5. Input files

Mandyoc accepts ASCII files to use as initial conditions for the simulation, such as an initial temperature file `input_temperature_0.txt`, a file containing the initial interfaces geometry `interfaces.txt` and a file containing the initial velocity field at the faces of the model `input_velocity_0.txt`. The next subsections will help you creating each one of these files.

5.1. ASCII temperature file

For a 2-D grid, the initial temperature configuration can be provided as an ASCII file called `input_temperature_0.txt`. Considering x to be the longitudinal direction and y the vertical direction, the next subsections will guide you through the understanding of the initial temperature file structure.

5.1.1. 2-D temperature grid

Consider a 2-D grid where the number of nodes in the x and y directions are $nx \geq 2$ and $ny \geq 2$, respectively, with $(nx - 1)$ elements in the x direction and $(ny - 1)$ elements in the y direction. Each node in the grid can be identified with a coordinates pair (x_n, y_m), where n and m are natural numbers, and x_n and y_m are the x and y positions (in meters) of any grid node. Fig. 5.1 shows a scheme for a 2-D grid, where the red dots represent the grid nodes and every node is labeled with a (x_n, y_m) pair. The dotted lines represent any number of intermediate nodes.

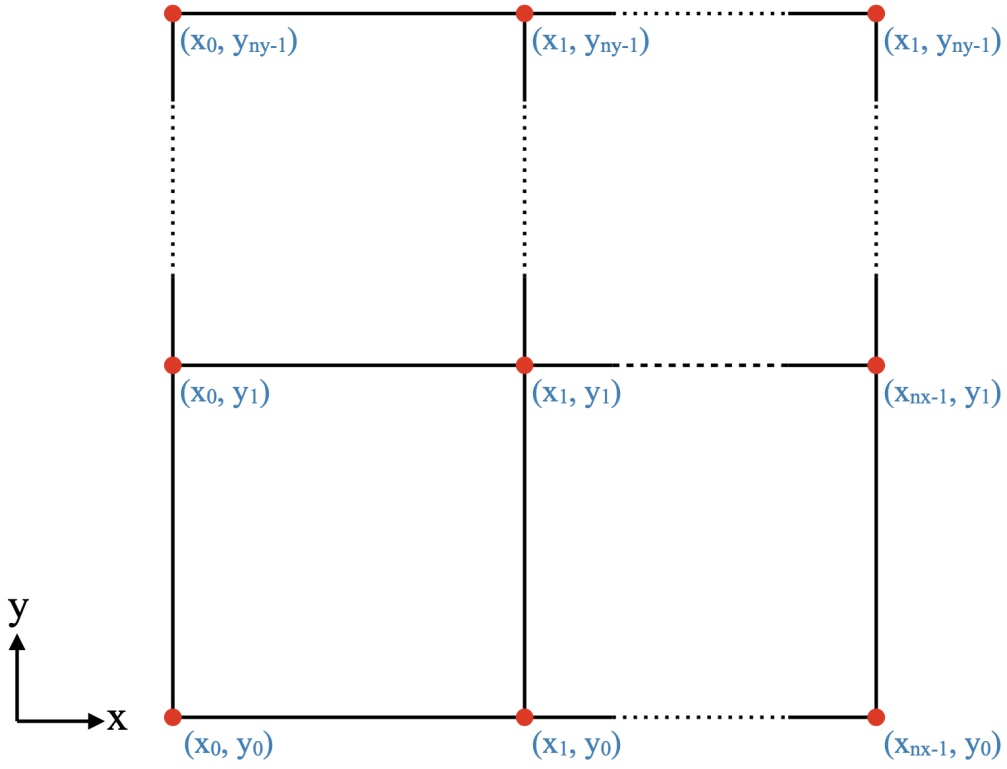

Fig. 5.1 *2-D grid scheme. The red dots represent the grid nodes where the values for the temperature must be defined, the dotted lines represent any number of grid nodes, for simplification, and the labels indicate the x and y position of each node.*

Note

Because of the way *Mandyoc* deals with the coordinates in 2-D, the node at (x_0, y_0) is always at the origin $(0, 0)$.

The example below shows how the `input_temperature_0.txt` file must be written for the 2-D grid in Fig. 5.1: after writing four lines of comments, the file must include the temperature $T(x_n, y_m)$ at each node starting at the bottom left of the model and going up in the y direction. Intuitively, the temperature is given in horizontal layers for every x_n.

```
 1 # Header line 0
 2 # Header line 1
 3 # Header line 2
 4 # Header line 3
 5 T(x_0, y_0)
 6 T(x_1, y_0)
 7 T(x_2, y_0)
 8 T(x_3, y_0)
 9 [...]
10 T(x_{nx-1}, y_0)
11 T(x_0, y_1)
12 T(x_1, y_1)
13 T(x_2, y_1)
14 T(x_3, y_1)
15 [...]
16 T(x_{nx-1}, y_1)
17 T(x_0, y_2)
18 T(x_1, y_2)
19 T(x_2, y_2)
20 T(x_3, y_2)
21 [...]
22 T(x_{nx-1}, y_2)
23 T(x_0, y_3)
24 T(x_1, y_3)
25 T(x_2, y_3)
26 T(x_3, y_3)
27 [...]
28 T(x_{nx-1}, y_3)
29 [...]
30 T(x_0, y_{ny-1})
31 T(x_1, y_{ny-1})
32 T(x_2, y_{ny-1})
33 T(x_3, y_{ny-1})
34 [...]
35 T(x_{nx-1}, y_{ny-1})
```

Tip

It is always helpful to use the four comment lines to write down any useful information for later. Because these lines are simply skipped, there is no rule to write them, the '#' symbol is used only by convention.

5.2. ASCII interfaces file

The `interfaces.txt` file, for a 2-D grid, starts with seven lines of variables that are used by the rheology models. The variables that are in the interfaces are those that might be used in Eq. 2.32 or Eq. 2.31: C, ρ_r, H, A, n, Q, and V. Each one of these variables possesses a number of values (columns) that are assigned to each lithological unit in the model. Every interface is a boundary between two lithological units, therefore the number of columns is 1 plus the number of interfaces set in the `param.txt` file (see the parameter file section), such that if there are i interfaces, $i + 1$ values for each variable **must** be provided in the `interfaces.txt` file. More will be discussed about the order of these values shortly.

5.2.1. 2-D Initial interfaces

Below, the example corresponds to a 2-D grid with two interfaces and, therefore, three lithological units. The first column contains the vertical positions **in meters** of every grid node y_m that corresponds to the **deepest** interface boundary, starting at $x = x_0$ on line 8, and ending at $x = x_{nx-1}$ on line $nx + 7$. The second column contains the vertical position of every y_m that corresponds to the second interface boundary. When defining the interfaces, it is rather common for them to "touch". Because of that, all the interfaces must be provided in a "tetris" manner, where interfaces that are collinear in parts fit the interface below.

Note

Because the interfaces are defined linearly between the nodes, it is important to define them properly, so every point inside the grid can be attributed a lithological unit.

```
 1 C    1.0    1.0    1.0
 2 rho  3000.0 3000.0 3000.0
 3 H    0.0    0.0    0.0
 4 A    0.0    0.0    0.0
 5 n    0.0    0.0    0.0
 6 Q    0.0    0.0    0.0
 7 V    0.0    0.0    0.0
 8 y_1(x_0)    y_2(x_0)
 9 y_1(x_1)    y_2(x_1)
10 y_1(x_2)    y_2(x_2)
11 y_1(x_3)    y_2(x_3)
12 y_1(x_4)    y_2(x_4)
13 y_1(x_5)    y_2(x_5)
14 y_1(x_6)    y_2(x_6)
```

The values of the variables in the first seven lines are the values of the lithological units bound by the interfaces. For the 2-D grid with two interfaces, the first values of each variable refers to the lithological unit below the first interface, the second value of each variable refers to the lithological unit between the first and second interfaces, and the third value of each variable refers to the lithological unit above the second interface. If more interfaces are added, there will be more units bounded by an upper and a lower interface.

5.3. ASCII velocities files

For a 2-D grid, the `input_velocity_0.txt` file possesses four lines of headers followed by the velocity data for each node of the model.

An initial velocity file allows an initial velocity field to be established so the simulated fluid presents an initial momentum. Respecting the conservation of mass equation when designing the initial velocity field is fundamental so the simulation can run properly.

Additionally, the velocity boundary conditions defined in the `param.txt` (see the parameter file section) will be used to define the normal and tangential velocities on the faces of the model during the simulation.

5.3.1. 2-D initial velocity

For a 2-D grid, the velocity must be defined for both x and y directions. Taking the example in the file below, the `input_velocity_0.txt` file must be written in such a way that the values for the velocity in the x direction v_x and the velocity in the y direction v_y for the node (x_0, y_0) corresponds to lines 5 and 6, respectively, the velocity values for the (x_1, y_0) corresponds to lines 7 and 8, and so on until (x_{nx-1}, y_0). Once all the nodes in the y_0 are written, the values for all x_n are inserted for y_1, and so on until y_{ny-1}.

```
 1 # Header line 0
 2 # Header line 1
 3 # Header line 2
 4 # Header line 3
 5 v_x(x_0,y_0)
 6 v_y(x_0,y_0)
 7 v_x(x_1,y_0)
 8 v_y(x_1,y_0)
 9 v_x(x_2,y_0)
10 v_y(x_2,y_0)
11 [...]
12 v_x(x_{nx-1},y_0)
13 v_y(x_{nx-1},y_0)
14 v_x(x_0,y_1)
15 v_y(x_0,y_1)
16 v_x(x_1,y_1)
17 v_y(x_1,y_1)
18 v_x(x_2,y_1)
19 v_y(x_2,y_1)
20 [...]
21 v_x(x_{nx-1},y_1)
22 v_y(x_{nx-1},y_1)
23 [...]
24 v_x(x_0,y_{ny-1})
25 v_y(x_0,y_{ny-1})
26 [...]
27 v_x(x_{nx-1},y_{ny-1})
28 v_y(x_{nx-1},y_{ny-1})
```