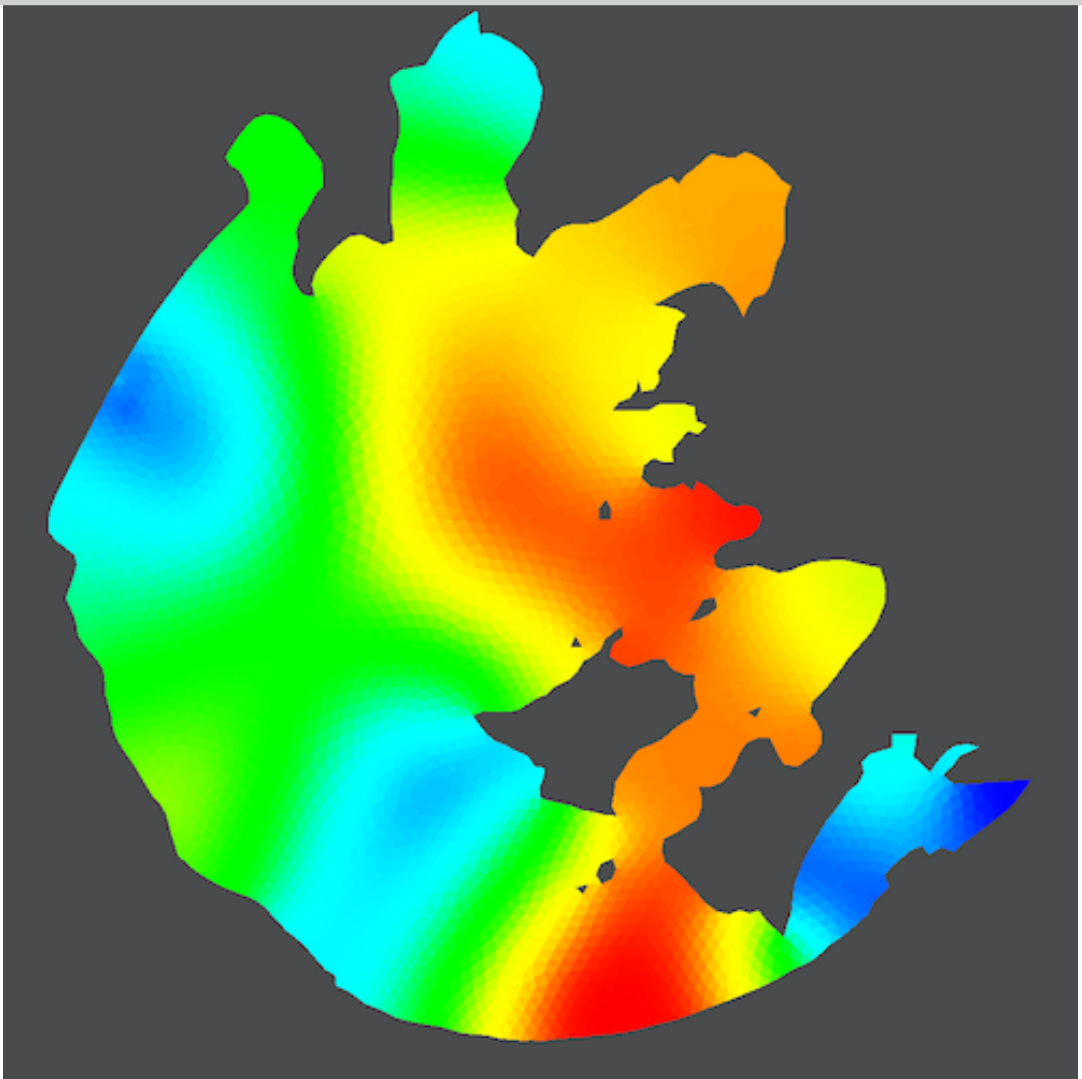
SW2D - Manual 2.1.0

Quick-start, User and Developer Guides

Team LEMON

May 2022



Contents

PART I

Quick Start Guide

CHAPTER 1

Overview

1.1 About This Guide

This *Quick Start* guide is a brief introduction to SW2D intended to help you run a first simulation to insure that every requirements are full-filled prior to running your own complex simulation. For more details on physical and numerical models, together with detailed specifications of input/output data, the reader is referred to the User Guide (see Part II).

1.2 Product Overview

SW2D (Shallow Water 2D) is a C++ package dedicated to shallow water modeling and includes additional features such as porosity modeling (upscaling).
The software can be installed thanks to dedicated binary files provided for Linux, MacOS and Windows operating systems. SW2D contains a numerical solver that can be used in command line and/or a graphic user interface in which numerical simulations can be configured, run and plotted.

1.3 About the SW2D Developer Team

LEMON is a research team between Inria Sophia-Antipolis Méditerranée, Hydrosciences Montpellier (HSM) and Institut Montpelliérain Alexander Grothendieck (IMAG). It is an interdisciplinary team working on the design and implementation of accurate and computationally inexpensive models for natural processes occurring in the littoral area. We develop theoretical and numerical tools (both deterministic and stochastic) to model physical processes that occur in the coastal region, both inland and in the sea. We see the shoreline as the natural interface between various environments: sea, sandy bottoms, urban coastal areas, river deltas, lagoons, etc. Our objective is to build and improve models to simulate those systems and to couple them (together or with external data) in order to produce a global coastal risk management system that better accounts for the return period, variety and intensity of natural phenomena.

1.4 Licensing

This software has been developed by researchers of LEMON project team of Inria (Institut National de Recherche en Informatique et Automatique) and UM (University of Montpellier).

Inria, Domaine de Voluceau, Rocquencourt - BP 105
78153 Le Chesnay Cedex, FRANCE

Université de Montpellier, 163 rue Auguste Broussonnet
34090 Montpellier, FRANCE

Inria and UM (hereinafter named "the Consortium") hold all the ownership rights on the Software. The Software has been registered with the Agence pour la Protection des Programmes (APP) under IDDN.FR.001.430019.001.S.P.2019.000.31235

Every user of the Software could communicate to the developers of SW2D (amdt-sw2d@inria.fr) his or her remarks as to the use of the Software.

THE USER CANNOT USE, EXPLOIT OR COMMERCIALLY DISTRIBUTE THE SOFTWARE WITHOUT PRIOR AND EXPLICIT CONSENT OF THE CONSORTIUM. ANY SUCH ACTION WILL CONSTITUTE A FORGERY.
THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT ANY WARRANTIES OF ANY NATURE AND ANY EXPRESS OR IMPLIED WARRANTIES,WITH REGARDS TO COMMERCIAL USE, PROFESSIONAL USE, LEGAL OR NOT, OR OTHER, OR COMMERCIALISATION OR ADAPTATION.
UNLESS EXPLICITLY PROVIDED BY LAW, IN NO EVENT, SHALL THE CONSORTIUM OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA, OR PROFITS OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Public Research License

The Software is still under development. It is one Consortium's aim to widely disseminate the Software in the scientific community and collect feedback enabling regular improvements.
For these reasons the Consortium has decided to distribute the Software.
The Consortium grants to the academic user, a free of charge, without right to sublicense non-exclusive right to use the Software for research purposes for a period of one (1) year from the date of the download of the compiled code. Any other use without of prior consent of the Consortium is prohibited.
The academic user explicitly acknowledges having received from the Consortium all information allowing him to appreciate the adequacy between the Software and his needs and to undertake all necessary precautions for his execution and use.
The Software is provided only as a compiled library file and decompiling process are strictly unauthorized.

If results obtained through the use of the Software were to be published, authors should cite the Software taking information from the following webpage: SW2D publication page.

Educational License

To be completed.

Private License

To be completed.

CHAPTER 2

Before You Install

2.1 Understand System Requirements

SW2D is made to be used on all Linux/MacOS/Windows machines. Most 64-bits processors will be compatible with the install binaries provided by the developper team. For specific configuration, please contact the technical team.

2.2 The SW2D Structure In A Nutshell

The SW2D software is based on several programs (see Figure 2.1). Geometrical pre-processing is separated from the main computation process to avoid to reprocess the geometry each time the hydrodynamics variables are modified by the user. The computation can be realised either in a GUI or in a solver mode (with no GUI). The current version of SW2D requires a complementary software to produce mesh and the parameter files. AR:En dire plus.

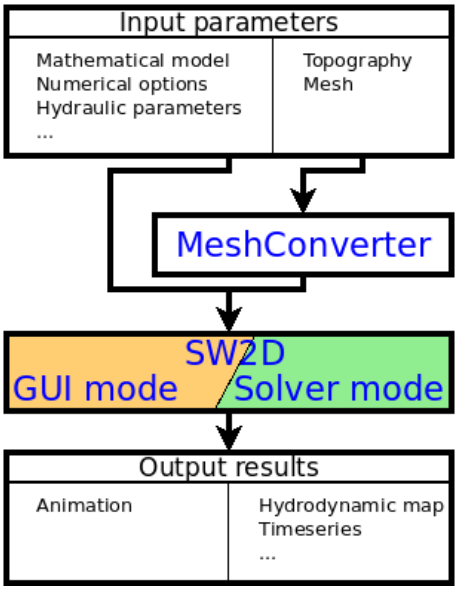


Figure 2.1: Workflow of a typical SW2D computation

fig:layout

CHAPTER 3

Install SW2D from binaries

3.1 Linux and MacOS

Installation from binary files is not available for MacOS and Linux under this version of the code. Users of these operating systems are invited to contact the developer team directly in order to obtain a direct SW2D download link.

3.2 Windows

..

Binary files for Windows operating systems, containing both SW2D and starting examples, can be found on the SW2D webpage.
Follow the following steps to get SW2D installed on your machine:

1. Locate the .exe file on the SW2D webpage and download it.
2. Locate and double-click the downloaded .exe file. (It will usually be in your Downloads folder.)
3. A dialog box will appear. Follow the instructions to install the software.

fig:install-windows-1

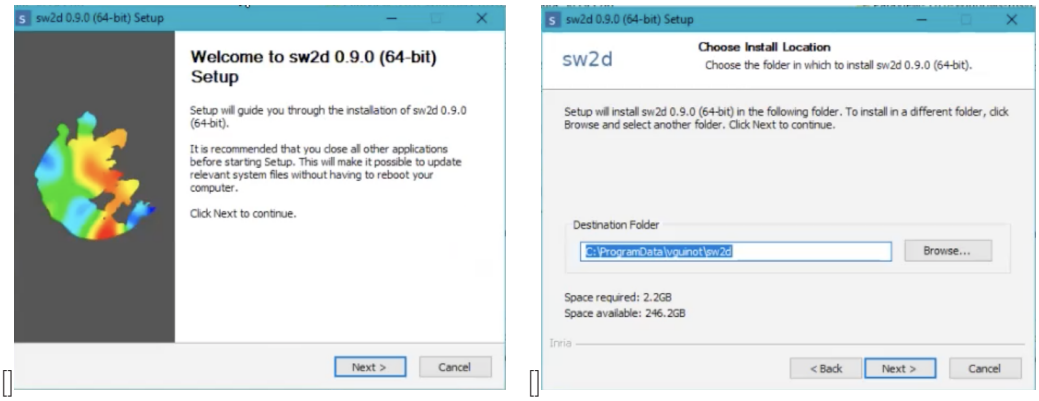


Figure 3.1: Installation process under windows: first steps

fig:install-windows-start

4. The software will be installed. For more convenience, it is recommended to add icons on your personal desktop.

Remark 3.2.1. Installation requires the program to be stored in a hidden directory on your computer. If necessary, you can view the hidden files and folders on your computer by following the procedure on the Microsoft Support site.

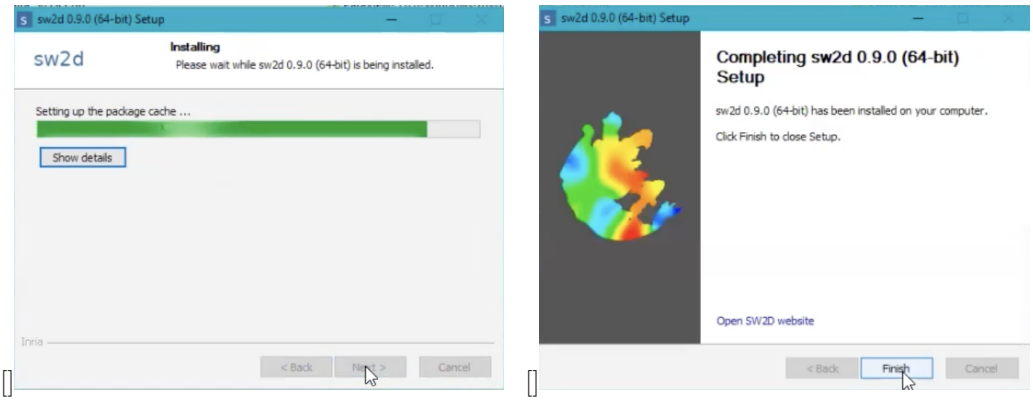


fig:install-windows-3

Figure 3.2: Installation process under windows: last steps

fig:install-windows-end

CHAPTER 4

First Examples

Together with the SW2D binary files (solver and GUI), the installer also provides several examples that can be immediately computed.
This guide presents, for each example, the input files provided and the model illustrated. Results and model physical information can be found on the dedicated SW2D webpage.

Folder structure

The **examples** folder includes:

- folder **converter**: contains the examples for *sw2dConverter*
- folder **solver**: contains the examples for *sw2dSolver*
- **check_all.sh**: script checking that all the examples are run successfully.
- **README.txt**

4.1 *sw2dConverter* examples

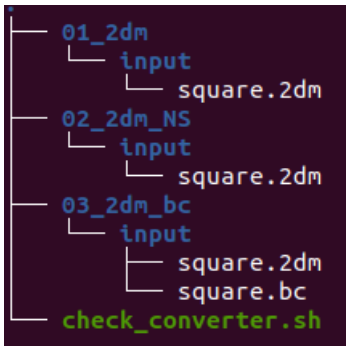


Figure 4.1: Initial structure of the **converter** folder

fig: tree_converter_examples

All the *sw2dConverter* examples aim to produce the *geometry.sw2d*. The *sw2dConverter* run only in command line.

Test	Input file(s)	Command
01_2dm	square.2dm	*sw2dConverter input/square.2dm –force -d -1*
02_2dm_NS	square.2dm	*sw2dConverter input/square.2dm –force -d -2*
03_2dm_bc	square.2dm square.bc	*sw2dConverter input/square.2dm –force -d -2 -b input/square.bc*

4.2 *sw2dSolver* examples

The **solver** folder contains the following structure:
AR:Ecrire ici comment on lance le code et ensuite comment ça impacte sur la structure.

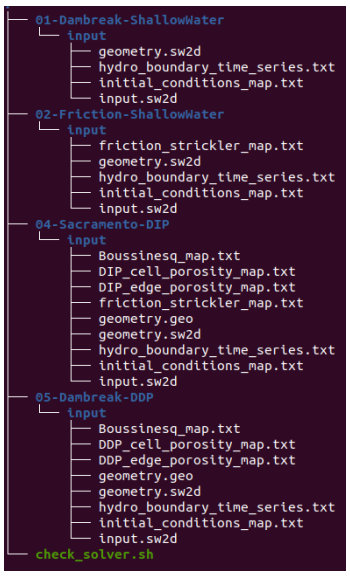


Figure 4.2: Initial structure of the **solver** folder

fig:tree_solver_examples

01-Dambreak-ShallowWater

This example reproduces a dambreak simulation in a rectangular channel using the classical shallow water model without friction.

input folder contents

file name	description
1D_Mesh.bc	description of the boundary type. This file is only used to generate the *geometry.geo* file.
1D_Mesh_zb0.2dm	mesh. This file is only used to generate the *geometry.geo* file.
geometry.geo	geometry file
hydro_boundary_time_series.txt	boundary condition time series
initial_conditions_map.txt	spatial distribution of the initial condition
input.sw2d	input file

Table 4.1: 01-Dambreak-ShallowWater example: description of the *input* folder contents

tab:sw_dambreak_InputFiles

expected output

file name	description
ResultBoundariesFlux.txt	time series of the discharge crossing each type of boundary.
ResultHydro_XXX.csvPFG:get the correct name	hydrodynamic results at time XXX. XXX is the time from the beginning of the simulation using the ISO...PFG:complete format.

Table 4.2: 01-Dambreak-ShallowWater example: description of the *output* results

tab:sw_dambreak_OutputFiles

02-Friction-ShallowWater

This example reproduces flow in a rectangular channel with friction using the classical shallow water model. It converges to a steady uniform flow in mild slope with a downstream boundary condition.

input folder contents

file name	description
friction_strickler_map.txt	spatial distribution of the Strickler coefficient (uniformly set to $K = 30\mathrm{m}^{1/3}.\mathrm{s}^{-1}$.
hydro_boundary_time_series.txt	boundary condition time series
initial_conditions_map.txt	spatial distribution of the initial condition
input.sw2d	input file
mesh.2dm	mesh. This file is only used to generate the *geometry.geo* file.

Table 4.3: 02-Friction-ShallowWater example: description of the *input* folder contents

tab:sw_friction_InputFiles

expected output

file name	description
ResultBoundariesFlux.txt	time series of the discharge crossing each type of boundary.
ResultHydro_XXX.csvPFG:get the correct name	hydrodynamic results at time XXX. XXX is the time from the beginning of the simulation using the ISO...PFG:complete format.

Table 4.4: 02-Friction-ShallowWater example: description of the *output* results

tab:sw_friction_OutputFiles

04-Sacramento-DIP

This example simulates the flood propagation due to a dike failure at Sacramento using the Dual Integral Porosity (DIP) model[1].

input folder contents

file name	description
Boussinesq_map.txt	spatial distribution of the Boussinesq coefficient (uniformly set to 1).
DIP_cell_porosity_map.txt	spatial distribution of the storage porosity in cells (DIP model) (uniformly set to $\phi_{omega} = 0.5$)
DIP_edge_porosity_map.txt	spatial distribution of the conveyance porosity along interfaces (DIP model) (uniformly set to $\phi_{gamma} = 0.25$)
friction_strickler_map.txt	spatial distribution of the Strickler coefficient (uniformly set to $K = 60\mathrm{m}^{1/3}.\mathrm{s}^{-1}$.
geometry.geo	geometry file
hydro_boundary_time_series.txt	boundary condition time series
initial_conditions_map.txt	spatial distribution of the initial condition
input.sw2d	input file

Table 4.5: 04-Sacramento-DIP example: description of the *input* folder contents

tab: dip_sacramento_InputFiles

expected output

file name	description
ResultBoundariesFlux.txt	time series of the discharge crossing each type of boundary.
ResultHydro_XXX.csvPFG:get the correct name	hydrodynamic results at time XXX. XXX is the time from the beginning of the simulation using the ISO...PFG:complete format.

Table 4.6: 04-Sacramento-DIP example: description of the *output* results

tab: dip_sacramento_OutputFiles

[1] Guinot V., Sanders B. F. & Schubert J. E. (2017). Dual integral porosity shallow water model for urban flood modelling. Advances in Water Resources 103, 16–31. https://doi.org/10.1016/j.advwatres.2017.02.009

05-Dambreak-DDP

This example reproduce a dambreak simulation in a triangular channel using the Depth-Dependant Porosity (DDP) model[2].

input folder contents

file name	description
Boussinesq_map.txt	spatial distribution of the Boussinesq coefficient
DDP_cell_porosity_map.txt	spatial distribution of the storage porosity in cells (DDP model)
DDP_edge_porosity_map.txt	spatial distribution of the conveyance porosity along interfaces (DDP model)
geometry.2dm	mesh. This file is only used to generate the *geometry.geo* file.
geometry.bc	description of the boundary type. This file is only used to generate the *geometry.geo* file.
geometry.geo	geometry file
hydro_boundary_time_series.txt	boundary condition time series
initial_conditions_map.txt	spatial distribution of the initial condition
input.sw2d	input file

Table 4.7: 05-Dambreak-DDP example: description of the *input* folder contents

tab: ddp_dambreak_InputFiles

expected output

file name	description
ResultBoundariesFlux.txt	time series of the discharge crossing each type of boundary.
ResultHydro_XXX.csvPFG:get the correct name	hydrodynamic results at time XXX. XXX is the time from the beginning of the simulation using the ISO...PFG:complete format.

Table 4.8: 05-Dambreak-DDP example: description of the *output* results

tab: ddp_dambreak_OutputFiles

[2]Guinot V., Delenne C., Rousseau A. & Boutron O. (2018). Flux closures and source term models for shallow water models with depth-dependent integral porosity. Advances in Water Resources 109, 133–157. https://doi.org/10.1016/j.advwatres.2018.09.014

PART II

User Guide

part:userguide

CHAPTER 5

Physical and Numerical Models

The interested reader is referred to the following reference:

Steinstraesser et al., *SW2D: a new software for upscaled shallow water modeling*, SimHydro 2021: Models for complex and global water issues, 2021. [Permanent link]

CHAPTER 6

Boundary conditions

The boundary edges, are always oriented so that there is no cell on the left hand side of the edge and a cell on the right hand side.
Looking for the fluxes (F1 and F2) through the boundary.
The parameters and the state variables are fully known: q_{in}, h_{in} or z_{in} (depending on the considered model).

$$u_{in} = q_{in}/h_{in} \tag{6.1a}$$
$$h_{in} = \max(z_{in} - z_{\min}, 0.) \tag{6.1b}$$

where $z_{\min}$ is the minimum bottom elevation on the edge.

6.1 time interpolation

The boundary time series are defined in the file hydro_boundary_time_series.txt for hydrologic conditions

$$bc\,(t) = \tag{6.2}$$

For the sake of simplicity, $bc\,(t)$ is simplified in bc.

6.2 f-type boundary condition

purpose

The f-type boundary condition produces the same fluxes as in the inner cell.
Such Dirichlet boundary condition is encountered in case of uniform flow. However, the definition of an uniform flow is valid at the scale of a fully wet cross-section that is not always coherent with a 2D discretization of the model. Moreover, the uniform flow condition is hydraulically coherent at the downstream (respectively upstream) boundary in case of mild (respectively steep) slope. The effect of such boundary condition in any other hydraulic configuration is unpredictable but without leading to any numerical problem.

formulation

SW and SP model

$$F_1 = q_{in} \tag{6.3a}$$
$$F_2 = q_{in}u_{in} + gh_{in}^2/2 \tag{6.3b}$$

DIP model

$$F_1 = q_{in} \tag{6.4a}$$
$$F_2 = q_{in}u_{in}Phi_R/Phi_G + gh_{in}^2/2 \tag{6.4b}$$

DDP model

$$
\begin{aligned}
F_1 &= \theta_\omega u_{in} && (6.5a)\\
F_2 &= \theta_\omega u_{in}^2 + pressure_force && (6.5b)\\
pressure_force &= pressureForceOmegaOfZ(celR, z_{in}) && (6.5c)
\end{aligned}
$$

6.3 c-type boundary condition

purpose

The c-type boundary condition imposes the Froude number value at the boundary. This corresponds to a depth-velocity relationship that can be seen as a classical rating curve for a rectangular weir. This boundary condition is compulsorily an outlet thus the boundary velocity is negative: $u_{bc} \leq 0$. The user defined boundary value bc is assumed to be positive. PFG:what happens if bc<0??

formulation

SW model

The Riemann invariant $u - 2c$ along the wave travelling from the inner cell to the boundary is used to determine the value of the pressure wave celerity c_{BC} on the boundary:

$$c_{bc} = \frac{u_{in} - 2 * c_{in}}{-bc - 2} \tag{6.6}$$

The minus in front of bc appears as the boundary velocity u_{bc} is negative whereas bc is positive. For supercritical flow with $\mathrm{Fr}_{in} > 2$ (thus entering the domain), the pressure wave speed at the boundary becomes negative $c_{bc} < 0$. As this is physically incoherent, c_{bc} is bounded to 0.
The variables at the boundary are computed as: $u_{bc} = -bc * c_{bc}$ and $h_{bc} = c_{bc}^2/g$.

$$
\begin{aligned}
F1 &= u * h && (6.7a)\\
F2 &= specificForce(uR, hR) + (F1 - q1) * L1 && (6.7b)
\end{aligned}
$$

PFG:what if inner cell torrentiel sortant?

CHAPTER 7

Running the code

7.1 *sw2dConverter*

The *sw2dSolver* requires a geometry defined in a single file *input/geometry.sw2d*. This file includes all the geometry properties and the BC codes for the boundary edges.
The *sw2dConverter* aims to produce *input/geometry.sw2d* from various type of inputs. The different commands are the following (each option can be activated with a single letter (-k) or with a keyword (–keyword)PFG:on ne voit pas les symboles "–"; for the sake of clarity, only the 'letter way' is presented here):

***sw2dConverter*input/2DMFILE -d -X** a single 2dm file (*input/2DMFILE*) is use to specify all the geometry. The BC codes are set by the nodestring numbers. For the boundary edges for which there is no nodestring, the BC code is set to X. X have to be positive.

***sw2dConverter*input/2DMFILE -b input/BCFILE -d -X** a single 2dm file (*input/2DMFILE*) is use to specify the cells and nodes. The BC codes are defined (in the 'old way') in the bc file (*input/BCFILE*). The nodestrings specified in the 2dm file are simply ignored. For the boundary edges that are not in the bc file (*input/BCFILE*), the BC code is set to X. X have to be positive.

***sw2dConverter*input/GEOFILE** a single geo file (*input/GEOFILE*) is used to specify all the geometry. When importing a geometry file, no other option can be used.

- Whatever the command used, *sw2dConverter* will raise an error message and stop if a file *input/geometry.sw2d* already exists. To write over the existing file, the argument -f can be used.
- the BCs with a code "0" (either specified from the .2dm, the .bc or the -d option) leads to no computation of the fluxes at this edge:

$$F_1 = 0$$
$$F_2 = h_{cell}u_{cell}^2 + 0.5gh_{cell}^2 + (F_1 - h_{cell}u_{cell})\max(u_{cell} + c_{cell}, 0)$$

PFG:on ne voit pas les symboles "–"

letter	keyword	description
-b	–bc	specify the input boundary file
-d	–default	Compulsory option; specify the default BC code
-f	–force	overwrite on existing file *input/geometry.sw2d*

7.2 *sw2dModeler*

The *sw2dModeler* allows to run the simulation using an user-interface (GUI). In the current version, a simulation **cannot** be created from the GUI (See section 7.2).
The interface can be launched by:

on windows double-clicking on the PFG:ADD the correct name shortcut on the desktop or with the PFG:ADD the correct name shortcut in the *Start Menu*

on Unix running the *./sw2dModeler*command line

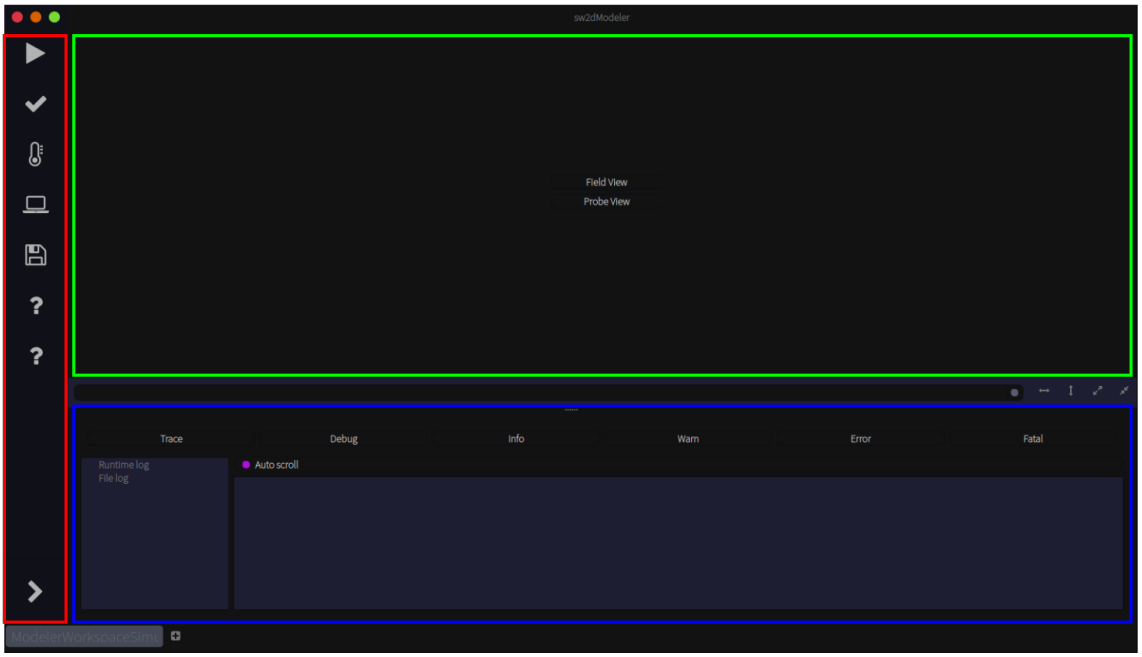


Figure 7.1: SW2D User interface

fig:sw2dGUI

on MacOS PFG:Carole, Antoine: à completer

The interface is presented on figure 7.1 and includes three parts:

red rectangle control zone (see 7.2)

green rectangle view zone (see 7.2)

blue rectangle log message zone (see 7.2)

subsec: controlZone

Control zone

The control zone contains the buttons activating the different menus to control the simulation.

subsubsec: StartSim

Start Simulation

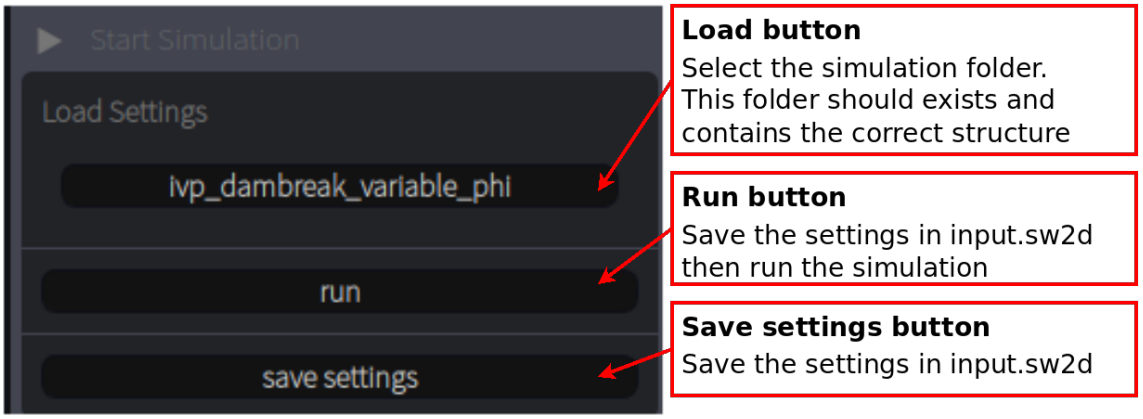


Figure 7.2: Start Simulation menu

fig:my_label

Prior to open a simulation folder, the user should insure (and create if necessary) that an *input* subfolder exists and contains an *input.sw2d* file (even empty) (see the simulation folder structure in PFG:add ref). Depending on the type of simulation, other files may be required but in any case, the GUI will be able to start and display for the missing files PFG:check this.

 **Board Activators**

This menu allows to show or hide the corresponding panels (PFG:list the activators).

 **Physical Parameters**

 **Simulation Parameters**

 **Output Parameters**

 **Help Menu**

 **Configuration**

subsec:viewZone

View zone

This zone allows to display the simulation results, either as a *Field View* (*i.e.* a spatial representation at a given time) or a *Probe View* (*i.e.* a temporal evolution at a given location).

subsec:
messageZone

Log message zone

The different messages are prompted in the zone. Each message is associated to a category allowing for an easy sort (using the buttons *Trace*, *Debug*, *Info*, *Warn*, *Error*, *Fatal*).
When the *Auto scroll* tick-box is activated, the the message zone automatically scroll to display the last message. For simulation with a lot of verbosity this option is known to slow down (or even freeze) the GUI refresh.

7.3 *sw2dSolver*

The *sw2dSolver* is run in command line:

sw2dSolver -o1 -o2...

where o1 and o2 are the computational options. Some options have to be followed by a user-defined value. Each option can be activated with a single letter (-k) or with a keyword (–keyword)PFG:on ne voit pas les symboles "–".

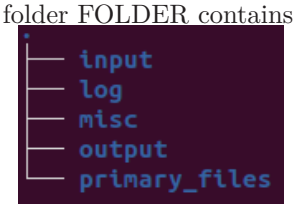
PFG:on ne voit pas les symboles "–"

letter	keyword	description
-f <FOLDER>	–folder <FOLDER>	specifies the simulation folder <FOLDER>
-h	–help	displays all the available command
-n	–norun	the simulation only performs the initialisation process and stops just before the computational part
-v	–version	displays the version number

CHAPTER 8

Simulation structure

Each SW2D simulation is organized on a file/folder structure that has to be respected. A simulation folder FOLDER contains at least 5 folders:



input folder containing the input files. This folder is compulsory 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 binary format (not readable by user). These files are used mainly by the GUI for the visualisation. 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.

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 9

Input File Format

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

9.1 Boussinesq_map.txt

Purpose

The Boussinesq momentum distribution coefficient (denoted by β in the governing equations) accounts for non-uniform flow velocity profiles in the computation of the momentum flux over a cross-section Γ

$$\beta = \frac{\overline{u^2}}{\overline{u}^2} \geq 1 \tag{9.1}$$

TODO : move the equation to physicalmodel.tex where the overbar denotes the average over a flow cross-section. The condition $\beta \geq 1$ is necessary for the preservation of model hyperbolicity and solution well-posedness.

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

1. β is uniform over the entire domain, in which case providing a single value is sufficient,
2. β 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 β is uniform
==== Default Param	Section separator line, leave unchanged
Boussinesq	Comment line, leave unchanged
beta	Uniform value
==== Distrib	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 β 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]	β value for each cell. Running index i is the index of the cell in the mesh. One value per line.

Sample files

Example#1. β is uniform over the domain.

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

Example#2. β 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
```

9.2 Buildings_exchange_map.txt

TODO: to be modified if new model versions

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 [Gui12]. The exchange law uses three parameters:

- the difference Δz between the building basement and the overland bottom elevations
- the plan view fraction ("porosity") ϕ_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 Δz_b, ϕ_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 Δz_b, ϕ_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]	Δz_b, ϕ_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
```

9.3 DDP_cell_porosity_map.txt

sec_Model!Hyperbolic!DDP!cellPorosityMap

Purpose

The Depth-Dependent Porosity (DDP) model requires that the variations in the storage porosity ϕ_Ω 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 [Gui+18]. The code provides 5 types of pre-defined law:

- Law type -1: the porosity law is specified as a sequence of couples (z_i, ϕ_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 ϕ_1 above.
- Law type 2: the porosity varies linearly from ϕ_1 to ϕ_2 between elevations z_1 and z_2 relative to the ground surface. It is zero below z_1 and ϕ_2 above z_2.
- Law type 3: the porosity varies linearly from ϕ_1 to ϕ_2 between elevations z_1 and z_2 relative to the ground surface. It is zero below z_1 and ϕ_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 $\phi_1 \ldots z_N$ ϕ_N	N: number of (z, ϕ) 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 ϕ_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 ϕ_1 ϕ_2	Porosity ϕ is zero below z_1, varies linearly from ϕ_1 to ϕ_2 between z_1 and z_2, is equal to ϕ_2 for $z > z_2$
3	3 z_1 z_2 ϕ_1 ϕ_2 ϕ_3	Porosity ϕ is zero below z_1, varies linearly from ϕ_1 to ϕ_2 between z_1 and z_2, is equal to ϕ_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 ϕ_Ω(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 ϕ_Ω(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, ϕ) 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
```

9.4 DDP_edge_porosity_map.txt

sec_Model!Hyperbolic!DDP!edgePorosityMap

Purpose

The Depth-Dependent Porosity (DDP) model requires that the variations in the connectivity porosity ϕ_Γ 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 [Gui+18]. The code provides 5 types of pre-defined law. They are described in Subsection 9.3.

Format description

The format is exactly the same as that of the cell porosity map, see 9.3. 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 9.3.

9.5 DIP_cell_porosity_map.txt

sec_Model!Hyperbolic!DIP!cellPorosityMap

Purpose

The Dual Integral Porosity (DIP) model requires that the storage porosity ϕ_Ω be specified on a cell-by-cell basis [GBJ17].

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 ϕ_Ω is uniform
==== Default Param	Section separator line, leave unchanged
PhiW	Comment line, leave unchanged
Phi_Omega	The uniform ϕ_Ω 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 ϕ_Ω 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 ϕ_Ω, 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
```

9.6 DIP_edge_porosity_map.txt

sec_Model!Hyperbolic!DIP!edgePorosityMap

Purpose

The Dual Integral Porosity(DIP) model requires that the variations in the connectivity porosity ϕ_Γ be specified on an interface-per-interface basis [GBJ17].

Format description

The format is exactly the same as that of the cell porosity map, see 9.5. 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 9.5.

9.7 SP_cell_porosity_map.txt

sec_Model!
Hyperbolic!SP!
cellPorosityMap

Purpose

The depth-independent Single Porosity (DIP) [GS06] model requires that the porosity ϕ 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 ϕ is uniform
==== Default Param	Section separator line, leave unchanged
Phi	Comment line, leave unchanged
Phi	The uniform ϕ 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 ϕ 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 ϕ, 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
```

9.8 extraction_times_list.txt

Purpose

This file contains a line for each extra result to display at chosen times that may be different from the storing times defined by **dtstor** parameter.
This option is activated by the presence of the file in the input folder (no specific parameter is required in the *input.sw2d*).
The results are stored in new output files (one per chosen time) with the same format as results stored with the **dtstor** timestep.

Format description

Each line contains the time in seconds from the beginning of the simulation, at which the results extraction is expected, following by a chosen name for this result.
Times do not need to be ordered in the file.
Lines starting with a hash are ignored as well as empty lines or space only lines. If two lines give the same time, the second one is ignored.
Note that the name following the time is mandatory.

Example

PFG:Répertoire d'exemple: mettre ce fichier dans distrib/file_template/input/... With the following input file

```
#
# some extractions times for specific data extractions
#
3  extract\_time\_3

10 time\_10

3     extract\_time\_1
#20  extract\_time\_20
```

the code will generate 2 output files with names:

```
ResultsTime_p0000d00m03s000_extract_time_3.csv
ResultsTime_p0000d00m10s000_time_10.csv
```

Note that the real extraction time is the nearest time computed according to the current timestep and may be different from the chosen time.

9.9 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, α_C), where C_1 and C_2 are the friction coefficients in the two principal directions of the horizontal plane, and α_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 α_C (α_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 α_C (α_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.
```

9.10 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 α_{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 α_{n_M} (α_{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 α_{n_M} (α_{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.
```

9.11 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, α_K), where K_1 and K_2 are the friction coefficients in the two principal directions of the horizontal plane, and α_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 α_K (α_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 α_K (α_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.
```

9.12 Hydro_boundary_time_series.txt

Purpose

Format description

Example

9.13 Infiltration_map.txt

Purpose

Format description

Example

9.14 initial_conditions_map.txt

sec_initialConditionsMap

Purpose

The well-posedness of the shallow water problem requires initial conditions and boundary conditions be provided. The present section deals with the specification of initial conditions.
The initial conditions may be either uniform or spatially distributed. They are provided on a cell-by-cell basis. Two types of variable must be provided : the water depth or free surface elevation, and the flow velocity or unit discharge components.

Format description

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

Field(s)	Comment(s)
Unif <Tab> **1**	Flag is 1 because the initial conditions are uniform
==== Default Param	Section separator line, leave unchanged
z or **h** <Tab> **u** or **q** <Tab> **v** or **r**	**h**: the first numerical value in the record is the water depth **z**: the first numerical value in the record is the surface elevation **u**: the second numerical value in the record is the $x-$ flow velocity **q**: the second numerical value in the record is the $x-$ unit discharge **v**: the third numerical value in the record is the $y-$ flow velocity **r**: the third numerical value in the record is the $y-$ unit discharge
z or h <Tab> u or q <Tab> v or r	The three numerical values fr the uniform initial condition
==== Distrib	Section separator line, leave unchanged

File format for a spatially variable initial condition. **Bold** : keyword, must appear as such in the file. <Tab>: tabulation character.

Field(s)	Comment(s)
Unif <Tab> **1**	Flag is 0 because the initial conditions are spatially variable
==== Default Param	Section separator line, leave unchanged
Val_1 <Tab> Val_2 <Tab> Val_3	These 3 numerical values will be ignored because Flag = 0
==== Distrib	Section separator line, leave unchanged
z[i] or **h**[i] <Tab> **u**[i] or **q**[i] <Tab> **v**[i] or **r**[i]	**h**: the first numerical value in the record is the water depth **z**: the first numerical value in the record is the surface elevation **u**: the second numerical value in the record is the $x-$ flow velocity **q**: the second numerical value in the record is the $x-$ unit discharge **v**: the third numerical value in the record is the $y-$ flow velocity **r**: the third numerical value in the record is the $y-$ unit discharge
z[i] or h[i] <Tab> u[i] or q[i] <Tab> v[i] or r[i]	Initial conditions for cell i, index i running sequentially from 1 to the number of cells

Sample files

Example #1: uniform initial conditions (elevation-velocity)

The initial condition is uniform. The free surface elevation (1 m) and the flow velocity (0 m/s in both directions of space) are the same in all cells.

```
Unif 1
==== Default Param
z u v
1 0 0
==== Distrib
```

Example #2: uniform initial conditions (depth-unit discharge)

The initial condition is uniform. The water depth (2 m) and the unit discharge (1 m^2/s in the $x-$direction, 0 m^2/s in the $y-$direction) are the same in all cells.

```
Unif 1
==== Default Param
h q r
2 1 0
==== Distrib
```

Example #3: distributed initial conditions

There are 3 cells in the model. In Cell#1, the initial free surface elevation is 10m, the flow velocity is zero. In Cell#2, the free surface elevation is 10m, the $x-$velocity is 1 m/s and the $y-$velocity is zero. In the third cell, the free surface elevation is 5 m, and the initial flow velocity is zero.

```
Unif 0
==== Default Param
z u v
1 0 0
==== Distrib
10. 0.  0.
10. 1.  0.
5.  0.  0.
```

9.15 input.sw2d

Purpose

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

Format description

This text file contains a parameter per line. The order of the parameters do not matter. Empty lines or lines starting with hash '#' or equal '=' signs are ignored.
The spelling of the parameters is not case sensitive.
If the same parameter is defined several times in the *input.sw2d*, SW2D will consider the last provided value.
Known parameters are listed hereafter: PFG:sort by name

2dmfile is a parameter naming the 2dm file to be used in the geometry settings.PFG:to be adapted once the 2dm-Geo converter will be ready

crmax *{Default value = 1}* defines the maximal Courant value to compute the timestep. Any positive value is allowed but the algorithm will automatically prevent large values, known to lead to instability issues (typically $Cr_{max} > 1$ for an explicit scheme).

divcorr *{Default value = none}* parameter to activate the divergence correction (see section PFG:add reference). Allowed values:

none no divergence correction

limit_flux prevents divergence by reducing the computing fluxes through the interfaces. The flux reduction implies an iterative process. The parameter **nit_div_max** have to be definePFG:really or default value??? For sure, nit_div_max cannot be negative. Is there a coeff for the flux reduction?

limit_timestep prevents divergence by reducing the computational time-step

dtmap *{Default value = 10}* specifies the interval of time to store the hydrodynamics results map (see **??**PFG:add reference to ResultHydro...)

dtmax *{Default value = 10}* specifies the maximal computational timestep

dtprobes *{Default value = 100}* specifies the time window to store the probes results (see section**??**PFG:add reference to ResultProbes...). This parameter is used only if results at the probes location is stored (this is simply activated filling *probes_location.txt* in the input folder) - see section 9.18.

gravity *{Default value = 9.81}* specifies the value of the gravitational acceleration.

hmin *{Default value* $= 10^{-5}$*}* represents the minimal water depth value considered by the Riemann solvers. When the water depth is below this value, the corresponding (cell-computed) unit discharge is neglected.

model_friction *{Default value = none}* sets how the basement friction effect is reproduced. Allowed values:

none no friction effect computed

chezy friction effect computed using the Chezy formulation (see section **??**PFG:add ref).

manning friction effect computed using the Manning formulation (see section **??**PFG:add ref).

strickler friction effect computed using the Strickler formulation (see section **??**PFG:add ref).

model_hyperb *{Default value = none}* sets the model used to compute the hyperbolic part of the equation (see section **??**PFG:add ref). Allowed values:

none no hyperbolic computation

ddp Depth-Dependant Porosity model (see section **??**PFG:add ref).

dip Dual-Integral Porosity model (see section **??**PFG:add ref).

sp Single Porosity model (see section **??**PFG:add ref).

swes Shallow water model (see section **??**PFG:add ref).

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 (see section **??**PFG:add ref).

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.

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 chosen hyperbolic model:

value for the parameter **model_hyperb**	default value	allowed value
ddp	godunov	godunov
dip	godunov	godunov
sp	godunov	godunov
swes	godunov	godunov, muscl_evr

See section **??**PFG:add ref for details regarding numerical scheme implementations.

storeflag *{Default value = default}* defines stored variables in hydrodynamics results maps. PFG:reprendre ce point si on fait des formats différents pour chaque modèle Allowed values: **none**, **default** and **all**. See section **??**PFG:add ref to resulthydro for the list of stored variables, depending on the user-defined hyperbolic model (parameter **model_hyperb**).

t0 *{Default value = 0}* Starting time of the simulation.PFG:need for mr_tests with t0 not 0 PFG:later one: is there a link between t0 and the hotstart?

tmax *{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*. Even if **verb_map_boussinesq** is set to 1, the file *log/boussinesq.txt* is written only if the user-defined hyperbolic model AR:comprends pasinvolved the Boussinesq coefficient.PFG:add a list of the hyperbolic models (all except swes?) using 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:

value for the parameter **model_hyperb**	files generated
ddp	*DDP_debug.txt* *DDP_PhiG.txt* *DDP_PhiW.txt*
dip	*DIP_PhiG.txt* *DIP_PhiW.txt*
sp	*SP_PhiW.txt*
swes	

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*.

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_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_time_series *{Default value = 0}* specifies the level of verbosity of the time series reader function. The higher the value, the stronger the verbosity (0 = no verbosity)

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
========
2dmfile mesh.2dm
```

9.16 precipitation_boundary_time_series.txt Why "boundary"? It is a source term.

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 static wind test
#wind stations
3
#
# t P   P   P
0  1e-6    0e0   1e-5
300 1e-6    1e-5    0e0
360 0e0     0e0     0e0
```

9.17 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
```

9.18 probes_location.txt

sec: probes_location

Purpose

Format description

Example

9.19 singular_head_loss_map.txt

Purpose

Format description

Example

9.20 wind_station_codes_map.txt

sec_SourceTerm! Wind! windStationMap

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
```

sec_SourceTerm!
Wind!
windTimeSeries

9.21 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 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.

Warning. Assume that n and N are respectively the smallest and largest (integer) station numbers in a given station code map (See Section 9.20). 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)
`#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 times the sequence `( <Tab> u_norm <Tab> u_angle)`	`t` : time in seconds ; `u_norm`: wind speed in m/s ; `u_angle`: 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
```

CHAPTER 10

Ouptut Files Format

This chapter describes the format of the result files generated by the simulation process either in the *output* of the *misc* folders. The files in the *output* folder are text file whereas the *misc* folder contains binary files.

10.1 output/ResultBoundariesFlux.txt

Purpose

This file stores the instantaneous discharge crossing each boundary every **dtmap**. In case of the boundary is composed of several interfaces, the stored discharge is the sum of the discharge crossing each interface. As the unit vector normal to the boundary interfaces is oriented from the exterior to the interior, a positive discharge enters the domain.

Format description

<Tab>: tabulation character.

Headerline	time <Tab> BC_{x1}<Tab>...<Tab> BC_{xN} where x1, ..., xN represent the nodestring numbers as defined in the *2dm file*
t <Tab> Q_1 <Tab> ...<Tab> Q_N	t is the stored time, Q_i is the discharge (in $m^3.s^{-1}$) through the boundary xN
...	one line per stored time

Example

```
time BC_1 BC_2 BC_3
2.0000000000e+1 0.0000000000e+0 4.3274586317e+1 -1.4193271476e+1
3.0000000000e+1 0.0000000000e+0 4.3149535653e+1 -1.4227591312e+1
4.0000000000e+1 0.0000000000e+0 4.3074013841e+1 -1.4246148173e+1
5.0000000000e+1 0.0000000000e+0 4.3030038602e+1 -1.4258119344e+1
6.0000000000e+1 0.0000000000e+0 4.3000513361e+1 -1.4266604580e+1
```

10.2 output/ResultHydro_pDDDDdHHhMMmSSsXXX.txt

Purpose

This file stores the hydrodynamic results every **dtmap**. To avoid any computational timestep reduction to store the results precisely every **dtmap**, the results are stored when PFG:explain algorithm. The stored variables depend upon the chosen hyperbolic model (defined by the parameter **model_hyperb**) and the **storeflag** parameter:

model_hyperb	**storeflag** value	
	default	all
ddp		
dip		
sp		
swes	h, q, r	

PFG:complete based on issue #265

Format description

The precise simulation time for storing the variables is indicated in the file name using the ISO 8601 format. The file name is *ResultHydro_pDDDDdHHhMMmSSsXXX.txt* where *pDDDDdHHhMMmSSsXXX* is the elapsed time from the beginning of the simulation:

- DDDD is the number of days (always on 4 digits)
- HH is the number of hours (always on 2 digits)
- MM is the number of minutes (always on 2 digits)
- SS is the number of seconds (always on 2 digits)
- XXX is the number of milliseconds (always on 3 digits)

PART III

Developer Guide

CHAPTER 11

Before You Install

11.1 Understand System Requirements

Conda

The SW2D software strongly relies on Conda, an open source package management. This software is mandatory for any install from sources. Installing SW2D thanks to binary files *does not* require installing Conda.
Conda is a tool to install and manage different software libraries, possibly at different versions, into independent system configurations called environments. Each environment has a name and stores its own version of each library (distinct from the one of your general system) so that you can make installations that would be incompatible with the rest of your system without impacting its functioning and breaking everything! Concretely, miniconda will create a new directory for each environment where it will download libraries, binaries and packages.
SW2D has its own set of required libraries at specific versions, and the safest way to install it and its requirements without affecting the rest of your system is to do it through a new independent conda environment.

Download Conda for Linux, MacOs and Windows on conda.io. PFG:add precision: which version of python for conda, what install options? (add to path, admin mode, etc.)

Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

In the unlikely event that Git is not already installed on your machine, you can download it here, regardless of your operating system.

11.2 The SW2D Structure In A Nutshell

The SW2D software is based on several program in order to optimize the computational time (see Figure 11.1). The process of the model geometry is separated from the main computation process to avoid to reprocess the geometry each time the hydrodynamics variables are modified by the user. The computation can be realised either in a GUI or in a solver mode (with no GUI). The current version of SW2D requires a complementary software to produce the mesh and the parameter files.

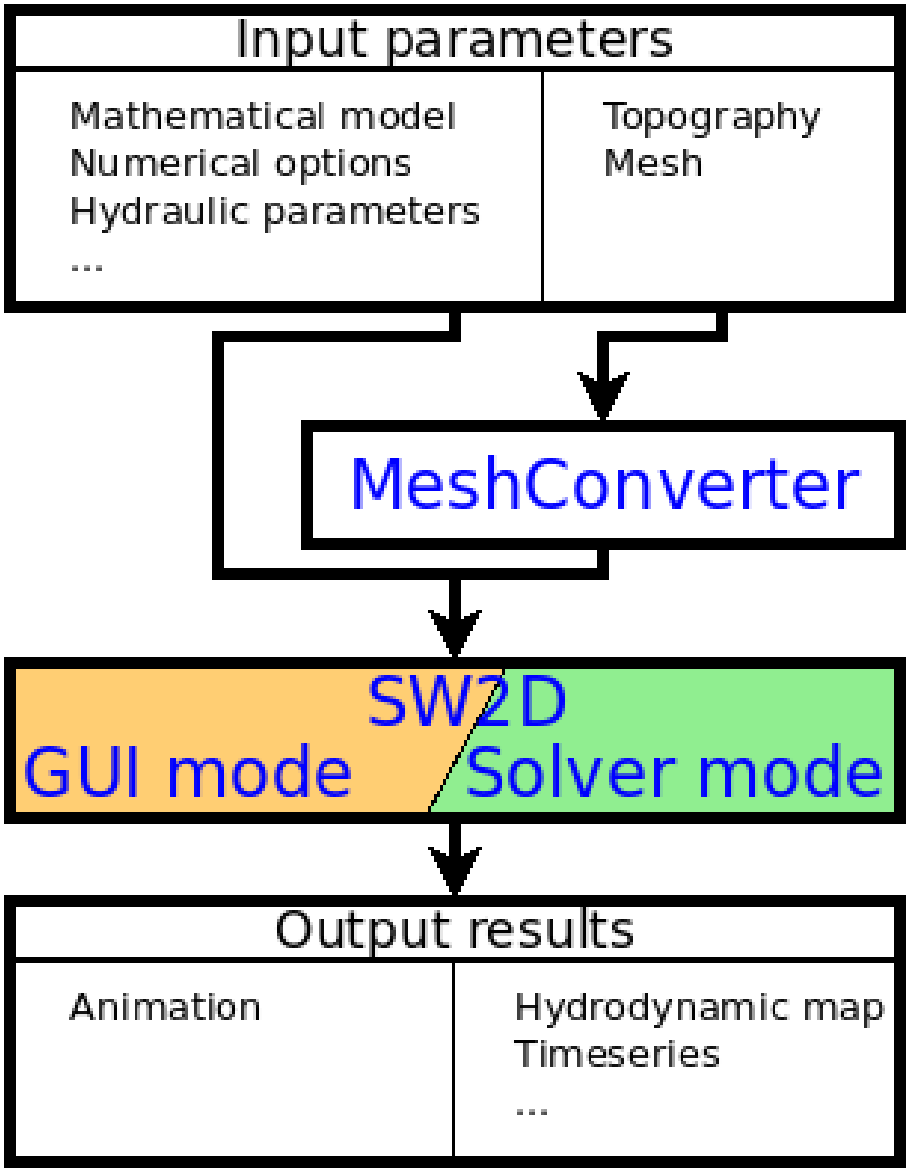


fig:Dev_layout

Figure 11.1: Workflow of a typical SW2D computation

CHAPTER 12

Install from sources

12.1 Linux

1. Additional packages
   - SW2D considers the following packages are installed (default packages in Fedora): libqt5x11extras5-dev, qtbase5-dev, qtdeclarative5-dev, uuid-dev. If not, please install them:

```
#!/bin/bash
sudo apt install libqt5x11extras5-dev qtbase5-dev
        qtdeclarative5-dev uuid-dev
```

   - Under Ubuntu there are some additional packages that should be installed:

```
#!/bin/bash
sudo apt install libx11-dev libxt-dev libxext-dev libglu1-mesa-dev
    build-essential mesa-common-dev uuid-dev
```

2. Install Conda from the official website repo.continuum.io:
   - Download Conda install script
   - Run the install script and remove it:

```
#!/bin/bash
bash Miniconda3-latest-MacOSX-x86_64.sh
rm Miniconda3-latest-MacOSX-x86_64.sh
```

   - Now you should be able to run *conda* in your terminal

   Remark 12.1.1. Warning: make sure you do not install under the root account or you may not have access to the program from your personal account.

3. [Optional] If you already previously installed SW2D and wish to make a clean install, you should remove previous conda environment:

```
#!/bin/bash
conda deactivate
conda env remove -n sw2d
conda clean --all
```

4. Download the sources from the SW2D website and unzip the SW2D folder. **A**lternatively, you may clone the SW2D repository from the Inria Gitlab:

```
#!/bin/bash
git clone git@gitlab.inria.fr:lemon/sw2d.git
```

5. Navigate into the SW2D directory, create the SW2D conda environment and activate it:

```
#!/bin/bash
cd <YOUR-DIRECTORY-TO-SW2D>
conda env create -f pkg/sw2d-linux.yaml
conda activate sw2d
```

6. Create a build directory from the root and navigate into it before configuring and building:

```
#!/bin/bash
mkdir build
cd build
cmake .. -DCMAKE\_PREFIX\_PATH=\$CONDA_PREFIX
    -DCMAKE\_BUILD\_TYPE=Debug
make -j8
```

Remark 12.1.2. You may want to replace 'Debug' by 'Release', depending on the type of optimization you want.

7. You're all set!

12.2 MacOS

1. Make sure XCode command line tools are installed:

```
#!/bin/bash
sudo xcode-select --install
```

2. Install Conda from the official website repo.continuum.io:
   - Download Conda install script
   - Run the install script and remove it:

```
#!/bin/bash
sudo bash Miniconda3-latest-MacOSX-x86_64.sh
rm Miniconda3-latest-MacOSX-x86_64.sh
```

   - Now you should be able to run *conda* in your terminal
3. [Optional] If you already previously installed SW2D and wish to make a clean install, you should remove previous conda environment:

```
#!/bin/bash
conda deactivate
conda env remove -n sw2d
conda clean --all
```

4. Download the sources from the SW2D website and unzip the SW2D folder. **A**lternatively, you may clone the SW2D repository from the Inria Gitlab:

```
#!/bin/bash
git clone git@gitlab.inria.fr:lemon/sw2d.git
```

5. Navigate into the SW2D directory, create the SW2D conda environment and activate it:

```
#!/bin/bash
cd <YOUR-DIRECTORY-TO-SW2D>
conda env create -f pkg/sw2d-mac.yaml
conda activate sw2d
```

6. Create a build directory from the root and navigate into it before configuring and building:

```
#!/bin/bash
mkdir build
cd build
cmake .. -DCMAKE_PREFIX_PATH=$CONDA_PREFIX
    -DCMAKE_BUILD_TYPE=Debug -DCMAKE_CXX_FLAGS="-g -O2"
make -j8
```

Remark 12.2.1. You may want to replace 'Debug' by 'Release', depending on the type of optimization you want. The optimization flags [-DCMAKE_CXX_FLAGS="-g -O2"] are optional.

7. You're all set!

12.3 Windows

Note: this method *should* work. This does not mean it is the only one but at least, it should work...

Install Git and download sources

Git is a software that allows to work in team on code, to mark its progress step by step and to gather the progress of everyone. In this procedure, you need to have access to the Inria Gitlab, with rights to download SW2D sources.

1. Download and install GIT. Choose you favorite text editor when prompted (if none, select Notepad). Confirm all other default pre-selected options.
2. Set your SSH key.
   - Open your file browser where you wish to download and install SW2D. Right-click and select "Git bash here". This should open a console window.
   - Create your key (replace [EMAIL] by your personal email:

```
#!/bin/bash
ssh-keygen -t ed25519 -C "[EMAIL]"
```

   - When prompted, validate the key location and set your passphrase (twice).
   - Open a file browser and navigate until your C:/Users/Username/.ssh folder.
   - Edit the `id_ed25519.pub` file.
   - Copy the full content of the file.
   - Paste it into the dedicated textbox on your Gitlab personal profile key page and validate the key.
3. Download sources
   - Thanks to the Git console you just opened, clone the whole project:

```
#!/bin/bash
git clone git@gitlab.inria.fr:lemon/sw2d.git
```

     if you get an error message, try the download using the Http protocol:

```
git clone https://gitlab.inria.fr/lemon/sw2d.git
```

   - When prompted, answer yes to the security question related to the ssh host, and enter your passphrase. Download should start and a sw2d folder should be created in your current directory.
   - As soon has the download is over, you may configure your git account (still in the Git Bash console):

```
#!/bin/bash
cd sw2d
git config user.name "[FIRSTNAME] [LASTNAME]"
git config user.email "[EMAIL]"
```

VGt: tout ce qui est fait ci-dessus marche aussi dans une console Conda - est-il préférable de faire installer Conda d'abord ? You're all set. You may now close the GIT Bash console. Let us now move to the Conda environment.

Install Conda and set up your environment

1. Download and install Miniconda3 for Windows from the official website repo.continuum.io.
   - You can set up the directory of your choice when asked, e.g. /.miniconda. WARNING: the installation target you choose must be without any space character (this should be prompted during the install process)
   - Make sure to answer YES when asked to add conda to your PATH
   - Do not register Miniconda3 as the system Python
2. [Optional] If you already previously installed SW2D and wish to make a clean install, you should remove previous conda environment from a **Anaconda Prompt (Miniconda3** console:

   ```
   #Conda prompt
   conda deactivate
   conda env remove -n sw2d
   conda clean --all
   ```

3. Create my Conda environment
   - In your startup menu, choose launch **Anaconda Prompt (Miniconda3)**. A console window should open.
   - Navigate to the location of the SW2D sources (recently downloaded)

     ```
     #Conda prompt
     cd [PathToSW2D]
     ```

   - Configure the SW2D environment

     ```
     #Conda prompt
     conda env update -f pkg\env\sw2d-windows.yaml
     ```

     This will download and install all required packages (this may take a while).

You're all set with Conda. Now you should be able to activate the SW2D environment at any moment:

```
#Conda prompt
conda activate sw2d
```

Install Visual Studio and compile SW2D

We shall use Visual Studio 2019 (Community edition) to build targets that can be launched from the console.

1. Install the Community version (no need to run it so far).
   - In the component list, make sure that **Development Desktop in C++** is selected
   - The install process is very long, take a break!

2. Create the SW2D project and compile

   ATTENTION : JE DOIS ENCORE FAIRE DES MODIFS ICI (ANTOINE) VGt: La procédure ci-dessous est tirée du Readme et ne fonctionne pas

```
set CONDA_PREFIX="C:\Users\vguinot.AD\Miniconda3\envs\sw2d"
set BUILD_CONFIG=Release
set LIBRARY_LIB=%CONDA_PREFIX%\Library\lib
set LIBRARY_PREFIX=%CONDA_PREFIX%\Library
conda activate sw2d
cmake .. -G "Visual Studio 16 2019" ^
-Wno-dev ^
 -DCMAKE_INSTALL_PREFIX=%LIBRARY_PREFIX% ^
 -DCMAKE_PREFIX_PATH=%LIBRARY_PREFIX% ^
 -DCMAKE_INSTALL_RPATH:STRING=%LIBRARY_LIB% ^
 -DCMAKE_INSTALL_NAME_DIR=%LIBRARY_LIB%
```

   Voici les messages à la console:

```
-- Selecting Windows SDK version 10.0.18362.0 to target Windows 10.0.18363.
CMake Error at CMakeLists.txt:171 (add_subdirectory):
  add_subdirectory given source "src" which is not an existing directory.
CMake Error at CMakeLists.txt:172 (add_subdirectory):
  add_subdirectory given source "app" which is not an existing directory.
CMake Error at CMakeLists.txt:173 (add_subdirectory):
  add_subdirectory given source "test/tst_ci" which is not an existing
  directory.
CMake Error: File D:/VGt/sw2d_branches/cmake/sw2dConfig.cmake.in does not exist.
CMake Error at C:/Users/vguinot.AD/Miniconda3/envs/sw2d/Library/share/cmake-3.23/Modules/CM
  configure_file Problem configuring file
Call Stack (most recent call first):
  CMakeLists.txt:188 (configure_package_config_file)
CMake Error: File D:/VGt/sw2d_branches/cmake/sw2dConfig.cmake.in does not exist.
CMake Error at C:/Users/vguinot.AD/Miniconda3/envs/sw2d/Library/share/cmake-3.23/Modules/CM
  configure_file Problem configuring file
Call Stack (most recent call first):
  CMakeLists.txt:194 (configure_package_config_file)
-- Configuring incomplete, errors occurred!
See also "D:/VGt/sw2d_branches/sw2d/CMakeFiles/CMakeOutput.log".
See also "D:/VGt/sw2d_branches/sw2d/CMakeFiles/CMakeError.log".
```

   VGt - fin de ce qui ne marche pas avec le message d'erreur

   - Launch Visual Studio 2019 and open your SW2D folder (the one that you downloaded thanks to GIT)
   - Generate cash thanks to Project/Generate Cash (or Project/Configure SW2D). Remark: this step may be done automatically when you open the SW2D folder.
   - Build your target.

Launch SW2D binaries

You're all set now!

- Open an **Anaconda Prompt** console and navigate wherever you have input files for SW2D
- Activate your SW2D environment

```
#Anaconda prompt
conda activate sw2d
```

- Now you may run all SW2D binaries, such as: VGT : attention, ça ne fonctionne pas !

```
#Anaconda prompt
sw2dSolver.exe
sw2dModeler.exe
sw2dConverter.exe
```

Index