AMIE configuration guide

Alain B. Giorla

October 2, 2015

This document shows how to write initiation files for AMIE. These files can be used to set up variables in simulations without re-compiling the main executable. An example of initiation file usage in AMIE can be found in the `main_2d_composite` example.

Contents

1 Structure of the initiation file

Each line in the initiation file contains a variable for the AMIE simulation, using the following format:

```
.parameter = value # comment
```

1.1 Hierarchy

The number of leading dots before `parameter` indicates the level of the current line. A `parameter` at level n+1 is attached to the `parameter` at level n placed directly above. In the following example, `son` (level 2) is attached to `father` (level 1) and not `step_father_1` or `step_father_2` (both level 1 items as well).

```
.step_father_1
.father
..son
.step_father_2
```

The order in which items at the same level (with the same father) is not relevant. The previous example and the following are therefore equivalent:

```
.step_father_2
.step_father_1
.father
..son
```

However, if an item with the same name is defined multiple times, only the last will be accounted for.

1.2 Special characters

Several characters have specific roles that are defined later in the document, and summarized here:

- Empty spaces are ignored
- `#` indicates a comment. It can be placed at the start of a line or later. Any character after it will be ignored.
- `=` makes the separation between the parameter name and its value. In some cases, it may be preceded by an operator sign among `+`, `-`, `*`, or `/`.
- `@` indicates an input file variable. Variables can be used to ensure that several parameters always have the same values.
- `--` indicates a flag. Flags are specific token that can be activated with the command line and uses to ignore some sections of the initiation file.
- Analytic expressions must be written between quotes `"`.

1.3 Value types

Values can have different types:

- Number (using `e` to express powers of 10, as in `1.5e6` for 1.5×10^6
- Character strings (case-sensitive, spaces are ignored)

- Analytic function (defined between quotes `"`)
- Input variable (character string starting with an `@`)

Parameters may have no value at all, in which case the sign `=` must not be present on the line.

Paths and file names

Several items can be defined using additional files. These files are always defined using the path relative to the folder from which AMIE was called. Folder separators are always `"/"` even in a Windows environment. As spaces are ignored, files or folders must not include spaces in their name.

Units

S.I. units are generally used, except for times which are expressed in days. Notably, stresses are in Pascal.

1.4 First-level items

The initiation file must contain a certain number of items to be ran successfully. These items are described in more details in the following sections. Required first-level items are:

`.boundary_conditions` defines the external mechanical boundary conditions.

`.discretization` defines the parameters for the finite element mesh.

`.sample` defines the geometry and mechanical behaviour of the sample.

`.stepping` defines the time steps for the simulation.

Optional first-level items are:

`.define` defines special input variables.

`.export` defines which fields are extracted in a mesh file.

`.inclusions` defines the geometry and mechanical behaviour of the inclusions.

`.output` defines which fields are extracted in a table file.

2 Numerical parameters

This section defines how to write `.discretization` and `.stepping` items.

2.1 Spatial discretization

The `.discretization` item defines a certain number of parameters that control the mesh density and element type. It has the following structure:

```
.discretization
..sampling_number = 4
..order = LINEAR
..sampling_restriction = 0
..refinement_zone
```

These parameters are described below:

`..order` (string) defines the order of the finite element approximation. The string must be a valid finite element order. `LINEAR_TIME_LINEAR` is required for visco-elastic analysis.

`..refinement_zone` describes a section of the mesh in which the mesh density is increased. Multiple `refinement_zone` items can be defined at the same level. Each `refinement_zone` is a rectangle defined by its height, width and center as a `sample` item (without mechanical behaviour).

`..sampling_number` (number) indicates the number of mesh nodes along one side of the sample.

`..sampling_restriction` (number) is a threshold so that inclusions with less mesh points than the restriction are ignored. This parameter essentially removes the smallest inclusions from the simulation. It is optional (default value of 0: all inclusions are accounted for).

2.2 Time and damage stepping

The `.stepping` item defines a certain number of parameters required for the time-stepping as well as the damage algorithm.

Damage stepping

Three parameters control the damage stepping. They are all optional (default values are assumed if not defined).

```
.stepping
..solver_precision = 1e-8
..maximum_iterations_per_step = 256
..minimum_time_step = 1e-9
```

These parameters are described below:

`..maximum_iterations_per_step` (number) defines the maximum number of internal iterations of the damage algorithm. If the damage algorithm has not converged after that amount of steps, the results will be extracted according to the `export` and `output` first-level items and the current time step will be repeated.

`..minimum_time_step` (number) defines the smallest amount of time lapsed between two damage increments. This value is relative to the length of the current time step, so it must be lower than 1. Values lower than 0.001 are recommended.

..solver_precision (number) indicates the tolerance for the finite element resolution.

Time stepping

Option A - Fixed time step: This method allows to set a fixed time step for the entire simulation.

```
.stepping
..time_step = 1
..number_of_time_steps = 1
..logarithmic = FALSE
..first_time_step = 0.1
```

The required parameters are:

..time_step (number) defines the interval between two instants of the simulation. It must be strictly positive.

..number_of_time_steps (number) defines the number of time steps to perform. It must be strictly positive.

The optionals parameters are:

..first_time_step (number) defines the first time step of the simulation if the time step is defined as logarithmic. It must be strictly positive.

..logarithmic (boolean). If this is TRUE, then the time steps are spaced regularly in the logarithmic(10) of time, using time_step as the interval in the logarithmic scale. first_time_step is then used to define the value of the first instant of the simulation.

Option B - From an external file: This method reads the time steps from an external file.

```
.stepping
..list_of_time_steps = path_to_file
```

The required parameter is:

..list_of_time_steps (string) defines the path to the file containing the list of instants at which the simulation is carried out. The file must only contain a single column of numbers, starting with 0 and sorted by increasing value.

If both options A and B are defined, option B will be activated.

3 Sample

The sample is a rectangle defined by its width, height, center and mechanical behaviour.

```
.sample
..height = 1
..width = 1
..center
...x = 0
...y = 0
..behaviour
```

All parameters except the behaviour are optional.

`..height` (number) defines the vertical dimension of the sample.

`..width` (number) defines the horizontal dimension of the sample.

`..center` (point) defines the x and y coordinates of the center of the sample.

`..behaviour` defines the mechanical behaviour of the sample (see appropriate section).

4 Boundary conditions

The `boundary_conditions` item contains a list of `boundary_condition` sub-items:

```
.boundary_conditions
..boundary_condition
# description of the first BC
..boundary_condition
# description of the second BC
..boundary_condition
# etc
```

There is no limit to the number of sub-items in the list.

4.1 Common boundary condition parameters

All boundary condition sub-items must have the following parameters:

```
..boundary_condition
...condition = FIX_ALONG_XI
...position = TOP
...value = 0
```

In which:

`...condition` (string) defines the type of boundary condition, either in displacements (with `SET_ALONG_XI`) or in stress (with `SET_STRESS_XI`). Displacements can be also fixed to 0 with `FIX_ALONG_XI`. `XI` denotes the horizontal direction, and can be replaced with `ETA` foir the vertical direction.

`...position` (string) is the edge or corner of the sample on which the boundary condition is applied. Edges are `LEFT`, `RIGHT`, `BOTTOM` and `TOP`. Corners are combination of one vertical and horizontal edge, starting with the vertical direction, as in `BOTTOM_RIGHT`. For visco-elastic analysis, the position must also be completed with `_AFTER` (such as `TOP_AFTER`).

`...value` (number) the value of the imposed displacement or stress. Fixed displacements (such as `FIX_ALONG_XI`) do not need a value.

Without additional parameters, the boundary condition is applied to the entire edge (or corner), and the value is constant.

4.2 Spatial restriction

It is possible to apply the boundary condition on a section of an edge or a specific point instead of the entire edge. Both options are incompatible: the segment restriction has priority over the point restriction.

Option A - Segment restriction: this option defines a rectangular box. The boundary condition will be applied on points located on the edge defined by `position` and located within the box. The definition follows:

```
..boundary_condition
...restriction
....top_right
.....x = 1
.....y = 1
```

```
....bottom_left
.....x = -1
.....y = -1
```

Where `bottom_left` and `top_right` define the coordinates of the points located on the bottom-left and top-right corners of the restriction box.

Option B - Point restriction: the boundary condition will be applied on the nearest node from the specified position and located on the edge defined by `position`. This is defined with:

```
..boundary_condition
...point
....x = 0
....y = 0
```

4.3 Time evolution

It is possible to define a specific evolution of the value of a boundary condition, either using a constant rate or by defining the time evolution in an external file. Both options are incompatible: the rate-definition has priority over the external file.

Option A - Rate: the value of the boundary condition will be evaluated at each time step of the simulation as the rate multiplied by the current instant. It is defined by:

```
..boundary_condition
...rate = 1
```

Option B - Time-evolution: the value of the boundary condition will be evaluated at each time step of the simulation using an external file as input. It is defined by:

```
..boundary_condition
...interpolation = path_to_file
```

The file must contain two columns of numbers, the time in the first column, and the corresponding value in the second. The series of instants must be strictly increasing, but it does not need to correspond to the actual time steps used in the simulation. AMIE will perform a linear interpolation to get the value of the boundary condition at each time step of the simulation.