# SW2D-LEMON
# User Guide

Team LEMON

June 2023

# Contents

# Chapter 1

# Working environment

## 1.1 Simulation sequence

The SW2D-LEMON simulation sequence is the following.

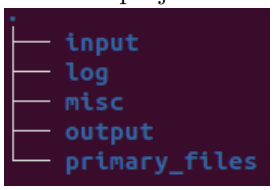
**Prior to the simulation.** It is assumed that a mesh file is readily availablre. At present, the `.2dm` file format is accepted.

**Typical simulation sequence.**

1. Process the mesh. This is done via the *sw2dConverter* executable, see Chapter 2.
2. Create the input files for the various spatially distributed model parameters (e.g. friction coefficient, porosity, etc.) This step is carried out using the *sw2dZone* executable (see Chapter 3).
3. Create the input files for time-varying series, such as rainfall, boundary condition time series. This step must be done manually. See Chapter 4 for time series file format specifications.
4. Run the simulation. This is done by starting the *sw2dSolver* executable (see Chapter 4).

## 1.2 Simulation folder structure

All SW2D-LEMON project folders must obey a specific structure, otherwise leading to runtime errors. A project folder contains at least 5 folders (see Figure)




**input** folder containing the input files. This folder is needed to run *sw2dSolver*.

**log** This folder contains the log files allowing to trace the computation process. If the folder does not exists prior to a new simulation, it is created by *sw2dSolver* or *sw2dModeler*. If the folder exists at the beginning of a simulation, the previous content is deleted.

**misc** This folder contains the simulation results in a global .res file under text format. This file is exclusively used in case of hottartse. If the folder does not exists prior to a new simulation, it is created by *sw2dSolver*. If the folder exists at the beginning of a simulation, the previous content is deleted.

**output** This folder contains the simulation results in an user-readable (text) format. If the folder does not exists prior to a new simulation, it is created by *sw2dSolver* or *sw2dModeler*. If the folder exists at the beginning of a simulation, the previous content is deleted.

**primary_files** This folder contains the files necessary to construct *geometry.sw2d*. It is compulsory for running *sw2dConverter* but unused by *sw2dModeler* and *sw2dSolver*. Since the content of this folder is not deleted by the program, the user can store complementary files.

# Chapter 2

# Using *sw2dConverter*

## 2.1 Invoking *sw2dConverter*

The *sw2dConverter* executable is used as follows.

1. Navigate to the parent folder that contains the simulation folder. The simulation folder must contain a subfolder named `primary_files` with the following files
   - (mandatory) `input.conv`
   - (mandatory) a mesh file (.2dm is the only supported format at present)
   - (optional) a Boundary Condition (BC) code file
2. Open a shell/prompt (command line window).
3. type

   ```
   sw2dConverter.exe [model name]
   ```

   This results in the creation of file `input/geometry.sw2d`.

## 2.2 File `input.conv`

### 2.2.1 General

`input.conv` is a text file specifying how the *sw2dConverter* executable is to be run form mesh processing. Each line in the file is made of a pair (keyword, value). The separator between the keyword and the value may be a space or a tabulation character. Comment lines in the file start with the character `=` or `#`.

### 2.2.2 Format description

| Field(s)/keyword | Specification |
|---|---|
| 2 comment lines | user-defined text |
| `=== Converter` | header line 3/3 |
| `overwrite` | 0 : **Q:** What happens then?<br>1 : the `input.sw2d` file will be overwritten<br>? : **Q:** Any other available options? |
| `default bc` | an integer specifying the default boundary code for all the interfaces that are not included in the boundary code file |
| `mesh_file` | name of the .2dm file (only the .2dm format is supported in this version of the code) |
| `bc_file` | name of the boundary code file |
| `========` | indicates the end of the file |

### 2.2.3 Sample file

```
= This is header line 1/2 for the input.conv sample file
# characters indicating a comment line : # or =
=== Converter
overwrite   1
default_bc  1
mesh_file   2d_hills.2dm
bc_file 2d_hills.bc
========
```

## 2.3 Mesh file format

In the current version of SW2D-LEMON , .2dm is the only supported mesh file format. More information on this format can be found at https://www.xmswiki.com/wiki/SMS:2D_Mesh_Files_*.2dm

## 2.4 Boundary code file format

### 2.4.1 Purpose

This file is used to specify the types of Boundary Conditions (BCs) to be used over the various parts of the boundary during the simulation. All boundary interfaces sharing the same boundary code (e.g. 1, or 2, or any integer number) are assigned the same BC type and numerical value at a given simulated time (e.g. boundary type 2 is a prescribed water depth $h = 3\,\mathrm{m}$ at $t = 5.0\,\mathrm{s}$). The boundary time series are specified in a different file.
*sw2dConverter* starts with assigning the default BC code (the value for `default_bc` in file `input.conv`) to all boundary interfaces. In a second step, the BC codes are modified for all the boundary interfaces listed in the BC code file.
This file is a text file that must be hand-edited by the user. It is optional. It is not needed if the boundary codes are specified directly within the .2dm file using nodestrings (a functionality of the SMS mesh generator).
If nodestrings are not used the boundary codes must be defined manually within the BC code file. This file is usually named with extension .bc to allow for faster visual identification within the folder, but the .bc extension is not mandatory.

### 2.4.2 Format specification

| Field(s) | Specification |
|---|---|
| $N$ | number of interfaces in the present file |
| $b_1 \rightharpoonup e_1 \rightharpoonup c_1$ | $b_1$ Id of the beginning node of the 1st interface<br>$e_1$ ID of the end node of the 1st interface in the file<br>$c_1$ BC code of the 1st interface<br>$\rightharpoonup$ Tabulation character or space |
| $\vdots$ | (repeat) |
| $b_N \rightharpoonup e_N \rightharpoonup c_N$ | $b_N$ Id of the beginning node of the $N$th interface<br>$e_N$ ID of the end node of the $N$th interface in the file<br>$c_N$ BC code of the $N$th interface |

Note: the BC code $c_i$ may be negative for the sake of compatibility with older SW2D-LEMON versions. Its absolute value is retained for boundary condition assignment (i.e. -2 and 2 will yield exactly the same result).

### 2.4.3 Sample file

In the following file, the first 5 boundary interfaces are assigned boundary condition type 2, while the remaining 5 are assigned boundary condition type 3.

```
10
1995 1741 -2
1741 1742 -2
1742 1743 -2
1743 1744 -2
1744 1745 -2
1745 1746 -3
1746 1747 -3
1747 1938 -3
1938 1939 -3
1939 1940 -3
```

## 2.5 Troubleshooting for *sw2dConverter*

**The geometry.sw2d file cannot be created**

- Error message: `The file does not exist .\[model_name]\input \geometry.sw2d`
- Solution: The \input folder does not exist within the directory structure. It must be created manually.

**A procedure entry point cannot be found**

- Error message: `The procedure entry point [usually a very long name] cannot be found in the dynamic link library`.
- Solution:
  1. Download the MSYS utility on your computer
  2. Navigate to the MSYS folder (usually C:\Msys64 \ucrt64 \bin
  3. Copy all the .dll files
  4. Paste them into the folder that contains the *sw2dConverter* and *sw2dSolver* executables

# Chapter 3

# Using *sw2dZone*

## 3.1 Invoking *sw2dZone*

The *sw2dSolver* executable is used to generate spatially distributed parameter files. Files `geometry.sw2d` and `input.zone` must be created before running this exuctable. It is run as follows.

1. Navigate to the parent folder of the SW2D-LEMON project folder.
2. Make sure that the `input.zone` file is present in the `primary_files` folder
3. Open a shell/prompt (command line window).
4. type

   ```
   sw2dZone.exe [model name]
   ```

5. A text file for the spatially distributed parameter is generated in the `[model_name]/input` folde. The name of this file is defined automatically as a function of the parameter names entered in `input.zone` (see Table 3.1).

## 3.2 File `input.zone`

### 3.2.1 Purpose

This file is the main file controlling the *sw2dZone* executable. Its purpose is to create spatially distributed (cell- or interface-based) parameter files to be read as inputs by *sw2dSolver*. The parameters are assumed to be uniform within zones defined by polygons.

### 3.2.2 Format description

Empty lines or lines starting with hash '#' or equal '=' signs are ignored. The spelling of the parameters is case sensitive.
This text file contains three mandatory blocks that are delimited by specific line strings.

| Block | starting line string | ending line string |
|---|---|---|
| Block 1 | `Block1: zones list BEGIN (compulsory line)` | `Block1: zones list END (compulsory line)` |
| Block 2 | `Block2: zones list BEGIN (compulsory line)` | `Block2: zones list END (compulsory line)` |
| Block 3 | `Block3: zones list BEGIN (compulsory line)` | `Block3: zones list END (compulsory line)` |

#### Block 1

Each line of Block 1 defines one polygon. For each polygon, the summits can be either identified by there coordinates or as a node of the mesh. Each line should correspond to one of the two possible formats:

- ```
  <NAME> $P$ $x_1$ $y_1$ $x_2$ $y_2$ ... $x_i$ $y_i$ ...
  ```

  where `&lt;NAME&gt;` is the name of the zone, $P$ indicates that the summit are defined by physical coordinates, and the $x_i, y_i$ couples are the coordinates of the polynomial vertices.
  ```

- `<NAME> $N$ $n_1$ $n_2$ ... $n_i$ ...`

  where `<NAME>` is the name of the zone, $P$ indicates that the summit are defined by physical coordinates, and the $n_i$ are the node identifiers of the vertices of the polynomial. The $n_i$ must therefore correspond to actual mesh node numbers.

**Block 2**

Each line of Block 2 defines the default value for one parameter. The format is the following

```
<PARAM> <DEFAULT VALUE>
```

where `<PARAM>` is the name of the parameter (see Table 3.1 for the list of allowed parameter names) and `<DEFAULT VALUE>` is the default value for this parameter. Note that

- SI units are used for all parameters
- *sw2dZone* does not check whether the value is physically meaningful

.

**Block 3**

Each line in Block 3 assigns a numerical value to a parameter within a polygon previously defined in Block 1. The format is the following

```
<PARAM> <VALUE> <NAME>
```

where `<PARAM>` is the name of the parameter (see Table 3.1, `<VALUE>` is its numerical value and `<NAME>` is the name of one of the polygons defined in Block 1.
If two or more blocks overlap, the parameter values provided in the last polygon of the list will overwrite those previously defined in the overlapping area.

### 3.2.3 Sample file

In this sample file, a parameter file is to be filled with three different parameters : h, q and v. The default values for h, q and v are 50, 0.0 and -1.0 respectively. A quadrangular polygon (called "Zone1") is defined by the coordinates of its vertices $(-1,-1),(400,-1),(400,1),(-1,1)$. Within this polygon, the parameter `h` is set to 10, while q and v are equal to their default values. A seond polygon (called "Zone2") is defined as a draft (the # character indicates that this is a comment line) using the Ids of its 7 vertices.

```
= line of comments
# symbol to comment : (# =)
Block1: zones list BEGIN (compulsory line)
Zone1 P -1 -1 400 -1 400 1 -1 1
#Zone2 N 1   2   3   4   5   8   12
Block1: zones list END (compulsory line)
Block2: default parameter value BEGIN (compulsory line)
h 50.
q 0.
v -1.
Block2: default parameter value END (compulsory line)
Block3: linking variable-zone BEGIN (compulsory line)
h 10. Zone1
Block3: linking variable-zone END (compulsory line)
```

Table 3.1: List of the allowed parameters

| Parameter name | File name | Description |
|---|---|---|
| alphaC | friction_chezy_map.txt | angle of the first principal direction for defining the Chezy coefficient with the x-axis |
| alphaK | friction_strickler_map.txt | angle of the first principal direction for defining the Strickler coefficient with the x-axis |
| alpha_mu | DIP_cell_momentum_dissipation_map.txt | |
| alphan | friction_manning_map.txt | angle of the first principal direction for defining the Manning coefficient with the x-axis |
| Beta | Boussinesq_map.txt | Boussinesq coefficient |
| Cx | friction_chezy_map.txt | component of the Chezy coefficient in the first principal direction |
| Cy | friction_chezy_map.txt | component of the Chezy coefficient in the second principal direction |
| DipPhiG | DIP_edge_porosity_map.txt | |
| DipPhiW | DIP_cell_porosity_map.txt | |
| h | initial_conditions_map.txt | water depth for the initial condition |
| Kx | friction_strickler_map.txt | component of the Strickler coefficient in the first principal direction |
| Ky | friction_strickler_map.txt | component of the Strickler coefficient in the second principal direction |
| mu1 | DIP_cell_momentum_dissipation_map.txt | |
| mu2 | DIP_cell_momentum_dissipation_map.txt | |
| nx | friction_manning_map.txt | component of the Manning coefficient in the first principal direction |
| ny | friction_manning_map.txt | component of the Manning coefficient in the second principal direction |
| PrecSta | precipitation_station_codes_map.txt | |
| q | initial_conditions_map.txt | $x$ component of the unit-discharge for the initial condition |
| r | initial_conditions_map.txt | $y$ component of the unit-discharge for the initial condition |
| SpPhiW | SP_cell_porosity_map.txt | |
| u | initial_conditions_map.txt | $x$ component of the velocity for the initial condition |
| v | initial_conditions_map.txt | $y$ component of the velocity for the initial condition |
| WindSta | wind_station_codes_map.txt | |
| z | initial_conditions_map.txt | free surface elevation for the initial condition |

# Chapter 4

# Using *sw2dSolver*

## 4.1 Invoking *sw2dSolver*

The *sw2dSolver* executable is used as follows.

1. Navigate to the parent folder of the SW2D-LEMON project folder.
2. Open a shell/prompt (command line window).
3. type

   ```
   sw2dSolver.exe [model name]
   ```

## 4.2 Minimal set of files and keywords *sw2dSolver*

Prior to running the *sw2dSolver* executable the user must make surre that the following, minimal set of files is available.

1. the `input.sw2d` simulation configuration file file exists (even if it is empty)
2. the `geometry.sw2d` geometry file
3. `initial_conditions_map.txt`
4. `hydro_boundary_time_series.txt`

Besides, if specific models such as porosity-based models are activated, additional parameters may be required (porosity maps, etc.).

## 4.3 File `input.sw2d`

### 4.3.1 Purpose

This file is the main file controlling the simulation. It defines the models that have to be used and the key parameters.

### 4.3.2 Format description

`<input.sw2d>` is a text file. It contains one keyword per line, with format

```
<KEYWD> <VALUE>
```

The order of the keywords does not matter. Empty lines or lines starting with hash '#' or equal '=' signs are considered as comment lines.
Note that

- The keywors and its value must be separated by at least one space or tabulation characters.
- The spelling of the keywords is not case sensitive.

- If a given keyword occurs several times in the file, the last read value overwrites the previously specified ones.
- All physical parameters (e.g. gravitational acceleration, etc.) are in S.I. units.

### 4.3.3 List of allowed keywords

**cflmax** {*Default value = 1*} defines the maximum permissible Courant/CFL number for timestep computation. Any strictly positive value may be specified. However, the CFL may be limited automatically to a maximum value ensuring stability depending on the **num_scheme** used (e.g. 1 for the Godunov or MUSCL-EVR scheme).

**divcorr** {*Default value = none*} specifies whether divergence correction is to be activated. Allowed values:

**none** no divergence correction

**limit_flux** prevents cell drying issues by reducing the divergence of numerical fluxes across the interfaces. This is an iterative process, requiring that the value for **nitdivmax** be set.

**limit_timestep** applies a reduction factor to the computation time step to avoid cell overdrying

**dtmap** (s) {*Default value = 10*} specifies the time interval for storing the hydrodynamic variables in the form of maps (in the **output** folder).

**dtmax** (s) {*Default value = 10*} specifies the maximum permissible computational timestep.

**dtprobes** (s) {*Default value = 100*} specifies the time interval for pointwise storage of the results (at probes). This parameter is used as soon as the file *probes_location.txt* in the input folder is not empty.

**frmax** {*Default value = 100*} is the maximum permissible value for the Froude number Fr. Should Fr exceed this value at the end of a time step, the flow velocity is reduced such that $\max ||\mathbf{v}|| = \mathrm{frmax}\sqrt{gh}$.

**gravity** ($\mathrm{ms}^{-2}$) {*Default value = 9.81*} specifies the value of the gravitational acceleration.

**hmin** (m) {*Default value* $= 10^{-5}$} is a threshold depth. The mass and momentum fluxes between two adjacent cells are computed only provided that the water depth is larger than **hmin** in at least one of the cells.

**hotstart_time** (s) {*Default value = 0*} If defined, the initial conditions are read from a previously computed ResultHydro file. This file must be copied into the input folder and will supersede the initial conditions file. The value for hotstart_time must correspond to the numerical value of the time present in the name of the ResultHydro file.

**hstop** {*Default value = false*} specifies whether the simulation should stop when the water depth becomes negative in at least one of the cells.

**false** the computation will not be stopped if a negative depth is computed in one of the cells at the end of the hyperbolic time step.

**true** the computation will be stopped as soon as the water depth is negative in at least one cell at the end of the hyperbolic time step.

Note: the occurrence of negative depths can be reduced using the divergence correction function (see **divcorr**).

**lcfl_convolution_kernel** {*Default value = default*} is used only if **num_scheme = lcfl**. Allowed values:

**const** a constant convolution kernel, $w(r) = a$, where $a$ is computed automatically.

**linear** a lineraly decreasing kernelbetween distances 0 and `lcfl_domain_size`, $w(r) = a(1 - r/D)$, where $a$ is computed automatically and $D$ is `lcfl_domain_size`.

**exponential** an exponentially decreasing convolution kernel, , $w(r) = a\exp(-(r/D)^b)$, where $a$ is computed automatically, $b$ is user-specified, and $D$ is `lcfl_domain_size`.

**polynomial** a decreasing polynomial convolution kernel, $w(r) = a(1 - (r/D)^b)$, where $a$ is computed automatically, $b$ is user-specified, and $D$ is `lcfl_domain_size`.

**lcfl_domain_size** (m) {*Default value = 0.0*} . Used only if **num_scheme = lcfl**. Size of the convolution kernel, in m . Note that the code will release an error if the default value is used because this parametr must be strictly positive. This forces the user to select a strictly positive value.

**model_building** {*Default value = none*} sets how the buildings are accounted. Allowed values:

**none** no building effect computed

**model_friction** {*Default value = none*} specifies the bottom friction model. Allowed values:

**none** no friction model

**chezy** Chezy's model is used .

**manning** the Manning formulation is used .

**strickler** friction is modelled using Strickler's model .

**model_gates** {*Default value = none*} sets how the gates are accounted. Allowed values:

**none** no gate effect computed

**model_hyperb** {*Default value = none*} Specifies the hyperbolic model . Allowed values:

**ddp** Depth-Dependent Porosity model .

**dip** Dual-Integral Porosity model. Requires the files `DIP_cell_momentum_dissipation_map.txt`, `DIP_cell_porosity_map.txt` .

**sp** Single Porosity model .

**swes** Shallow water model .

**model_infiltration** {*Default value = none*} sets how the infiltration is accounted. Allowed values:

**none** no infiltration

**model_orifice** {*Default value = none*} sets how the orifices are accounted. Allowed values:

**none** no orifice effect computed

**model_precip** {*Default value = none*} sets how the rain is accounted. Allowed values:

**none** no rain

**active** rainfall is accounted for (files `precipitation_station_codes_map.txt` and `precipitation_boundary_time_series.txt` must be provided by the user.

**model_pump** {*Default value = none*} sets how the pumps are accounted. Allowed values:

**none** no pump effect computed

**model_singheadloss** {*Default value = none*} sets how the singular headloss are accounted. Allowed values:

**none** no singular headloss computed

**model_logjam** {*Default value = none*} sets the model for log transport. Allowed values:

**none** no log transport computed

**model_transport_reaction** {*Default value = none*} Not used yet

**model_wind** {*Default value = none*} sets how the wind effect is computed. Allowed values:

**none** no wind effect computation

**smithbanke** wind effect computation using the Smith & Banke model .

**negative_depth_detection** {*Default value = 0*} activates the negative depth detection. Allowed values:

**0** no check

**1** stop the program if the water depth is negative.

**momentum_dissipation** {*Default value = 0*} defines if the momentum dissipation is accounted. Allowed values:

**0** no momentum dissipation

**1** momentum dissipation is accounted for. In this case, the file `DIP_edge_momentum_dissipation_map.txt` must be provided in the `input` folder.

**nitdivmax** {*Default value = 0*} sets the maximal number of iterations allowed in the divergence correction process. This parameter is used only if the parameter **divcorr** is set to limit_flux.

**num_scheme** defines the numerical scheme used to compute the hyperbolic part of the equation. Default and other possible values depend on the hyperbolic model:

| **model_hyperb** keyword | default value for **num_scheme** | allowed values for **num_scheme** |
|---|---|---|
| ddp | godunov | godunov |
| dip | godunov | godunov |
| sp | godunov | godunov |
| swes | godunov | godunov, muscl_evr, lcfl |

If `lcfl` is selected, the following, additional keywords are used: `lcfl_domain_size`, `lcfl_convolution_kernel`

**read_wall_from_file** {*Default value = 0*} Set how the boundaries are defined for the log transport. Allowed values:

**0** no supplementary file

**1** read transport walls from *walls_for_transport.txt*

**storeflag** {*Default value = default*} defines stored variables in hydrodynamics results maps. Allowed values:

**none** Nothing is stored in the output files

**default** The minimal sel of variables to reconstruct all other variables is stored: $x, y, z_b$, + conserved variables

**all** As default + additional variables (hydraulic head, flow velocity, Froude number, CFL, etc.)

The list of stored variables depends on the hyperbolic model.

**t0** (s) {*Default value = 0*} Starting time of the simulation.

**tmax** (s) {*Default value = 100*} Ending time of the simulation.

**verb_map_boussinesq** {*Default value = 1*} specifies if the Boussinesq coefficients map used in the computation has to be written (0: no / > 0: yes) in *log/boussinesq.txt*. Note that this option will be disregarded when model_hyperb = swes because this model does not use the Boussinesq coefficient.

**verb_map_init_cond** {*Default value = 1*} specifies if the initial conditions map used in the computation has to be written (0: no / > 0: yes) in *log/initial_conditions_map.txt*.

**verb_map_porosity** {*Default value = 1*} {*Default value = 1*} specifies if the porosity values maps used in the computation have to be written (0: no / > 0: yes). The files generated depend also on the user-defined hyperbolic model:

| **model_hyperb** keyword | files generated |
|---|---|
| ddp | *DDP_debug.txt*<br>*DDP_PhiG.txt*<br>*DDP_PhiW.txt* |
| dip | *DIP_PhiG.txt*<br>*DIP_PhiW.txt* |
| sp | *SP_PhiW.txt* |
| swes | none |

**verb_process_ddp** {*Default value = 0*} specifies the level of verbosity for all DDP model-specific computations. The bigger the value, the more verbosity you get (0 = no verbosity).

**verb_process_friction** {*Default value = 0*} specifies the level of verbosity of the friction computation function. The bigger the value, the more verbosity you get (0 = no verbosity) NOT IMPLEMENTED

**verb_process_hyperb** {*Default value = 0*} specifies the verbosity level for of the functions computing the hyperbolic part. The higher the value, the stronger the verbosity (0 = no verbosity)

**verb_process_momentum_diff** {*Default value = 0*} NOT IMPLEMENTED

**verb_process_overall** {*Default value = 0*} specifies the level of verbosity of the main workflow. The higher the value, the stronger the verbosity (0 = no verbosity).

**verb_process_velocity_corr** {*Default value = 0*} NOT IMPLEMENTED

**verb_properties_cells** {*Default value = 1*} specifies if the cells properties have to be written (0: no / > 0: yes) in *log/cells.txt*.

**verb_properties_interfaces** {*Default value = 1*} specifies if the interfaces properties have to be written (0: no / > 0: yes) in *log/interfaces.txt*.

**verb_properties_nodes** {*Default value = 1*} specifies if the nodes properties have to be written (0: no / > 0: yes) in *log/nodes.txt*.

**vmax** ($ms^{-1}$) {*Default value = 100*} specifies the maximum permissible velocity. Should $||\mathbf{v}||$ exceed this value at the end of a time step, the flow velocity is reduced such that $\max ||\mathbf{v}|| =$ **vmax**.

### 4.3.4 Example

```
= unread line
# sign to comment a full line : (# =)
=== Model
model_hyperb ddp
num_scheme godunov
tmax 100.
dtmax 1.5
dtmap 10.
dtprobes 15.
========
model_friction chezy
========
```

This chapter describes the format of the parameter files in the *input* folder.

## 4.4 Input files for spatially distributed parameters

### 4.4.1 Boussinesq_map.txt

**Purpose**

The Boussinesq momentum distribution coefficient is a cell attribute. Two options are available for specifying this coefficient:

1. $\beta$ is uniform over the entire domain, in which case providing a single value is sufficient,
2. $\beta$ is spatially variable, in which case its value must be specified on a cell-by-cell basis.

**Format description**

File format for a uniform Boussinesq coefficient. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | The value 1 for the flag indicates that $\beta$ is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `Boussinesq` | Comment line, leave unchanged |
| `beta`<br>`==== Distrib` | Uniform value<br>Section separator line, leave unchanged |

File format for a spatially variable Boussinesq coefficient. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | The zero value indicates that $\beta$ is spatially variable |
| `==== Default Param` | Section separator line, leave unchanged |
| `Boussinesq` | Comment line, leave unchanged |
| `beta` | Uniform value, will be ignored because the flag is 0 |
| `==== Distrib` | Section separator line, leave unchanged |
| `beta[i]` | $\beta$ value for each cell. Running index $i$ is the index of the cell in the mesh. One value per line. |

**Sample files**

**Example#1.** $\beta$ is uniform over the domain.

```
Unif 1
==== Default Param
Boussinesq
2.
```

**Example#2.** $\beta$ is variable over the domain, the mesh is made of 3 cells.

```
Unif 0
==== Default Param
Boussinesq
2.
==== Distrib
1.2
1.1
3.5
```

### 4.4.2 Purpose

In porosity models, part of the connected, overland water may be exchanged with buildings, that are considered as stagnant zones. The volume exchange rate between the overland and stagnant regions is proportional to the difference of free surface elevation via an exchange constant $k\_b$, as in [**?**]. The exchange law uses three parameters:

- the difference $\Delta z$ between the building basement and the overland bottom elevations
- the plan view fraction ("porosity") $\phi_b$ of space occupied by the buildings
- an exchange coefficient $k_b$ in $s^{-1}$.

**Format description**

File format for a uniform parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because the parameter set is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `delta_z <Tab> phi_b <Tab> k_b` | Comment line, leave unchanged |
| `delta_z <Tab> phi_b <Tab> k_b` | Uniform values for $\Delta z_b$, $\phi_b$ and $k_b$ |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially variable parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> flag` | Flag is 0 because the parameter set is spatially variable |
| `==== Default Param` | Section separator line, leave unchanged |
| `delta_z <Tab> phi_b <Tab> k_b` | Comment line, leave unchanged |
| `delta_z <Tab> phi_b <Tab> k_b` | Uniform values for $\Delta z_b$, $\phi_b$ and $k_b$ (will be ignored because the flag is 0) |
| `==== Distrib` | Section separator line, leave unchanged |
| `delta_z[i] <Tab> phi_b[i] <Tab> k_b[i]` | $\Delta z_b$, $\phi_b$ and $k_b$ values for each cell. Running index $i$ is the index of the cell in the mesh. |

**Sample file**

Case of a uniform parameter distribution.

```
Unif 1
==== Default Param
Delta_z Phi_b k_b
-1. 0.1 1e-3
```

### 4.4.3 DDP_cell_porosity_map.txt

**Purpose**

The Depth-Dependent Porosity (DDP) model requires that the variations in the storage porosity $\phi_\Omega$ with the water depth be specified on a cell-by-cell basis. This is the domain-based $\phi_\Omega(z)$ law in the original publication [**?**]. The code provides 5 types of pre-defined law:

- Law type -1: the porosity law is specified as a sequence of couples $(z_i, \phi_i)$, where the $z_i$ are ranked in ascending order. The $z_i$ are relative to the ground surface elevation, that is computed from the 2D mesh.
- Law type 0: the porosity law is specified by a sequence of elevations $(z_1, \ldots, z_N)$ in ascending order. The porosity is piecewise constant, equal to $i/N$ between $z_i$ and $z_i + 1$ and to unity above $z_N$. The $z_i$ are relative to the ground surface elevation as in Law type -1.
- Law type 1: the porosity is zero below an elevation $z_1$ relative to the ground elevation, and equal to $\phi_1$ above.
- Law type 2: the porosity varies linearly from $\phi_1$ to $\phi_2$ between elevations $z_1$ and $z_2$ relative to the ground surface. It is zero below $z_1$ and $\phi_2$ above $z_2$.
- Law type 3: the porosity varies linearly from $\phi_1$ to $\phi_2$ between elevations $z_1$ and $z_2$ relative to the ground surface. It is zero below $z_1$ and $\phi_3$ above $z_2$.

If the laws are spatially distributed, each cell may obey a different law type.

**Law type specification format**

Five porosity law types are allowed in the current version of the software.

| Law type | Parameters to be provided in file (separated using <Tab> characters) | Comments |
|---|---|---|
| -1 | -1 $N$ $z_1$ $\phi_1 \ldots z_N$ $\phi_N$ | $N$: number of $(z, \phi)$ couples. |
| 0 | 0 $N$ $z_1 \ldots z_N$ | $N$ : number of points in the $\phi(z)$ law. $z_i$ : elevations relative to the ground level , by ascending order. |
| 1 | 1 $z_1$ $\phi_1$ | $z_1$: altitude (relative to local bottom elevation) above which $\phi = \phi_1$; $\phi = 0$ for $z < z_1$ |
| 2 | 2 $z_1$ $z_2$ $\phi_1$ $\phi_2$ | Porosity $\phi$ is zero below $z_1$, varies linearly from $\phi_1$ to $\phi_2$ between $z_1$ and $z_2$, is equal to $\phi_2$ for $z > z_2$ |
| 3 | 3 $z_1$ $z_2$ $\phi_1$ $\phi_2$ $\phi_3$ | Porosity $\phi$ is zero below $z_1$, varies linearly from $\phi_1$ to $\phi_2$ between $z_1$ and $z_2$, is equal to $\phi_3$ for $z > z_2$ |

**Input file format**

File format for a uniform parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because the parameter set is uniform |
| `NtabMax <Tab> NtabMax` | NtabMax is the number of vertical discretization levels used for the $\phi_\Omega$(z) law |
| `==== Default Param` | Section separator line, leave unchanged |
| `Uniform parameter set` | Comment line, leave unchanged |
| Parameter set | The parameter set should obey the format given in the previous table |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because the parameter set is spatially variable |
| `NtabMax <Tab> NtabMax` | NtabMax is the number of vertical discretization levels used for the $\phi_\Omega$(z) law |
| `==== Default Param` | Section separator line, leave unchanged |
| `Uniform parameter set` | Comment line, leave unchanged |
| Parameter set | The parameter set should obey the format given in the previous table |
| `==== Distrib` | Section separator line, leave unchanged |
| Parameter set [i] | Parameter set, provided on a cell-by-cell basis. Index i runs sequentially from 1 to the number of cells. The parameter set should obey the format given in the Law type specification format table |

**Sample files**

**Example#1: Uniform law type 2** The porosity varies linearly from $\phi_1 = 0.$ to $\phi_2 = 1.$ between $z_1 = 3$ m and $z_2 = 10$ m.

```
Unif 1
NtabMax 50
==== Default Param
Type Parameters(type)
2 3. 10. 0. 1.
==== Distrib
```

**Example#2: Distributed law type** There are 3 cells in the model, the law types for cells 1, 2 and 3 are respectively 1, 0 and -1. The law type 0 in Cell 2 is given by 5 elevations. The Law type -1 in Cell 3 uses 2 $(z, \phi)$ couples.

```
Unif 0
NtabMax 50
```

```
==== Default Param
Type Parameters(type)
2 3. 10. 0. 1.
==== Distrib
Type Parameters(type)
1   2.3 0.5
0   5   0.1 0.4 0.7 1.2 2.5
-1  2   0.1 0.3 2.7 0.95
```

### 4.4.4 DDP_edge_porosity_map.txt

#### Purpose

The Depth-Dependent Porosity (DDP) model requires that the variations in the connectivity porosity $\phi_\Gamma$ with the water depth be specified on an interface-per-interface basis. This is the domain-based $\phi_\Gamma(z)$ law in the original publication [**?**]. The code provides 5 types of pre-defined law. They are described in Subsection 4.4.3.

#### Format description

The format is exactly the same as that of the cell porosity map, see 4.4.3. The only difference is that the running index i goes from 1 to the number of interfaces (not the number of cells).

#### Sample files

See Subsection 4.4.3.

### 4.4.5 DIP_cell_momentum_dissipation_map.txt

#### Purpose

The Dual Integral Porosity (DIP) model requires that the momentum dissipation tensor $\mu$ be specified on a cell-by-cell basis [**?**]. This tensor is characterised by the momentum disspation coefficients $(\mu_1, \mu_2)$ in two orthogonal principal directions and the angle $\alpha_\mu$ between the 1st principal direction and the $x-$ axis.

#### Input file format

File format for a uniform tensor. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because the tensor is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `mu1 mu2 alphamu` | Comment line, leave unchanged |
| $\mu_1$ $\mu_2$ $\alpha_\mu$ | The uniform values for $(\mu_1, \mu_2, \alpha_\mu)$ |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because the tensor is spatially distributed |
| `==== Default Param` | Section separator line, leave unchanged |
| `PhiW` | Comment line, leave unchanged |
| `dummy dummy dummy` | These 3 values are ignored because `Flag` = 1 |
| `==== Distrib` | Section separator line, leave unchanged |
| $\mu_1$ $\mu_2$ $\alpha_\mu$ | $(\mu_1, \mu_2, \alpha_\mu)_i$, provided on a cell-by-cell basis (1 line per cell). Index $i$ runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform dissipation tensor** Momentum dissipation coefficients in the principal directions are 0.25 and 0.10 respectively. The angle of the 1st principal direction with the $x-$ axis is 10 degrees.

```
Unif 1
==== Default Param
mu1 mu2 alpha_mu
0.25 0.10 10.0
==== Distrib
```

**Example#2: Distributed storage porosity** There are 3 cells in the model. Cells 1 to 3 have the same dissipation coefficients, but the angle of the 1st principal direction increases from 10 to 30 degrees from Cell 1 to Cell 3.

```
Unif 0
==== Default Param
PhiW
0. 0. 0.
==== Distrib
0.25 0.10 10.0
0.25 0.10 20.0
0.25 0.10 30.0
```

### 4.4.6 DIP_cell_porosity_map.txt

**Purpose**

The Dual Integral Porosity (DIP) model requires that the storage porosity $\phi_\Omega$ be specified on a cell-by-cell basis [**?**].

**Input file format**

File format for a uniform storage porosity. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because $\phi_\Omega$ is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `PhiW` | Comment line, leave unchanged |
| `Phi_Omega` | The uniform $\phi_\Omega$ value |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because $\phi_\Omega$ is spatially variable |
| `==== Default Param` | Section separator line, leave unchanged |
| `PhiW` | Comment line, leave unchanged |
| `Phi_Omega` | This value will be ignored because `Flag` = 1 |
| `==== Distrib` | Section separator line, leave unchanged |
| `Phi_Omega [i]` | Storage porosity $\phi_\Omega$, provided on a cell-by-cell basis. Index i runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform storage porosity** The porosity is identical for all cells, equal to 0.5.

```
Unif 1
==== Default Param
PhiW
```

```
0.5
==== Distrib
```

**Example#2: Distributed storage porosity** There are 3 cells in the model, cells 1 to 3 have storage porosities of 0.5, 0.7 and 0.1 respectively.

```
Unif 0
==== Default Param
PhiW
0.5
==== Distrib
0.5
0.7
0.1
```

### 4.4.7 DIP_edge_porosity_map.txt

#### Purpose

The Dual Integral Porosity(DIP) model requires that the variations in the connectivity porosity $\phi_\Gamma$ be specified on an interface-per-interface basis [**?**].

#### Format description

The format is exactly the same as that of the cell porosity map, see 4.4.6. The only difference is that the running index i goes from 1 to the number of interfaces (not the number of cells).

#### Sample files

See Subsection 4.4.6.

### 4.4.8 friction_chezy_map.txt

#### Purpose

If Chezy's law is selected for the friction model, the Chezy coefficient must be provided on a cell-by-cell basis. The Chezy may be anisotropic. Therefore, it is a tensor. It is specified in the form $(C_1, C_2, \alpha_C)$, where $C_1$ and $C_2$ are the friction coefficients in the two principal directions of the horizontal plane, and $\alpha_C$ is the angle of the first principal direction with the $x-$axis.

#### Input file format

File format for a uniform Chezy tensor $C$. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because $C$ is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `Chezy_1 Chezy_2 alpha` | Comment line, leave unchanged |
| `Chezy_1 <Tab> Chezy_2 <Tab> alpha` | The uniform values for $C_1$, $C_2$ and $\alpha_C$ ($\alpha_C$ in degrees) |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because $C$ is spatially variable |
| `==== Default Param` | Section separator line, leave unchanged |
| `Chezy_1 Chezy_2 alpha` | Comment line, leave unchanged |
| `Chezy_1 Chezy_2 alpha` | This value will be ignored because `Flag` = 1 |
| `==== Distrib` | Section separator line, leave unchanged |
| `Chezy_1[i] <Tab> Chezy_2[i] <Tab> alpha[i]` | Values for $C_1$, $C_2$ and $\alpha_C$ ($\alpha_C$ in degrees), provided on a cell-by-cell basis. Index i runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform Chezy tensor** The Chezy tensor is uniform for all cells. It is respectively 60 and 70 in the 1st and 2nd principal directions. The 1st principal direction makes a 45 degree angle with the $x-$axis.

```
Unif 1
==== Default Param
Chezy_1 Chezy_2 alpha
60. 70. 45.
==== Distrib
```

**Example#2: Distributed Chezy tensor** There are 3 cells in the model. The Chezy is respectively 60 and 70 in the 1st and 2nd principal directions. Its angle with the $x-$axis is respectively 0, 10 and 30 degrees in Cells 1, 2 and 3.

```
Unif 0
==== Default Param
Chezy_1 Chezy_2 alpha
60. 70. 45.
==== Distrib
60. 70. 0.
60. 70. 10.
60. 70. 30.
```

### 4.4.9 friction_manning_map.txt

**Purpose**

If Manning's law is selected for the friction model, the Manning coefficient must be provided on a cell-by-cell basis. The Manning may be anisotropic. Therefore, it is a tensor. It is specified in the form $(n_{M_1}, n_{M_2}, \alpha_{n_M})$, where $n_{M_1}$ and $n_{M_2}$ are the friction coefficients in the two principal directions of the horizontal plane, and $\alpha_{n_M}$ is the angle of the first principal direction with the $x-$axis.

**Input file format**

File format for a uniform Manning tensor. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because $n_M$ is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `Manning_1 Manning_2 alpha` | Comment line, leave unchanged |
| `Manning_1 <Tab> Manning_2 <Tab> alpha` | The uniform values for $n_{M_1}$, $n_{M_2}$ and $\alpha_{n_M}$ ($\alpha_{n_M}$ in degrees) |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed Manning tensor. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because $n_M$ is spatially variable |
| `==== Default Param` | Section separator line, leave unchanged |
| `Manning_1 Manning_2 alpha` | Comment line, leave unchanged |
| `Manning_1 Manning_2 alpha` | This value will be ignored because `Flag` $= 1$ |
| `==== Distrib` | Section separator line, leave unchanged |
| `Manning_1[i] <Tab> Manning_2[i] <Tab> alpha[i]` | Values for $n_{M_1}$, $n_{M_2}$ and $\alpha_{n_M}$ ($\alpha_{n_M}$ in degrees), provided on a cell-by-cell basis. Index i runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform Manning tensor** The Manning tensor is uniform for all cells. It is respectively 0.025 and 0.02 in the 1st and 2nd principal directions. The 1st principal direction makes a 45 degree angle with the $x-$axis.

```
Unif 1
==== Default Param
Manning_1 Manning_2 alpha
0.025    0.02 45.
==== Distrib
```

**Example#2: Distributed Manning tensor** There are 3 cells in the model. The Manning is respectively 0.025 and 0.2 in the 1st and 2nd principal directions. Its angle with the $x-$axis is respectively 0, 10 and 30 degrees in Cells 1, 2 and 3.

```
Unif 0
==== Default Param
Manning_1 Manning_2 alpha
0.025    0.02 45.
==== Distrib
0.025    0.02 0.
0.025    0.02 10.
0.025    0.02 30.
```

### 4.4.10 friction_strickler_map.txt

**Purpose**

If Strickler's law is selected for the friction model, the Strickler coefficient must be provided on a cell-by-cell basis. The Strickler may be anisotropic. Therefore, it is a tensor. It is specified in the form $(K_1, K_2, \alpha_K)$, where $K_1$ and $K_2$ are the friction coefficients in the two principal directions of the horizontal plane, and $\alpha_K$ is the angle of the first principal direction with the $x-$axis.

**Input file format**

File format for a uniform Strickler tensor. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because $K$ is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `Strickler_1 Strickler_2 alpha` | Comment line, leave unchanged |
| `Strickler_1 <Tab> Strickler_2 <Tab> alpha` | The uniform values for $K_1$, $K_2$ and $\alpha_K$ ($\alpha_K$ in degrees) |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed Strickler tensor. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because $K$ is spatially variable |
| `==== Default Param` | Section separator line, leave unchanged |
| `Strickler_1 Strickler_2 alpha` | Comment line, leave unchanged |
| `Strickler_1 Strickler_2 alpha` | This value will be ignored because `Flag` $= 1$ |
| `==== Distrib` | Section separator line, leave unchanged |
| `Strickler_1[i] <Tab> Strickler_2[i] <Tab> alpha[i]` | Values for $K_1$, $K_2$ and $\alpha_K$ ($\alpha_K$ in degrees), provided on a cell-by-cell basis. Index `i` runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform Strickler tensor** The Strickler tensor is uniform for all cells. It is respectively 40 and 50 in the 1st and 2nd principal directions. The 1st principal direction makes a 45 degree angle with the $x-$axis.

```
Unif 1
==== Default Param
Strickler_1 Strickler_2 alpha
40. 50. 45.
==== Distrib
```

**Example#2: Distributed Strickler tensor** There are 3 cells in the model. The Strickler is respectively 40 and 50 in the 1st and 2nd principal directions. Its angle with the $x-$axis is respectively 0, 10 and 30 degrees in Cells 1, 2 and 3.

```
Unif 0
==== Default Param
Strickler_1 Strickler_2 alpha
40. 50. 45.
==== Distrib
40. 50. 0.
40. 50. 10.
50. 50. 30.
```

### 4.4.11 Infiltration_map.txt

**Purpose**

**Format description**

**Sample file**

### 4.4.12 Precipitation_station_codes_map.txt

**Purpose**

**Input file format**

File format for a uniform station code. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because there is only one wind station for the entire domain (uniform station map) |
| `==== Default Param` | Section separator line, leave unchanged |
| `Sta` | Comment line, leave unchanged |
| `Sta` | The (uniform) precipitation station code to be used over the entire domain |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed wind station code. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because there is more than one wind station for the entire domain |
| `==== Default Param` | Section separator line, leave unchanged |
| `Sta` | Comment line, leave unchanged |
| `Sta` | This value will be ignored because `Flag` is zero |
| `==== Distrib` | Section separator line, leave unchanged |
| `Sta[i]` | Precipitation station codes, provided on a cell-by-cell basis. Index i runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform wind station code** In this example, the station code is uniformly 1 over the entire domain.

```
Unif 1
==== Default Param
Xind
1
==== Distrib
```

**Example#2: Distributed wind station code** In this example, there are three cells in the domain. Cells 1 and 3 have station code 1, cell 2 has station code 4.

```
Unif 0
==== Default Param
Xind
1
==== Distrib
1
4
1
```

### 4.4.13 SP_cell_porosity_map.txt

**Purpose**

The depth-independent Single Porosity (DIP) [**?**] model requires that the porosity $\phi$ be specified on a cell-by-cell basis.

**Input file format**

File format for a uniform porosity. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because $\phi$ is uniform |
| `==== Default Param` | Section separator line, leave unchanged |
| `Phi` | Comment line, leave unchanged |
| `Phi` | The uniform $\phi$ value |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed parameter set. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because $\phi$ is spatially variable |
| `==== Default Param` | Section separator line, leave unchanged |
| `PhiW` | Comment line, leave unchanged |
| `Phi` | This value will be ignored because `Flag` = 1 |
| `==== Distrib` | Section separator line, leave unchanged |
| `Phi [i]` | Storage porosity $\phi$, provided on a cell-by-cell basis. Index i runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform porosity** The porosity is identical for all cells, equal to 0.5.

```
Unif 1
==== Default Param
PhiW
0.5
==== Distrib
```

**Example#2: Distributed porosity** There are 3 cells in the model, cells 1 to 3 have porosities of 0.5, 0.7 and 0.1 respectively.

```
Unif 0
==== Default Param
PhiW
0.5
==== Distrib
0.5
0.7
0.1
```

### 4.4.14 singular_head_loss_map.txt

**Purpose**

**Format description**

**Sample file**

### 4.4.15 wind_station_codes_map.txt

**Purpose**

SW2D takes as an input a wind forcing that is variable in both space and time. At a given time, the wind is taken uniform over zones (also called "stations") within the computational domain. The shape and extension of these zones remain constant over the entire simulation. The purpose of the file `wind_stations_codes_map.txt` is to specify the spatial extension of these zones on a cell-by-cell basis. The stations are assigned integer numbers, that are not necessarily consecutive. Non-consecutive station numbering allows maximum flexibility. Assume for instance the user is willing to make a scenario analysis using two different time series over the same area. In the first simulation, Station code 1 will be assigned do the entire domain. In the second run, Station code 2 will be assigned to all cells. This does not require that the compete time series be changed.

**Input file format**

File format for a uniform station code. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 1` | Flag is 1 because there is only one wind station for the entire domain (uniform station map) |
| `==== Default Param` | Section separator line, leave unchanged |
| `Xind` | Comment line, leave unchanged |
| `Xind` | The (uniform) wind station code to be used over the entire domain |
| `==== Distrib` | Section separator line, leave unchanged |

File format for a spatially distributed wind station code. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `Unif <Tab> 0` | Flag is 0 because there is more than one wind station for the entire domain |
| `==== Default Param` | Section separator line, leave unchanged |
| `Xind` | Comment line, leave unchanged |
| `Xind` | This value will be ignored because `Flag` is zero |
| `==== Distrib` | Section separator line, leave unchanged |
| `Xind[i]` | Wind station codes, provided on a cell-by-cell basis. Index i runs sequentially from 1 to the number of cells. |

**Sample files**

**Example#1: Uniform wind station code** In this example, the station code is uniformly 1 over the entire domain.

```
Unif 1
==== Default Param
Xind
1
==== Distrib
```

**Example#2: Distributed wind station code** In this example, there are three cells in the domain. Cells 1 and 3 have station code 1, cell 2 has station code 4.

```
Unif 0
==== Default Param
Xind
1
==== Distrib
1
4
1
```

## 4.5 Input files for time series

### 4.5.1 Hydro_boundary_time_series.txt

**Purpose**

This file is made to prescribe the boundary conditions for hydrodynamics (number, type and values). These conditions come in addition with the default one, corresponding to a wall condition $q = 0$.

**Input file format**

| Field(s) | Comment(s) |
|---|---|
| `#Comment` | Comment line, could be changed but not removed |
| `#bcs` | Comment line, could be changed but not removed |
| `n` | Integer $\geq 0$, number of (non default) boundary conditions |
| `#` | Comment line, could be changed but not removed |
| `#` | Comment line, could be changed but not removed |
| `Type_1 <Tab> (...) Type_n` | Field types for each of the $n$ boundary conditions. Separated by tabs.<br>Fr (–) prescribed outflowing Froude value<br>h (m) prescribed water depth<br>q ($m^2s^{-1}$) prescribed unit discharge (positive if inflowing)<br>z (m) prescribed free surface elevation |
| `#` | Comment line, could be changed but not removed |
| `#` | Comment line, could be changed but not removed |
| `Time_0 Value_1 <Tab> (...) Value_n` | Time value, then field values for each of the $n$ boundary conditions. Separated by tabs. |
| `Time_1 Value_1 <Tab> (...) Value_n` | Time value, then field values for each of the $n$ boundary conditions. Separated by tabs. |

Note that

- Type Fr: this boundary should always correspond to outflowing conditions, otherwise leading to instability.

- Types q and u: a positive numerical value corresponds to inflowing conditions. In this case, the flow velocity vector is assumed to be locally orthogonal to the boundary. If the numerical value is negative (outflowing condition), the transverse flow velocity is advected outside the computational domain.

**Sample files**

**Example #1: wall boundary condition everywhere** Here, no additional boundary condition is implemented ($n = 0$). All boundaries will be processed with default ($q = 0$) condition. Note that line #6 is empty and no line is needed at the end since no boundary condition is declared.

```
\#
\#bcs
0
#
#

#
#
```

**Example #2: constant water depth with time-increasing discharge** Here we have $n = 2$ boundary conditions: $h = 0.5$ (stationnary condition since both values are identical in lines 9-10) and $q$ changes from 0 to 1 between $t = 0$ and $t = 100$ (with interpolation in between, *i.e.* $q = 0.3$ for $t = 30$).

```
#
#bcs
2
#
#
h   q
#
#
0  0.5     0
100 0.5     1
```

### 4.5.2 precipitation_time_series.txt

**Purpose**

SW2D takes as an input a precipitation field that is variable in both space and time. At a given time, the precipitation rate is taken uniform over zones (also called "stations") within the computational domain. The shape and extension of these zones remain constant over the entire simulation. The purpose of the file `precipitation_time_series.txt` is to specify the variations of the precipitation rate with time on a station-by-station basis. The stations are assigned integer numbers, that are not necessarily consecutive.

**Warning.** Assume that $n$ and $N$ are respectively the smallest and largest (integer) station numbers in a given station code map (See Section **??**). Then, although there are only $N-n+1$ "active" stations in the model, the time series file should incorporate the time series for $N$ stations.

**Format description**

File format. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `#Precipitation time series` | Comment/header line |
| `#Precipitation stations` | Comment/header line |
| `N` | Number of stations in the time series file |
| `#` | Section separator line, leave unchanged |
| `#t` + $N$ times (`<Tab> P`) | Comment line, leave unchanged |
| `t` + $N$ times (`<Tab> P`) | `t` : time in seconds ; `P`: precipitation rate in m/s |

**Sample file**

There are 3 stations in the model.

```
#Precipitation time series for mr test
#wind stations
3
#
# t P   P   P
0  1e-6    0e0   1e-5
300 1e-6    1e-5    0e0
360 0e0     0e0     0e0
```

### 4.5.3 wind_time_series.txt

**Purpose**

SW2D takes as an input a wind forcing that is variable in both space and time. At a given time, the wind is taken spatitally uniform over zones (also called "stations") within the computational domain. The shape and extension of these zones remain constant over the entire simulation. The purpose of the file `wind_time_series.txt` is to specify the variations of the wind speed with time on a station-by-station basis. The stations are assigned integer numbers, that are not necessarily consecutive. The file `wind_time_series.txt` can be generated using the *sw2dZone* executable (see Chapter 3).

**Warning.** Assume that $n$ and $N$ are respectively the smallest and largest (integer) station numbers in a given station code map (See Section 4.4.15). Then, although there are only $N - n + 1$ "active" stations in the model, the time series file should incorporate the time series for Stations 1 to $N$.

**Format description**

File format. `Bold` : keyword, must appear as such in the file. `<Tab>`: tabulation character.

| Field(s) | Comment(s) |
|---|---|
| `#Wind time series` | Comment/header line |
| `#Wind stations` | Comment/header line |
| `N` | Number of stations in the time series file |
| `#` | Section separator line, leave unchanged |
| `#t \|u\| angle` | Comment line, leave unchanged |
| `t` , $N$ couples `( <Tab> u_norm <Tab> u_angle)` | `t` : time (s) ; `u_norm`: wind speed (m/s) ; `u_angle` (degrees): direction the wind comes from, counted positive clockwise from North |

**Sample file**

```
#Wind time series for static wind test
#wind stations
1
#
# t |u| angle
0  0.5 -90
1e6 0.5 -90
```

## 4.6 Miscellaneous input files

### 4.6.1 probes_location.txt

**Purpose**

This file specifies the locations at which the flow variables are to be stored at the pre-defined **dtprobes** time interval.

**Format description**

| Field(s) | Format |
|---|---|
| `x y name` | This is the header line. The string `x y name` must be inserted without modification |
| $x_1 y_1$ $\texttt{name}_1$ | $x$ and $y$ coordintates (in m) of the 1st probe and its name, separated by at least one space character |
| ... | |
| $x_N y_N$ $\text{textttname}_1$ | $x$ and $y$ coordintates (in m) of the $N$th probe and its name, separated by at least one space character |

**Example**

In the file hereafter, there are 4 probes

```
x y name
50 0 probe_1
150 0 probe_2
350 0 probe_3
450 0 probe_4
```