suffers from dependence on the initial guess

`'||r||/||x||'` - useful, but increases the cost of GMRES and FGMRES significantly and isn't dimensionless

`'||r||'` - note that this test isn't scaled by some other characteristic "size" of the system

`'||x-xold||/||x||'` - inappropriate for nonstationary methods such as CG and GMRES - NOT RECOMMENDED

Notes: Unfortunately, in the current version of the code it is difficult to automatically set different default stopping criteria for different physics. Hence the user must set the stopping criterion explicitly. Recommended criteria for the various physics / numerics are shown in the following table:

Physics / Numerics	`LINEAR_SOLVER` namelist	**Criterion**
Pressure Poisson	`Projection_Linear_Solution`	$\|r\|$
Viscous Stress	`Viscous_Linear_Solution`	$\|r\|/\|b\|$

For the Pressure Poisson equation, used in the projection-step of the flow solution algorithm, the choice of $\|r\|$ makes it possible to relate the convergence criterion to the divergence error measured at each time step.

The divergence error is computed in terms of the fluxing (face) velocities as

$$\epsilon_{div} = \| \sum_f \mathbf{u}_f \cdot \mathbf{n}_f A_f \ \delta t / V \|$$

where $\mathbf{u}_f$ is the face-velocity, $\mathbf{n}_f$ is the face-normal, A_f is the face-area, δt is the time-step, and V is the cell volume.

Internally, the pressure Poisson problem is scaled in a non-dimensional way so that $\|r\| \approx \epsilon_{div}$. Thus, choosing $\|r\|$ as the convergence criteria ensures that the L_2 norm of the divergence error will be bounded by the specified convergence criteria for the pressure Poisson solve.

For the viscous stress terms, the default preconditioning method is chosen to be `diagonal`. This is adequate for most advectively dominated flow problems, but may not be sufficient in the Stokes' flow limit. This default is only active when `Inviscid = .false` in the `PHYSICS` namelist.

Chapter 18

MATERIAL Namelist

Overview

The MATERIAL namelist is used to define a few properties and other attributes of a material phase. Most properties are now defined using the PHASE and MATERIAL_SYSTEM namelists, which are expected to completely supplant this namelist in a future release.

Every PHASE namelist must have a corresponding MATERIAL namelist with the same name, and vice versa. The only exception is for a so-called void material, which is defined using this namelist by specifying a Density value of 0; there must not be a corresponding PHASE namelist for such a material. Note that there can be at most one void material.

MATERIAL Namelist Features

Required/Optional: Required

Single/Multiple Instances: Multiple. One for each material referenced by other namelists.

Components

- Density
- Immobile
- Material_Feature
- Material_Name
- Material_Number
- Permeability_Constant
- Priority
- Sound_Speed
- Void_Temperature

Density

Description: Constant mass density of the material. A value of zero establishes this material the *void* material. This density value is otherwise unused. Use the named PHASE namelist property "density" to specify the density of non-void materials.

Physical dimension: M/L^3

Type: real

Default: none

Valid values: $>= 0$

Notes: There can be at most one void material. A value must be specified for non-void materials even though it is ignored.

`Immobile`

Description: Used to specify those materials that are solid, and so do not flow.

Type: logical

Default: false

`Material_Feature`

Description: The input file must specify one and only one material as 'background'. Truchas uses the background material in its setup phase to fill in any portion of the mesh that is not explicitly defined in the 'Body' namelists. If in some Body namelist the 'Surface_Name' variable is set equal to 'background', then the 'Material_Feature' of the material specified in that body must also be set equal to 'background'.

Type: string

Default: none

Valid values: `"background"`

`Material_Name`

Description: Descriptive name of the material.

Type: string

Default: none

`Material_Number`

Description: A unique identifier for this material.

Type: integer

Default: none

Valid values: ≥ 1

`Permeability_Constant`

Description: An array of up to 3x10 material directional flow permeabilities. One value is specified for each coordinate axis. The flow permeabilities are the coefficients in the Carman-Koseny porous media drag correlation for flow in the mushy zone of solidifying alloys. These are only used for immobile materials.

This is only relevant to the legacy flow model.

Physical dimension: $\mathsf{M/(L^3\ T)}$

Type: real

Default: 0

Valid values: ≥ 0

`Priority`

Description: Material flow priority. The value is only used if the material is fluid. Lower priority fluids are moved first in the volume tracking method.

This is only relevant to the legacy flow model. Use the `material_priority` variable in the `FLOW` namelist for the standard flow model.

Type: integer

Default: none

Valid values: ≥ 1

`Sound_Speed`

Description: This variable is only used for void (zero density) fluid materials. In this case, it is the adiabatic sound speed that is used in computing the compressibility of each cell containing the material. Note that this is not a real sound speed, but a numerical artifice used to permit collapse of small void bubbles.

This is only relevant to the legacy flow model.

Physical dimension: $\mathsf{L/T}$

Type: real

Default: 0

Valid values: ≥ 0

`Void_Temperature`

Description: The temperature to be assigned to all cells that contain only material of zero density. The temperature of such cells cannot be calculated because both the enthalpy and the specific heat are zero. The value of this input variable does not affect the result of the simulation, but it does impact graphical images because of interpolation that takes place between cells, and because it may set the overall scale of contour plots.

Physical dimension: $\mathsf{\Theta}$

Type: real

Default: 0

Chapter 19

`MATERIAL_SYSTEM` Namelist

Overview

The `MATERIAL_SYSTEM` namelist groups together one or more material phases defined by `PHASE` namelists to complete the description of a material.

Need a figure that clarifies how the phase transition variables are used.

`MATERIAL_SYSTEM` Namelist Features

Required/Optional: Required

Single/Multiple Instances: Multiple

Components

- `Name`
- `Phases`
- `Number_of_Components`
- `Temperature_Dependent`
- `Reference_Temp`
- `Reference_Enthalpy`
- `Transition_Temps_Low`
- `Transition_Temps_High`
- `Smoothing_Radius`
- `Latent_Heat`

`Name`

Description: A unique name for this material system.

Type: string (31 characters max)

Default: none

`Phases`

Description: The list of one or more `PHASE` names that comprise this material system. The phases must be listed in order from low to high temperature phases.

Type: array of strings

Default: none

Note: It is legitimate for a material system to consist of a single phase. If there are multiple phases, the system must be temperature dependent and the variables associated with phase transformation must be defined.

Multi-phase, multi-component systems are not allowed currently.

`Number_of_Components`

Description: The number of components the material system contains.

Type: integer

Default: 1

Valid values: ≥ 1

Note: A multi-component material system is only compatible with problems that include species transport. This value must be 1 more than the value of `Number_of_Species`.

Multi-phase, multi-component systems are not allowed currently.

`Temperature_Dependent`

Description: Declares whether the material system phase diagram is temperature-dependent or not.

Type: logical

Default: `.true.`

Note: A single-phase, multi-material system is the only type of material system that may be independent of temperature.

`Reference_Temp`

Description: The reference temperature used to generate the specific enthalpy for the material system.

Physical dimensions: Θ

Type: real

Default: 0.0

Note: To uniquely define the specific enthalpy function from the specific heat by integration, the enthalpy at a reference temperature must be given. In the multi-phase case, the reference temperature must lie in the temperature range of the lowest-temperature phase.

`Reference_Enthalpy`

Description: The reference enthalpy used to generate the specific enthalpy for the material system.

Physical dimensions: E/M

Type: real

Default: 0.0

Note: To uniquely define the specific enthalpy function from the specific heat by integration, the enthalpy at a reference temperature must be given.

`Transition_Temps_Low`

Description: The low temperatures of the phase transition intervals.

Physical dimensions: Θ

Type: real

Default: none

Note: Between each pair of phases in sequence is a phase transformation, and that transformation is defined to occur over a temperature interval $[T_\text{low}, T_\text{high}]$, with $T_\text{low} < T_\text{high}$. This variable defines the T_low values.

In the case of multiple phase transformations, the user must ensure that the transition intervals do not overlap.

This variable is only relevant to multi-phase material systems.

`Transition_Temps_High`

Description: The high temperatures of the phase transition intervals.

Physical dimensions: Θ

Type: real

Default: none

Note: Between each pair of phases in sequence is a phase transformation, and that transformation is defined to occur over a temperature interval $[T_\text{low}, T_\text{high}]$, with $T_\text{low} < T_\text{high}$. This variable defines the T_high values.

In the case of multiple phase transformations, the user must ensure that the transition intervals do not overlap.

This variable is only relevent to multi-phase material systems.

`Smoothing_Radius`

Description: The transition function smoothing radii for the phase transformations.

Physical dimensions: Θ

Type: real

Default: $0.25\Delta T$

Valid values: $[0, 0.5\Delta T)$

Note: For better numerical performance the transition function needs to be smoothed by rounding off the corners that occur at the end points of the transition interval $[T_{\text{low}}, T_{\text{hi}}]$. This value gives the radius of the smoothing in temperature units, and defaults to one quarter of the transition width $\Delta T = T_{\text{hi}} - T_{\text{low}}$. Note that the effective transition interval width is increased by this value at either endpoint. This needs to be taken into account when ensuring that the transition intervals from multiple transformations do not overlap.

`Latent_Heat`

Description: The latent heats of the phase transformations.

Physical dimensions: E/M

Type: real

Default: none

Valid values: > 0.0

Note: Between each pair of phases in sequence is a phase transformation and associated with that transformation is a latent heat. An n-phase system requires the specification of $n - 1$ latent heats.

This variable is only relevent to multi-phase material systems.

Chapter 20

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 [13] is the only supported mesh format (also sometimes known as Genesis). This well-known format is used by some mesh generation tools (Cubit [14], 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`
- `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`

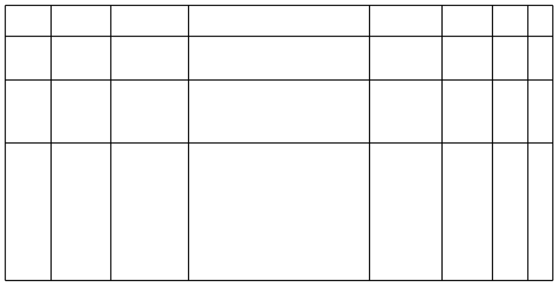
Data that describes the grid in each of the coordinate directions. The tensor product of these grids define the nodes of the 3D mesh. The data for each coordinate grid consists of these three component arrays:

`%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 20 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 20.1: 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

`partitioner`

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

Type: case-insensitive string

Default: `"chaco"`

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

Notes:

`"chaco"` uses a graph partitioning method from the Chaco library [15] to compute the mesh decomposition at run time. This is the standard method long used by Truchas.

`"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

Chapter 21

NONLINEAR_SOLVER Namelist

Overview

The NONLINEAR_SOLVER namelist sets parameters used in the solution of nonlinear systems of equations as defined in the NUMERICS namelist.

NONLINEAR_SOLVER Namelist Features

Required/Optional: Optional
Single/Multiple Instances: Multiple

Components

- Convergence_Criterion
- Damper_Parameters
- Linear_Solver_Name
- Maximum_Iterations
- Method
- Name
- NLK_Max_Vectors
- NLK_Vector_Tolerance
- Perturbation_Parameter

In the sections that follow we use the following notation.

- N is the number of unknowns
- $F(x) = 0$ is the nonlinear system being solved
- x_0 is the initial estimate of the solution vector
- x_k is the current solution vector
- x_{k+1} is an estimate of the future solution vector
- δx is the change in the solution, $x_{k+1} - x_k$
- $\|\delta x\|_\infty$ is the infinity, or max, norm of the solution delta
- $\|F(x)\|_\infty$ is the infinity, or max, norm of the nonlinear residual
- $\|F(x)\|_2$ is the 2-norm of the nonlinear residual
- $J(x_k)\delta x = -F(x_k)$ is the linear system solved at each nonlinear iteration
- $J(x_k)$ is the Jacobian matrix, $J_{ij} = \frac{\partial F_i(x)}{\partial x_j}$

NLK_Max_Vectors

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

Type: `integer`

Default: 20

Valid values: $[0, \infty)$

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: $(0, 1)$

Convergence_Criterion

Description: Value used to determine when nonlinear convergence has been reached

Type: `real`

Default: 10^{-5}

Valid values: $(0, 0.1)$

Note: We refer to the input value of `Convergence_Criterion` as ϵ.

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.

`Damper_Parameters`

Description: Parameters used to improve stability of the nonlinear iteration in some cases (α_L and α_H).
Type: `real`
Default: 1.0, 1.0
Valid values: $[0, \infty)$
Note: `Damper_Parameters` is only relevant when `Use_Damper` is set to `.true.` (note that the default value of `Use_Damper` is `.true.`).

The damper is applied as follows. Referring to the two input parameters as α_L and α_U, for lower and upper:

$$\alpha = \min_{1 \le i \le N} \begin{cases} -(1-\alpha_L)\frac{x_{k+1}(i)}{\delta x(i)} & \text{if} \quad x_{k+1}(i) + \delta x(i) < \alpha_L x_{k+1}(i) \\ -(\alpha_H - 1)\frac{x_{k+1}(i)}{\delta x(i)} & \text{if} \quad x_{k+1}(i) + \delta x(i) > \alpha_H x_{k+1}(i) \end{cases}$$

then:

$$\alpha = \max(0.1, \min(1.0, \alpha))$$

and finally:

$$x_{k+1} = x_k + \alpha\, \delta x$$

If convergence problems are encountered during nonlinear solves, try adjusting `Damper_Parameters` or setting `Use_Damper` to `.false.`.

`Linear_Solver_Name`

Description: String matching the `Name` setting for one `LINEAR_SOLVER` namelist
Type: string
Default: `'default'`
Valid values: arbitrary string

`Maximum_Iterations`

Description: Maximum number of iterations allowed
Type: `integer`
Default: 30
Valid values: $[0, \infty)$

`Method`

Description: Algorithm used for solution of nonlinear systems
Type: string
Default: `'nk'`
Valid values: **`'default'`**
`'nk'`
`'nlk'`
Note: If 'nlk' is specified, we ignore the following `NONLINEAR_SOLVER` namelist entries: `Damper_Parameters`, `Perturbation_Parameter`, `Use_Damper`.

`Name`

Description: Arbitrary but unique string to identify a set of nonlinear solver settings

Type: string

Default: `'default'`

Valid values: arbitrary string

`Perturbation_Parameter`

Description: Differencing parameter used in the Taylor series approximation of the product of the Jacobian and a vector

Type: `real`

Default: 10^{-6}

Valid values: $(0.0, 1.0)$

Note: In Truchas a Jacobian-Free implementation of Newton's method is used for the nonlinear solution. This algorithm uses a Krylov subspace method to solve the linear systems at each iteration of Newton's method. Since Krylov subspace methods require the coefficient only for matrix-vector products, we use a first-order Taylor series expansion to approximate the Jacobian times a vector:

$$Jv \approx \frac{F(x+\epsilon v)-F(x)}{\epsilon}$$

The accuracy of this approximation is obviously affected by the differencing parameter, ϵ, which in Truchas is computed as:

$$\epsilon = \frac{b\|x_{k+1}\|_\infty}{N\|v\|_2} + b$$

where b is the input variable `Perturbation_Parameter`.

The default value is usually appropriate, so this parameter should be changed only with extreme caution.

`Use_Damper`

Description: Whether or not to apply damping during the nonlinear iteration

Type: `logical`

Default: `.true.`

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

Note: See `Damper_Parameters` for a description of the damping algorithm.

If convergence problems are encountered during nonlinear solves, try adjusting `Damper_Parameters` or setting `Use_Damper` to `.false.`.

Chapter 22

`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 23

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 24

PHASE Namelist

Overview

The PHASE namelist is used to define (most of) the thermo-physical properties of a material phase. The namelist consists primarily of three parallel arrays Property_Name, Property_Constant, and Property_-Function. The name of a property is assigned to an element of Property_Name and either its constant value assigned to the corresponding element of Property_Constant, or the name of a FUNCTION namelist that defines a function to calculate the value assigned to the corresponding element of Property_Function. The namelist does not prescribe a set of property names, nor is the location of a property in the arrays significant. Although any names may be specified, Truchas will only look for and use the names listed in the following section; all others are silently ignored. The properties that are required will depend on which physics models are used. A few properties have default values, but most do not and must be specified when required. Note also that the MATERIAL_SYSTEM namelist, which groups material phases together, is required to complete the description of a material.

Because of the way materials are currently assigned to domain regions, it is necessary that every PHASE namelist have a corresponding MATERIAL namelist with the same name, and vice versa. The only exception is for the so-called void material (one with a zero density), if any; it must *not* have a corresponding PHASE namelist.

The MATERIAL namelist is also still used to assign a few material phase properties, specifically those related to fluid flow and some related to solid mechanics. Expect the PHASE and MATERIAL_SYSTEM namelists to completely supplant the MATERIAL namelist in a future release.

An example input for the PHASE namelist is shown in Fig. 24.1. This defines a solid copper phase where the density and thermal conductivity are specified as constants and the specific heat is specified as a function defined by a FUNCTION namelist with the name "Cp s-Cu".

Properties

This section describes the properties that Truchas currently recognizes and uses.

"density" Dimension: M/L^3

Mass density of the material phase. This must be a constant, and all phases belonging to the same material system must have the same density value. This property is required regardless of which physics models are used.

Properties for Fluid Flow. The following properties are used by the fluid flow model, and are only needed for fluid material phases (Immobile = .false.)

"density deviation" Dimensionless

The relative deviation $(\rho(T) - \rho_0)/\rho_0$ of the true temperature-dependent density $\rho(T)$ from the reference density ρ_0 given by the property "density". 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.

`"viscosity"` Dimension: M/(L T)

The dynamic viscosity μ of a fluid phase. If a functional form is used it is expected to be a function of temperature only. This is required only for viscous flow (`Inviscid = .false.`)

Properties for Heat Transport. The following properties are used by the heat transport model.

`"specific heat"` Dimension: E/(M Θ)

Specific heat of the material phase. If not constant, it must either be a tabular function in T or a polynomial in T without a $(\cdot)^{-1}$ term. An analytic antiderivative of the function will be generated internally and used as the enthalpy per unit mass; see the `Reference_Temp` and `Reference_Enthalpy` variables of the `MATERIAL_SYSTEM` namelist.

`"conductivity"` Dimension: E/(T L Θ)

Thermal conductivity K of the material phase. If a functional form is used, it should be a function of T, or $(T, \phi_1, \ldots, \phi_n)$ when species transport is included.

Properties for Species Transport. The following properties are relevant to the solutal species transport model. The model allows for an arbitrary number of species, and the properties for each are indicated by suffixing the property name with the index i, of the species. If a functional form is used for a property it 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 $(T, \phi_1, \ldots, \phi_n)$.

`"diffusivity`*i*`"` Dimension: L^2/T

The diffusivity D_i of species component i in the material phase.

`"soret`*i*`"` Dimension: Θ^{-1}

The Soret coefficient S_i for species component i in the material phase. This is relevant only when species transport is coupled with heat transfer. If not specified, the thermodiffusion term of the species flux will not be included. If defined for one phase, it must be defined for all phases.

Properties for solid mechanics. The following properties are used by the solid mechanics model, and are only needed for solid material phases (`Immobile = .true.`). Additional viscoplasticity parameters are defined using the `MATERIAL` namelist.

`"TM Reference Density"` Dimension: M/L^3

Mass density of the solid material phase at the specified reference temperature.

`"TM Reference Temperature"` Dimension: Θ

Temperature at which this solid material phase is stress-free.

`"TM linear CTE"` Dimension: Θ^{-1}

The linear coefficient of thermal expansion for the solid material phase.

`"Lame1"` Dimension: F/L^2

The first Lamé constant λ for the material phase. If a functional form is used, it is expected to be a function of temperature T only.

`"Lame2"` Dimension: F/L^2

The second Lamé constant G for the material phase. If a functional form is used, it is expected to be a function of temperature T only.

```
&PHASE
  Name = "solid copper"
  Property_Name(2) = "density",       Property_Constant(2) = 8920.0
  Property_Name(4) = "specific heat", Property_Function(4) = "Cp s-Cu"
  Property_name(5) = "conductivity",  Property_Constant(5) = 400.0
/
&FUNCTION
  Name = "Cp s-Cu"
  ...
/
```

Figure 24.1: Example input for a PHASE namelist.

Properties for induction heating

`"electrical conductivity"` Dimension: T I^2/(E L)

The electrical conductivity of the material phase. The default dimension is $\Omega^{-1}\text{m}^{-1}$(SI); to use other units see the discussion on units in the chapter on the ELECTROMAGNETICS namelist. This property has a default value of zero.

`"electric susceptibility"` Dimensionless

The electric susceptibility χ_e of the material phase. The relative permittivity is $1+\chi_e$. This property has a default value of zero, which is appropriate in most cases.

`"magnetic susceptibility"` Dimensionless

The magnetic susceptibility χ_m of the material phase. The relative permeability is $1+\chi_m$. This property has a default value of zero, which is appropriate in most cases.

PHASE Namelist Features

Required/Optional: Required, except for pure flow problems.

Single/Multiple Instances: Multiple

Components

- Name
- Property_Name
- Property_Constant
- Property_Function

Name

Description: A unique name for this phase.

Type: case-sensitive string (31 characters max)

Default: None

Note: Each PHASE namelist must correspond to a MATERIAL namelist having the same Material_Name, and vice versa.

Property_Name

Description: A list of properties being defined for this phase.

Type: case-sensitive strings (31 characters max)

Default: None

Valid values: The required properties depend on the physics models being used.

Note: The listed properties do not need to appear in any specific order, and the list may have gaps in it. The only requirement is that the value of a property is found in the corresponding location of the `Property_Constant` or `Property_Function` arrays.

Any names may be specified, but Truchas will only look for and use expected property names; all others are silently ignored.

Property_Constant

Description: The constant values of the properties.

Type: real

Default: None

Note: For each used component, either `Property_Constant` or `Property_Function` must specified, but not both.

Property_Function

Description: The names of `FUNCTION` namelists that define the property functions.

Type: string

Default: none

Note: The functions will be expected to be functions of certain sets of variables. Which variables will depend on the type of diffusion system; refer to the overview of the physics models given for the `PHYSICS` namelist.

For each used component, either `Property_Function` or `Property_Constant` must specified, but not both.

Chapter 25

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 26

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 kernel models multi-material, incompressible flow with interface tracking. A gravitational body force is defined using the Body_Force_Density variable, and the constant density of each material phase using the PHASE property "density". The effect of small variations in density due to temperature dependence is modeled using the Boussinesqu approximation and is indicated by specifying the PHASE property "density deviation". Viscous flow requires requires the dynamic viscosity μ of each fluid material phase to be defined using the PHASE property "viscosity".

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

$$\frac{\partial H}{\partial t} = \nabla \cdot K \nabla T + Q + Q_{\text{joule}} + Q_{\text{adv}}, \tag{26.1}$$

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. The PHASE properties "density", "specific heat" and "conductivity" are used to define the density ρ, specific heat c_p and thermal conductivity K. 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

$$\frac{\partial \phi_i}{\partial t} = \nabla \cdot D_i (\nabla \phi_i \, [+S_i \nabla T]) + Q_i + Q_{i,\text{adv}}, \quad i = 1, \ldots, n, \tag{26.2}$$

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. The PHASE properties "diffusivity*i*" and "soret*i*" are used to define the diffusivity D_i and Soret coefficient 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. The PHASE properties "electrical conductivity", "electric susceptibility", and "magnetic susceptibiility" define properties relevant to the model, and 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. The kernel uses a special set of PHASE properties to define the material density as a function of temperature: "TM_Reference_Density", "TM_Reference_Temperature", and "TM_linear_CTE". The PHASE properties "Lame1" and "Lame2" are used to define the Lamé elastic constants. 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
- Legacy_Flow
- Number_of_Species
- Solid_Mechanics
- Species_Transport

Flow

Description: Enables the simulation of fluid flow.
Type: logical
Default: false

Legacy_Flow

Description: Enables the simulation of fluid flow using the original flow solver.
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 27

`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 28

`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 29

REGION Namelist

Overview

The REGION namelist defines regions of the mesh and assigns attributes to those regions. The current implementation can only define rectangular regions through upper and lower bounds. A cell is contained in the region if its centroid lies between the upper and lower bounds in every direction. The only attribute that is implemented is fluid flow solution.

REGION Namelist Features

Required/Optional: Optional
Single/Multiple Instances: Multiple

Components

- x1
- y1
- z1
- x2
- y2
- z2
- flow_off

x1

Description: Lower bound of the region in the x direction.
Physical dimension: L
Type: real
Default: 0.0
Valid values: $(-\infty, \infty)$

y1

Description: Lower bound of the region in the y direction.
Physical dimension: L

Type: `real`
Default: 0.0
Valid values: $(-\infty, \infty)$

`z1`

Description: Lower bound of the region in the z direction.
Physical dimension: `L`
Type: `real`
Default: 0.0
Valid values: $(-\infty, \infty)$

`x2`

Description: Upper bound of the region in the x direction.
Physical dimension: `L`
Type: `real`
Default: 0.0
Valid values: $(-\infty, \infty)$

`y2`

Description: Upper bound of the region in the y direction.
Physical dimension: `L`
Type: `real`
Default: 0.0
Valid values: $(-\infty, \infty)$

`z2`

Description: Upper bound of the region in the z direction.
Physical dimension: `L`
Type: `real`
Default: 0.0
Valid values: $(-\infty, \infty)$

`flow_off`

Description: Logical input describing the state of fluid flow in the region. (Set this variable to `.true.`to turn off flow in the region.)
Type: `logical`
Default: `.false.`

Chapter 30

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

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

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 31

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

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: 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.

Chapter 32

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

$$\phi_j = c \text{ on } \Gamma \tag{32.1}$$

  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

$$-D_j \nabla \phi_j \cdot \hat{n} = q \text{ on } \Gamma, \tag{32.2}$$

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

$$-D_j (\nabla \phi_j + S_j \nabla T) \cdot \hat{n} = q \text{ on } \Gamma \tag{32.3}$$

  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 33

SURFACE_TENSION Namelist

Overview

The SURFACE_TENSION namelist defines the parameters associated with the surface tension model for the dynamic interface between two fluid phases.

An additional experimental capability is available for modeling a special simplified case motivated by welding processes. In this case, the interface between liquid and gas is modeled as a fixed planar surface on the boundary of the computational domain, with the unmodeled gas outside of the domain. With a temperature-dependent surface tension coefficient, there will be tangentially directed forces at the interface (Marangoni effect) in the presence of thermal gradients, which will produce convection in the fluid. This model is enabled using the CSF_Boundary option, which is exclusive of both CSF_Normal and CSF_Tangential. The relevant parameters for this model are Bndry_Face_Set_IDs and DSig_DT.

SURFACE_TENSION Namelist Features

Required/Optional: Required when Surface_Tension is true.

Single/Multiple Instances: Single

Components

- Bndry_Face_Set_IDs
- CSF_Boundary
- CSF_Normal
- CSF_Tangential
- DSig_DT
- Interface_Materials
- Sigma_Constant
- Sigma_Function
- Smoothing_Kernel

CSF_Normal

Description: When true, the interface normal component of the surface tension will be included as a body force using the continuum surface force (CSF) method.

Type: logical

Default: false

`CSF_Tangential`

Description: When true, the interface tangential component of the surface tension will be included as a body force using the continuum surface force (CSF) method.

Type: logical

Default: false

Notes: This force (Marangoni effect) is only present when the surface tension coefficient is non-constant.

`Interface_Materials`

Description: Specifies the pair of materials that form the material interface. Materials are specified using their `Material_Number`.

Type: integer pair

Default: none

`Sigma_Constant`

Description: The constant value of the surface tension coefficient σ.

Physical dimension: `F/L`

Type: real

Default: none

Valid values: ≥ 0

Notes: Either `Sigma_Constant` or `Sigma_Function` must be specified, but not both.

`Sigma_Function`

Description: The name of a `FUNCTION` namelist that defines the surface tension coefficient σ. The function is expected to be a function of temperature T alone.

Type: string

Default: none

Notes: Either `Sigma_Function` or `Sigma_Constant` must be specified, but not both.

`Smoothing_Kernel` (Expert Parameter)

Description: Specifies the choice of smoothing kernel used in the calculation of the surface normal.

Type: string

Default: `"Rudman"`

Valid values: `"Rudman"` or `"Williams"`

`CSF_Boundary` (Experimental)

Description: When true, the special boundary tangential surface tension force described in the Overiew section will be included as a body force using the continuum surface force (CSF) method. The parameters `Bndry_Face_Set_IDs` and `DSig_DT` must also be specified. This model is exclusive of both `CSF_Normal` and `CSF_Tangential`.

Type: logical

Default: false

Notes: The current implementation of this model is subject to several restrictions. The boundary surface must be planar and orthogonal to the z coordinate direction. Further, the cells adjacent to the boundary are required to be hexehedra.

`Bndry_Face_Set_IDs` (Experimental)

Description: A list of boundary face set IDs that specify the subset of the boundary where the `CSF_-Boundary` model may be applied. Note that the model is only applied to that part of the specified boundary adjacent to a fluid phase, which may be changing in time.

Type: integer list

Default: none

Notes: The specified boundary surface must be orthogonal to the z coordinate direction in the current implementation.

`DSig_DT` (Experimental)

Description: The constant value of $d\sigma/dT$ used to compute the tangential surface tension force with the `CSF_Boundary` model.

Type: real

Default: 0.0

Notes: Both positive and negative values are possible and they yield the opposite effect; i.e., they generate opposite surface velocities due to the tangential surface tension force.

Chapter 34

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

  $$T = T_b \text{ on } \Gamma \tag{34.1}$$

  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

  $$-\kappa\nabla T \cdot \hat{n} = q_b \text{ on } \Gamma \tag{34.2}$$

  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

  $$-\kappa\nabla T \cdot \hat{n} = \alpha(T - T_\infty) \text{ on } \Gamma \tag{34.3}$$

  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

  $$-\kappa\nabla T \cdot \hat{n} = \epsilon\sigma\big((T - T_0)^4 - (T_\infty - T_0)^4\big) \text{ on } \Gamma \tag{34.4}$$

  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. When enclosure radiation systems are present, however, no boundary condition should be imposed on the part of the boundary that belongs to the enclosures unless it is a heat tranfer condition. In that case the net 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

$$-\kappa\nabla T\cdot\hat{n} = -\alpha[T] \text{ on } \Gamma, \tag{34.5}$$

  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

$$-\kappa\nabla T\cdot\hat{n} = \epsilon_\Gamma\sigma\big((T_- - T_0)^4 - (T_+ - T_0)^4\big) \text{ on } \Gamma, \tag{34.6}$$

  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

$$\epsilon_\Gamma = \frac{\epsilon_-\epsilon_+}{\epsilon_- + \epsilon_+ - \epsilon_-\epsilon_+}. \tag{34.7}$$

  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 35

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 36

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

$$\nu_t = c_\mu k^{\frac{1}{2}} l \tag{36.1}$$

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

$$k = f \cdot \tfrac{1}{2} u^2 \tag{36.2}$$

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.

A reference is needed for this model. The P&A document, where this description is taken, doesn't have one.

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 (36.1).
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 (36.2).

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 (36.1).

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 37

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

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

`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 38

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

$$\dot{\epsilon} = A \, \exp(-Q/RT) \, \sigma^n, \tag{38.1}$$

where A, n, Q, and R are parameters given by this namelist.

MTS Model. The MTS model uses the strain rate relation

$$\dot{\epsilon} = \dot{\epsilon}_{0i} \, \exp\left[-\frac{\mu b^3 g_{0i}}{kT} \left(1 - \left(\frac{\mu_0}{\mu \, \sigma_i} (\sigma - \sigma_a) \right)^{p_i} \right)^{q_i} \right], \qquad \mu = \mu_0 - \frac{D}{\exp(T_0/T) - 1} \tag{38.2}$$

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 (38.2).

Physical dimension: L

Type: real

Default: none

Valid values: > 0

MTS_d

Description: Constant D used in (38.2).

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

Type: real

Default: none

`MTS_edot_0i`

Description: Reference strain rate $\dot{\epsilon}_{0i}$ used in (38.2).

Physical dimension: T^{-1}

Type: real

Default: none

Valid values: > 0

`MTS_g_0i`

Description: Material constant g_{0i} used in (38.2).

Type: real

Default: none

Valid values: > 0

`MTS_k`

Description: Boltzmann's constant k used in (38.2).

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 (38.2).

`MTS_mu_0`

Description: Reference value μ_0 for the temperature dependent shear modulus used in (38.2).

Physical dimension: F/L^2

Type: real

Default: none

Valid values: > 0

`MTS_p_i`

Description: Exponent term p_i used in (38.2).

Type: real

Default: none

Valid values: > 0

`MTS_q_i`

Description: Exponent term q_i used in (38.2).

Type: real

Default: none

Valid values: > 0

`MTS_sig_a`

Description: The athermal stress term σ_a in (38.2).

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

Type: real

Default: none

Valid values: ≥ 0

`MTS_sig_i`

Description: A stress term σ_i related to obstacles to dislocation motion in (38.2).

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

Type: real

Default: none

Valid values: > 0

`MTS_temp_0`

Description: Constant T_0 used in the temperature dependent shear modulus equation in (38.2).

Physical dimension: Θ

Type: real

Default: none

Valid values: > 0

`Pwr_Law_A`

Description: Constant term A in (38.1).

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

Type: real

Default: none

Valid values: ≥ 0

`Pwr_Law_N`

Description: Stress exponent term n in (38.1).

Type: real

Default: none

Valid values: > 0

`Pwr_Law_Q`

Description: Activation energy Q in (38.1).

Physical dimension: E/mol

Type: real

Default: none

Valid values: ≥ 0

`Pwr_Law_R`

Description: Gas constant R in (38.1).

Physical dimension: $\mathsf{E}/(\Theta$ 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 (38.1).

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] V. R. Voller and C. Prakash. A fixed grid numerical modelling methodology for convection-diffusion mushy region phase-change problems. *International Journal of Heat and Mass Transfer*, 30(8):1709–1719, 1987.

[8] V. Venkatakrishnan. On the accuracy of limiters and convergence to steady state solutions. Technical Report 93–0880, AIAA, 1993. Presented at the 31st Aerospace Sciences Meeting.

[9] T. J. Barth and D. C. Jesperson. The design and application of upwind schemes on unstructured meshes. Technical Report AIAA–89–0366, AIAA, 1989. Presented at the 27th Aerospace Sciences Meeting and Exhibit.

[10] B. van Leer. Towards the ultimate conservative difference scheme. III. Upstream-centered finite-difference schemes for ideal compressible flow. *Journal of Computational Physics*, 23:263–275, 1977.

[11] B. van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *Journal of Computational Physics*, 23:276–299, 1977.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran*. Cambridge, 1986.

[13] 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/`.

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

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

Index