

Documentation  
of  
FullSWOF\_2D

v1.06.00 (2015-03-02)

Generated by Doxygen 1.8.9.1  
on Mon Mar 2 2015 14:12:16



# Chapter 1

## Todo List

### Class `Boundary_condition`

Add time and space dependency in the boundary conditions.

Improve boundary conditions at the second order for the wall and periodic conditions.

Take into account source terms (friction and topography) in the boundary conditions, see [Le Roux \[2001\]](#), [Bristeau and Coussin \[2001\]](#).



# Chapter 2

## Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Boundary_condition . . . . .	25
Bc_imp_discharge . . . . .	15
Bc_imp_height . . . . .	19
Bc_Neumann . . . . .	20
Bc_periodic . . . . .	22
Bc_wall . . . . .	24
Choice_condition . . . . .	28
Choice_flux . . . . .	30
Choice_friction . . . . .	32
Choice_infiltration . . . . .	35
Choice_init_huv . . . . .	36
Choice_init_topo . . . . .	37
Choice_limiter . . . . .	38
Choice_output . . . . .	39
Choice_rain . . . . .	42
Choice_reconstruction . . . . .	43
Choice_scheme . . . . .	45
Flux . . . . .	56
F_HLL . . . . .	51
F_HLL2 . . . . .	52
F_Rusanov . . . . .	54
Friction . . . . .	65
Fr_Darcy_Weisbach . . . . .	58
Fr_Laminar . . . . .	60
Fr_Manning . . . . .	62
No_Friction . . . . .	90
Hydrostatic . . . . .	77
Infiltration . . . . .	79
GreenAmpt . . . . .	69
No_Infiltration . . . . .	92
Initialization_huv . . . . .	81
Huv_generated . . . . .	71
Huv_generated_Radial_Dam_dry . . . . .	72
Huv_generated_Radial_Dam_wet . . . . .	74
Huv_generated_Thacker . . . . .	75

Huv_read . . . . .	76
Initialization_topo . . . . .	83
Topo_generated_flat . . . . .	146
Topo_generated_Thacker . . . . .	147
Topo_read . . . . .	148
Limiter . . . . .	84
Minmod . . . . .	86
VanAlbada . . . . .	150
VanLeer . . . . .	151
Output . . . . .	97
Gnuplot . . . . .	68
No_Evolution_File . . . . .	89
Vtk_Out . . . . .	152
Parameters . . . . .	102
Parser . . . . .	126
Rain . . . . .	127
No_Rain . . . . .	93
Rain_generated . . . . .	129
Rain_read . . . . .	130
Reconstruction . . . . .	131
ENO . . . . .	46
ENO_mod . . . . .	49
MUSCL . . . . .	87
Scheme . . . . .	134
Order1 . . . . .	94
Order2 . . . . .	96

# Chapter 3

## Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Bc_imp_discharge</a>	Imposed discharge . . . . .	15
<a href="#">Bc_imp_height</a>	Imposed water height . . . . .	19
<a href="#">Bc_Neumann</a>	Neumann condition . . . . .	20
<a href="#">Bc_periodic</a>	Periodic condition . . . . .	22
<a href="#">Bc_wall</a>	Wall condition . . . . .	24
<a href="#">Boundary_condition</a>	Boundary condition . . . . .	25
<a href="#">Choice_condition</a>	Choice of boundary condition . . . . .	28
<a href="#">Choice_flux</a>	Choice of numerical flux . . . . .	30
<a href="#">Choice_friction</a>	Choice of friction law . . . . .	32
<a href="#">Choice_infiltration</a>	Choice of infiltration law . . . . .	35
<a href="#">Choice_init_huv</a>	Choice of initialization for h and $U=(u,v)$ . . . . .	36
<a href="#">Choice_init_topo</a>	Choice of initialization for the topography . . . . .	37
<a href="#">Choice_limiter</a>	Choice of slope limiter . . . . .	38
<a href="#">Choice_output</a>	Choice of output format . . . . .	39
<a href="#">Choice_rain</a>	Choice of initialization for the rain . . . . .	42
<a href="#">Choice_reconstruction</a>	Choice of reconstruction . . . . .	43
<a href="#">Choice_scheme</a>	Choice of numerical scheme . . . . .	45
<a href="#">ENO</a>	ENO reconstruction . . . . .	46

<a href="#">ENO_mod</a>	Modified ENO reconstruction . . . . .	49
<a href="#">F_HLL</a>	HLL flux . . . . .	51
<a href="#">F_HLL2</a>	HLL flux . . . . .	52
<a href="#">F_Rusanov</a>	Rusanov flux . . . . .	54
<a href="#">Flux</a>	Numerical flux . . . . .	56
<a href="#">Fr_Darcy_Weisbach</a>	Darcy-Weisbach law . . . . .	58
<a href="#">Fr_Laminar</a>	Laminar law . . . . .	60
<a href="#">Fr_Manning</a>	Manning law . . . . .	62
<a href="#">Friction</a>	Friction law . . . . .	65
<a href="#">Gnuplot</a>	Gnuplot output . . . . .	68
<a href="#">GreenAmpt</a>	Green-Ampt law . . . . .	69
<a href="#">Huv_generated</a>	No water configuration . . . . .	71
<a href="#">Huv_generated_Radial_Dam_dry</a>	Dry radial dam break configuration . . . . .	72
<a href="#">Huv_generated_Radial_Dam_wet</a>	Wet radial dam break configuration . . . . .	74
<a href="#">Huv_generated_Thacker</a>	Thacker configuration . . . . .	75
<a href="#">Huv_read</a>	File configuration . . . . .	76
<a href="#">Hydrostatic</a>	Hydrostatic reconstruction . . . . .	77
<a href="#">Infiltration</a>	Definition of infiltration law . . . . .	79
<a href="#">Initialization_huv</a>	Initialization of h, u and v . . . . .	81
<a href="#">Initialization_topo</a>	Initialization of z . . . . .	83
<a href="#">Limiter</a>	Slope limiter . . . . .	84
<a href="#">Minmod</a>	Minmod slope limiter . . . . .	86
<a href="#">MUSCL</a>	MUSCL reconstruction . . . . .	87
<a href="#">No_Evolution_File</a>	No output . . . . .	89
<a href="#">No_Friction</a>	No friction . . . . .	90
<a href="#">No_Infiltration</a>	No infiltration . . . . .	92



<a href="#">No_Rain</a>	No rain . . . . .	93
<a href="#">Order1</a>	Order 1 scheme . . . . .	94
<a href="#">Order2</a>	Order 2 scheme . . . . .	96
<a href="#">Output</a>	Output format . . . . .	97
<a href="#">Parameters</a>	Gets parameters . . . . .	102
<a href="#">Parser</a>	Parser to read the entries . . . . .	126
<a href="#">Rain</a>	Initialization of the rain . . . . .	127
<a href="#">Rain_generated</a>	Constant rain configuration . . . . .	129
<a href="#">Rain_read</a>	File configuration . . . . .	130
<a href="#">Reconstruction</a>	Reconstruction of the variables . . . . .	131
<a href="#">Scheme</a>	Numerical scheme . . . . .	134
<a href="#">Topo_generated_flat</a>	Flat configuration . . . . .	146
<a href="#">Topo_generated_Thacker</a>	Thacker configuration . . . . .	147
<a href="#">Topo_read</a>	File configuration . . . . .	148
<a href="#">VanAlbada</a>	Van Albada slope limiter . . . . .	150
<a href="#">VanLeer</a>	Van Leer slope limiter . . . . .	151
<a href="#">Vtk_Out</a>	VTK output . . . . .	152



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

Headers/libboundaryconditions/ <a href="#">bc_imp_discharge.hpp</a>	
Imposed discharge . . . . .	155
Headers/libboundaryconditions/ <a href="#">bc_imp_height.hpp</a>	
Imposed water height . . . . .	156
Headers/libboundaryconditions/ <a href="#">bc_neumann.hpp</a>	
Neumann condition . . . . .	156
Headers/libboundaryconditions/ <a href="#">bc_periodic.hpp</a>	
Periodic condition . . . . .	157
Headers/libboundaryconditions/ <a href="#">bc_wall.hpp</a>	
Wall condition . . . . .	158
Headers/libboundaryconditions/ <a href="#">boundary_condition.hpp</a>	
Boundary condition . . . . .	158
Headers/libboundaryconditions/ <a href="#">choice_condition.hpp</a>	
Choice of boundary condition . . . . .	159
Headers/libflux/ <a href="#">choice_flux.hpp</a>	
Choice of numerical flux . . . . .	160
Headers/libflux/ <a href="#">f_hll.hpp</a>	
HLL flux . . . . .	161
Headers/libflux/ <a href="#">f_hll2.hpp</a>	
HLL flux . . . . .	162
Headers/libflux/ <a href="#">f_rusanov.hpp</a>	
Rusanov flux . . . . .	162
Headers/libflux/ <a href="#">flux.hpp</a>	
Numerical flux . . . . .	163
Headers/libfrictions/ <a href="#">choice_friction.hpp</a>	
Choice of friction law . . . . .	164
Headers/libfrictions/ <a href="#">fr_darcy_weisbach.hpp</a>	
Darcy-Weisbach law . . . . .	164
Headers/libfrictions/ <a href="#">fr_laminar.hpp</a>	
Laminar law . . . . .	165
Headers/libfrictions/ <a href="#">fr_manning.hpp</a>	
Manning law . . . . .	166
Headers/libfrictions/ <a href="#">friction.hpp</a>	
Friction law . . . . .	166
Headers/libfrictions/ <a href="#">no_friction.hpp</a>	
No friction . . . . .	167

Headers/libinitializations/ <a href="#">choice_init_huv.hpp</a>	
Choice of initialization for h, u and v	168
Headers/libinitializations/ <a href="#">choice_init_topo.hpp</a>	
Choice of initialization for the topography	169
Headers/libinitializations/ <a href="#">huv_generated.hpp</a>	
No water configuration	169
Headers/libinitializations/ <a href="#">huv_generated_radial_dam_dry.hpp</a>	
Dry radial dam break configuration	170
Headers/libinitializations/ <a href="#">huv_generated_radial_dam_wet.hpp</a>	
Wet radial dam break configuration	171
Headers/libinitializations/ <a href="#">huv_generated_thacker.hpp</a>	
Thacker configuration	171
Headers/libinitializations/ <a href="#">huv_read.hpp</a>	
File configuration	172
Headers/libinitializations/ <a href="#">initialization_huv.hpp</a>	
Initialization of h, u and v	173
Headers/libinitializations/ <a href="#">initialization_topo.hpp</a>	
Initialization of z	173
Headers/libinitializations/ <a href="#">topo_generated_flat.hpp</a>	
Flat configuration	174
Headers/libinitializations/ <a href="#">topo_generated_thacker.hpp</a>	
Thacker configuration	175
Headers/libinitializations/ <a href="#">topo_read.hpp</a>	
File configuration	175
Headers/liblimitations/ <a href="#">choice_limiter.hpp</a>	
Choice of slope limiter	176
Headers/liblimitations/ <a href="#">limiter.hpp</a>	
Slope limiter	177
Headers/liblimitations/ <a href="#">minmod.hpp</a>	
Minmod limiter	178
Headers/liblimitations/ <a href="#">vanalbada.hpp</a>	
Van Albada limiter	178
Headers/liblimitations/ <a href="#">vanleer.hpp</a>	
Van Leer limiter	179
Headers/libparameters/ <a href="#">misc.hpp</a>	
Definitions	180
Headers/libparameters/ <a href="#">parameters.hpp</a>	
Gets parameters	182
Headers/libparser/ <a href="#">parser.hpp</a>	
Parser	183
Headers/librain_infiltration/ <a href="#">choice_infiltration.hpp</a>	
Choice of infiltration law	184
Headers/librain_infiltration/ <a href="#">choice_rain.hpp</a>	
Choice of initialization for the rain	184
Headers/librain_infiltration/ <a href="#">greenampt.hpp</a>	
Green-Ampt law	185
Headers/librain_infiltration/ <a href="#">infiltration.hpp</a>	
Infiltration law	186
Headers/librain_infiltration/ <a href="#">no_infiltration.hpp</a>	
No infiltration	187
Headers/librain_infiltration/ <a href="#">no_rain.hpp</a>	
No rain	187

Headers/librain_infiltration/rain.hpp	
Rain . . . . .	188
Headers/librain_infiltration/rain_generated.hpp	
Constant rain configuration . . . . .	188
Headers/librain_infiltration/rain_read.hpp	
File configuration . . . . .	189
Headers/libreconstructions/choice_reconstruction.hpp	
Choice of reconstruction . . . . .	190
Headers/libreconstructions/eno.hpp	
ENO reconstruction . . . . .	190
Headers/libreconstructions/eno_mod.hpp	
Modified ENO reconstruction . . . . .	191
Headers/libreconstructions/hydrostatic.hpp	
Hydrostatic reconstruction . . . . .	192
Headers/libreconstructions/muscl.hpp	
MUSCL reconstruction . . . . .	192
Headers/libreconstructions/reconstruction.hpp	
Reconstruction . . . . .	193
Headers/libsave/choice_output.hpp	
Choice of output format . . . . .	194
Headers/libsave/gnuplot.hpp	
Gnuplot output . . . . .	194
Headers/libsave/no_evolution_file.hpp	
No output . . . . .	195
Headers/libsave/output.hpp	
Output format . . . . .	196
Headers/libsave/vtk_out.hpp	
VTK output . . . . .	196
Headers/libschemes/choice_scheme.hpp	
Choice of numerical scheme . . . . .	197
Headers/libschemes/order1.hpp	
Order 1 scheme . . . . .	198
Headers/libschemes/order2.hpp	
Order 2 scheme . . . . .	198
Headers/libschemes/scheme.hpp	
Numerical scheme . . . . .	199
Sources/FullSWOF_2D.cpp	
Main function . . . . .	200
Sources/libboundaryconditions/bc_imp_discharge.cpp	
Imposed discharge . . . . .	201
Sources/libboundaryconditions/bc_imp_height.cpp	
Imposed water height . . . . .	201
Sources/libboundaryconditions/bc_neumann.cpp	
Neumann condition . . . . .	202
Sources/libboundaryconditions/bc_periodic.cpp	
Periodic condition . . . . .	203
Sources/libboundaryconditions/bc_wall.cpp	
Wall condition . . . . .	203
Sources/libboundaryconditions/boundary_condition.cpp	
Boundary condition . . . . .	204
Sources/libboundaryconditions/choice_condition.cpp	
Choice of boundary condition . . . . .	204

Sources/libflux/ <a href="#">choice_flux.cpp</a>	
Choice of numerical flux	205
Sources/libflux/ <a href="#">f_hll.cpp</a>	
HLL flux	205
Sources/libflux/ <a href="#">f_hll2.cpp</a>	
HLL flux	206
Sources/libflux/ <a href="#">f_rusanov.cpp</a>	
Rusanov flux	206
Sources/libflux/ <a href="#">flux.cpp</a>	
Numerical flux	207
Sources/libfrictions/ <a href="#">choice_friction.cpp</a>	
Choice of friction law	207
Sources/libfrictions/ <a href="#">fr_darcy_weisbach.cpp</a>	
Darcy-Weisbach law	208
Sources/libfrictions/ <a href="#">fr_laminar.cpp</a>	
Laminar law	208
Sources/libfrictions/ <a href="#">fr_manning.cpp</a>	
Manning law	209
Sources/libfrictions/ <a href="#">friction.cpp</a>	
Friction law	209
Sources/libfrictions/ <a href="#">no_friction.cpp</a>	
No friction	210
Sources/libinitializations/ <a href="#">choice_init_huv.cpp</a>	
Choice of initialization for h, u and v	210
Sources/libinitializations/ <a href="#">choice_init_topo.cpp</a>	
Choice of initialization for the topography	211
Sources/libinitializations/ <a href="#">huv_generated.cpp</a>	
No water configuration	211
Sources/libinitializations/ <a href="#">huv_generated_radial_dam_dry.cpp</a>	
Dry radial dam break configuration	212
Sources/libinitializations/ <a href="#">huv_generated_radial_dam_wet.cpp</a>	
Wet radial dam break configuration	212
Sources/libinitializations/ <a href="#">huv_generated_thacker.cpp</a>	
Thacker configuration	213
Sources/libinitializations/ <a href="#">huv_read.cpp</a>	
File configuration	213
Sources/libinitializations/ <a href="#">initialization_huv.cpp</a>	
Initialization of h, u and v	214
Sources/libinitializations/ <a href="#">initialization_topo.cpp</a>	
Initialization of z	214
Sources/libinitializations/ <a href="#">topo_generated_flat.cpp</a>	
Flat configuration	215
Sources/libinitializations/ <a href="#">topo_generated_thacker.cpp</a>	
Thacker configuration	216
Sources/libinitializations/ <a href="#">topo_read.cpp</a>	
File configuration	216
Sources/liblimitations/ <a href="#">choice_limiter.cpp</a>	
Choice of slope limiter	217
Sources/liblimitations/ <a href="#">limiter.cpp</a>	
Slope limiter	217
Sources/liblimitations/ <a href="#">minmod.cpp</a>	
Minmod limiter	218

Sources/liblimitations/ <a href="#">vanalbada.cpp</a>	
Van Albada limiter . . . . .	218
Sources/liblimitations/ <a href="#">vanleer.cpp</a>	
Van Leer limiter . . . . .	219
Sources/libparameters/ <a href="#">parameters.cpp</a>	
Gets parameters . . . . .	219
Sources/libparser/ <a href="#">parser.cpp</a>	
Parser . . . . .	220
Sources/librain_infiltration/ <a href="#">choice_infiltration.cpp</a>	
Choice of infiltration law . . . . .	220
Sources/librain_infiltration/ <a href="#">choice_rain.cpp</a>	
Choice of initialization for the rain . . . . .	221
Sources/librain_infiltration/ <a href="#">greenampt.cpp</a>	
Green-Ampt law . . . . .	221
Sources/librain_infiltration/ <a href="#">infiltration.cpp</a>	
Infiltration law . . . . .	222
Sources/librain_infiltration/ <a href="#">no_infiltration.cpp</a>	
No infiltration . . . . .	222
Sources/librain_infiltration/ <a href="#">no_rain.cpp</a>	
No rain . . . . .	223
Sources/librain_infiltration/ <a href="#">rain.cpp</a>	
Rain . . . . .	223
Sources/librain_infiltration/ <a href="#">rain_generated.cpp</a>	
Constant rain configuration . . . . .	224
Sources/librain_infiltration/ <a href="#">rain_read.cpp</a>	
File configuration . . . . .	224
Sources/libreconstructions/ <a href="#">choice_reconstruction.cpp</a>	
Choice of reconstruction . . . . .	225
Sources/libreconstructions/ <a href="#">eno.cpp</a>	
ENO reconstruction . . . . .	225
Sources/libreconstructions/ <a href="#">eno_mod.cpp</a>	
Modified ENO reconstruction . . . . .	226
Sources/libreconstructions/ <a href="#">hydrostatic.cpp</a>	
Hydrostatic reconstruction . . . . .	226
Sources/libreconstructions/ <a href="#">muscl.cpp</a>	
MUSCL reconstruction . . . . .	227
Sources/libreconstructions/ <a href="#">reconstruction.cpp</a>	
Reconstruction . . . . .	227
Sources/libsave/ <a href="#">choice_output.cpp</a>	
Choice of output format . . . . .	228
Sources/libsave/ <a href="#">gnuplot.cpp</a>	
Gnuplot output . . . . .	228
Sources/libsave/ <a href="#">no_evolution_file.cpp</a>	
No output . . . . .	229
Sources/libsave/ <a href="#">output.cpp</a>	
Output format . . . . .	229
Sources/libsave/ <a href="#">vtk_out.cpp</a>	
VTK output . . . . .	230
Sources/libschemas/ <a href="#">choice_scheme.cpp</a>	
Choice of numerical scheme . . . . .	230
Sources/libschemas/ <a href="#">order1.cpp</a>	
Order 1 scheme . . . . .	231

Sources/libschemas/ <a href="#">order2.cpp</a>	
Order 2 scheme . . . . .	231
Sources/libschemas/ <a href="#">scheme.cpp</a>	
Numerical scheme . . . . .	232



# Chapter 5

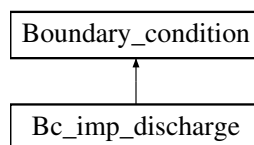
## Class Documentation

### 5.1 Bc\_imp\_discharge Class Reference

Imposed discharge.

```
#include <bc_imp_discharge.hpp>
```

Inheritance diagram for Bc\_imp\_discharge:



#### Public Member Functions

- **Bc\_imp\_discharge** (Parameters &, TAB &, int, int)  
*Constructor.*
- **SCALAR getValueOfPolynomial** (const SCALAR, const SCALAR, const SCALAR, const SCALAR, const SCALAR, int, int) const  
*Gives the value of the function that must vanish.*
- **SCALAR getValueOfDerivativeOfPolynomial** (const SCALAR, const SCALAR, const SCALAR, const SCALAR, const SCALAR, int, int) const  
*Gives the value of the derivative of the function that must vanish.*
- **SCALAR newtonSolver** (const SCALAR, const SCALAR, const SCALAR, const SCALAR, const SCALAR, int, int) const  
*Solves the equation with Newton iterative method.*
- void **calcul** (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int, int)  
*Calculates the boundary condition.*
- virtual **~Bc\_imp\_discharge** ()  
*Destructor.*

#### Additional Inherited Members

##### 5.1.1 Detailed Description

Imposed discharge.

Class that computes the boundary condition where the discharge is imposed. For supercritical flows, the water height is imposed too.

Definition at line 73 of file bc\_imp\_discharge.hpp.

### 5.1.2 Constructor & Destructor Documentation

**Bc\_imp\_discharge::Bc\_imp\_discharge ( Parameters & *par*, TAB & *z*, int *n1*, int *n2* )**

Constructor.

**Parameters**

in	<i>par</i>	parameter, contains all the values from the parameters file (unused).
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.
in, out	<i>z</i>	vector that represents the topography with suitable values on the fictive cells.

Definition at line 60 of file bc\_imp\_discharge.cpp.

**Bc\_imp\_discharge::~Bc\_imp\_discharge ( ) [virtual]**

Destructor.

Definition at line 251 of file bc\_imp\_discharge.cpp.

**5.1.3 Member Function Documentation**

**void Bc\_imp\_discharge::calcul ( SCALAR *hin*, SCALAR *unorm\_in*, SCALAR *utan\_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin\_oppbound*, SCALAR *unorm\_in\_oppbound*, SCALAR *utan\_in\_oppbound*, SCALAR *time*, int *n1*, int *n2* ) [virtual]**

Calculates the boundary condition.

Two cases are considered: subcritical and supercritical flows.

**Parameters**

in	<i>hin</i>	water height of the first cell inside the domain.
in	<i>unorm_in</i>	normal velocity of the first cell inside the domain.
in	<i>utan_in</i>	tangential velocity of the first cell inside the domain.
in	<i>hfix</i>	fixed (imposed) value of the water height (only for the supercritical case).
in	<i>qfix</i>	fixed (imposed) value of the discharge.
in	<i>hin_oppbound</i>	value of the water height of the first cell inside the domain at the opposite bound (unused).
in	<i>unorm_in_↔ oppbound</i>	value of the normal velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>utan_in_↔ oppbound</i>	value of the tangential velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>time</i>	current time (unused).
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.

**Warning**

Warning in the method [Bc\\_imp\\_discharge::calcul\(\)](#) The water height at the inflow is zero ... continuing!

**Modifies**

[Boundary\\_condition::hbound](#) water height on the fictive cell.

[Boundary\\_condition::unormbound](#) normal velocity on the fictive cell.

[Boundary\\_condition::utanbound](#) tangential velocity on the fictive cell.

Implements [Boundary\\_condition](#).

Definition at line 186 of file bc\_imp\_discharge.cpp.

**SCALAR Bc\_imp\_discharge::getValueofDerivativeOfPolynomial ( const SCALAR *HIN*, const SCALAR *UNORM\_IN*, const SCALAR *UTAN\_IN*, const SCALAR *H*, int *n1*, int *n2* ) const**

Gives the value of the derivative of the function that must vanish.

Computes  $3\sqrt{gH} - (n1 + n2)(UNORM\_IN + 2(n1 + n2)\sqrt{gHIN})$  where *n1*, *n2* are the normals.

**Parameters**

in	<i>HIN</i>	water height of the first cell inside the domain.
in	<i>UNORM_IN</i>	normal velocity of the first cell inside the domain.
in	<i>UTAN_IN</i>	tangential velocity of the first cell inside the domain (unused).
in	<i>H</i>	value for the variable of the polynomial function.
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.

**Returns**

The value of derivative of the polynomial function defined in `Bc_imp_discharge::getValueOfPolynomial()`.

Definition at line 128 of file `bc_imp_discharge.cpp`.

**SCALAR `Bc_imp_discharge::getValueOfPolynomial` ( const SCALAR *HIN*, const SCALAR *UNORM\_IN*, const SCALAR *UTAN\_IN*, const SCALAR *QFIX*, const SCALAR *H*, int *n1*, int *n2* ) const**

Gives the value of the function that must vanish.

Computes  $2H\sqrt{gH} - (n1 + n2)(UNORM\_IN + 2(n1 + n2)\sqrt{gHIN})H - |QFIX|$  where *n1*, *n2* are the normals.

**Parameters**

in	<i>HIN</i>	water height of the first cell inside the domain.
in	<i>UNORM_IN</i>	normal velocity of the first cell inside the domain.
in	<i>UTAN_IN</i>	tangential velocity of the first cell inside the domain (unused).
in	<i>QFIX</i>	fixed (imposed) value of the discharge.
in	<i>H</i>	value for the variable of the polynomial function.
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.

**Returns**

The value of the polynomial function.

Definition at line 108 of file `bc_imp_discharge.cpp`.

**SCALAR `Bc_imp_discharge::newtonSolver` ( const SCALAR *HIN*, const SCALAR *UNORM\_IN*, const SCALAR *UTAN\_IN*, const SCALAR *QFIX*, const SCALAR *H\_INIT*, int *n1*, int *n2* ) const**

Solves the equation with Newton iterative method.

Finds the root of the polynomial function corresponding to the imposed discharge. Needs the evaluation of the function and of its derivative.

**Parameters**

in	<i>HIN</i>	water height of the first cell inside the domain.
in	<i>UNORM_IN</i>	normal velocity of the first cell inside the domain.
in	<i>UTAN_IN</i>	tangential velocity of the first cell inside the domain.
in	<i>QFIX</i>	fixed (imposed) value of the discharge.
in	<i>H_INIT</i>	initialisation of the Newton solver.
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.

in	n2	integer to specify whether it is the bottom (-1) or the top (1) boundary.
----	----	---

### Warning

Warning: Newton bc did not converge.

### Returns

**h**: water height that satisfies Riemann invariants.

Definition at line 147 of file bc\_imp\_discharge.cpp.

The documentation for this class was generated from the following files:

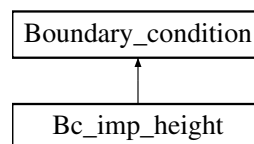
- Headers/libboundaryconditions/bc\_imp\_discharge.hpp
- Sources/libboundaryconditions/bc\_imp\_discharge.cpp

## 5.2 Bc\_imp\_height Class Reference

Imposed water height.

```
#include <bc_imp_height.hpp>
```

Inheritance diagram for Bc\_imp\_height:



### Public Member Functions

- [Bc\\_imp\\_height](#) ([Parameters](#) &, [TAB](#) &, int, int)  
*Constructor.*
- void [calcul](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), int, int)  
*Calculates the boundary condition.*
- virtual [~Bc\\_imp\\_height](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.2.1 Detailed Description

Imposed water height.

Class that computes the boundary condition where the water height is imposed, thanks to the modified method of characteristics. For supercritical flows, the discharge is imposed too.

Definition at line 76 of file bc\_imp\_height.hpp.

#### 5.2.2 Constructor & Destructor Documentation

**Bc\_imp\_height::Bc\_imp\_height** ( [Parameters](#) & *par*, [TAB](#) & *z*, int *n1*, int *n2* )

Constructor.

**Parameters**

in	<i>par</i>	parameter, contains all the values from the parameters file (unused).
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.
in, out	<i>z</i>	vector that represents the topography with suitable values on the fictive cells.

Definition at line 64 of file `bc_imp_height.cpp`.

**Bc\_imp\_height::~Bc\_imp\_height ( ) [virtual]**

Destructor.

Definition at line 175 of file `bc_imp_height.cpp`.

**5.2.3 Member Function Documentation**

**void Bc\_imp\_height::calcul ( SCALAR *hin*, SCALAR *unorm\_in*, SCALAR *utan\_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin\_oppbound*, SCALAR *unorm\_in\_oppbound*, SCALAR *utan\_in\_oppbound*, SCALAR *time*, int *n1*, int *n2* ) [virtual]**

Calculates the boundary condition.

Two cases are considered: subcritical and supercritical flows. In each case, the values to be imposed depend on the flow (inflow or outflow).

**Parameters**

in	<i>hin</i>	water height of the first cell inside the domain.
in	<i>unorm_in</i>	normal velocity of the first cell inside the domain.
in	<i>utan_in</i>	tangential velocity of the first cell inside the domain.
in	<i>hfix</i>	fixed (imposed) value of the water height.
in	<i>qfix</i>	fixed (imposed) value of the discharge.
in	<i>hin_oppbound</i>	value of the water height of the first cell inside the domain at the opposite bound (unused).
in	<i>unorm_in</i> ↔ <i>oppbound</i>	value of the normal velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>utan_in</i> ↔ <i>oppbound</i>	value of the tangential velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>time</i>	current time (unused).
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.

**Modifies**

[Boundary\\_condition::hbound](#) water height on the fictive cell.

[Boundary\\_condition::unormbound](#) normal velocity on the fictive cell.

[Boundary\\_condition::utanbound](#) tangential velocity on the fictive cell.

Implements [Boundary\\_condition](#).

Definition at line 107 of file `bc_imp_height.cpp`.

The documentation for this class was generated from the following files:

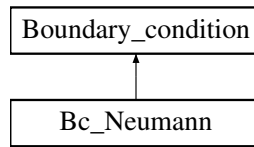
- [Headers/libboundaryconditions/bc\\_imp\\_height.hpp](#)
- [Sources/libboundaryconditions/bc\\_imp\\_height.cpp](#)

**5.3 Bc\_Neumann Class Reference**

Neumann condition.

```
#include <bc_neumann.hpp>
```

Inheritance diagram for Bc\_Neumann:



## Public Member Functions

- `Bc_Neumann (Parameters &, TAB &, int, int)`  
*Constructor.*
- `void calcul (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int, int)`  
*Calculates the boundary condition.*
- `virtual ~Bc_Neumann ()`  
*Destructor.*

## Additional Inherited Members

### 5.3.1 Detailed Description

Neumann condition.

Class that computes the boundary condition with Neumann condition (the normal derivative is null).

Definition at line 73 of file bc\_neumann.hpp.

### 5.3.2 Constructor & Destructor Documentation

**Bc\_Neumann::Bc\_Neumann ( Parameters & *par*, TAB & *z*, int *n1*, int *n2* )**

Constructor.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file (unused).
<i>in</i>	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
<i>in</i>	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.
<i>in, out</i>	<i>z</i>	vector that represents the topography with suitable values on the fictive cells.

Definition at line 61 of file bc\_neumann.cpp.

**Bc\_Neumann::~~Bc\_Neumann ( ) [virtual]**

Destructor.

Definition at line 141 of file bc\_neumann.cpp.

### 5.3.3 Member Function Documentation

**void Bc\_Neumann::calcul ( SCALAR *hin*, SCALAR *unorm\_in*, SCALAR *utan\_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin\_oppbound*, SCALAR *unorm\_in\_oppbound*, SCALAR *utan\_in\_oppbound*, SCALAR *time*, int *n1*, int *n2* ) [virtual]**

Calculates the boundary condition.

**Parameters**

in	<i>hin</i>	water height of the first cell inside the domain.
in	<i>unorm_in</i>	normal velocity of the first cell inside the domain.
in	<i>utan_in</i>	tangential velocity of the first cell inside the domain.
in	<i>hfix</i>	fixed (imposed) value of the water height (unused).
in	<i>qfix</i>	fixed (imposed) value of the discharge (unused).
in	<i>hin_oppbound</i>	value of the water height of the first cell inside the domain at the opposite bound (unused).
in	<i>unorm_in_↔ oppbound</i>	value of the normal velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>utan_in_↔ oppbound</i>	value of the tangential velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>time</i>	current time (unused).
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary (unused).
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary (unused).

**Modifies**

[Boundary\\_condition::hbound](#) water height on the fictive cell.

[Boundary\\_condition::unormbound](#) normal velocity on the fictive cell.

[Boundary\\_condition::utanbound](#) tangential velocity on the fictive cell.

Implements [Boundary\\_condition](#).

Definition at line 106 of file `bc_neumann.cpp`.

The documentation for this class was generated from the following files:

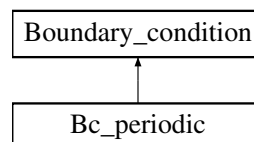
- Headers/libboundaryconditions/[bc\\_neumann.hpp](#)
- Sources/libboundaryconditions/[bc\\_neumann.cpp](#)

**5.4 Bc\_periodic Class Reference**

Periodic condition.

```
#include <bc_periodic.hpp>
```

Inheritance diagram for `Bc_periodic`:

**Public Member Functions**

- [Bc\\_periodic](#) ([Parameters](#) &, [TAB](#) &, int, int)  
*Constructor.*
- void [calcul](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), int, int)  
*Calculates boundary condition.*
- virtual [~Bc\\_periodic](#) ()  
*Destructor.*



## Additional Inherited Members

### 5.4.1 Detailed Description

Periodic condition.

Class that computes the periodic boundary condition

Definition at line 74 of file bc\_periodic.hpp.

### 5.4.2 Constructor & Destructor Documentation

**Bc\_periodic::Bc\_periodic ( Parameters & *par*, TAB & *z*, int *n1*, int *n2* )**

Constructor.

Parameters

in	<i>par</i>	parameter, contains all the values from the parameters file (unused).
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.
in, out	<i>z</i>	vector that represents the topography with suitable values on the fictive cells.

Definition at line 61 of file bc\_periodic.cpp.

**Bc\_periodic::~~Bc\_periodic ( ) [virtual]**

Destructor.

Definition at line 140 of file bc\_periodic.cpp.

### 5.4.3 Member Function Documentation

**void Bc\_periodic::calcul ( SCALAR *hin*, SCALAR *unorm\_in*, SCALAR *utan\_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin\_oppbound*, SCALAR *unorm\_in\_oppbound*, SCALAR *utan\_in\_oppbound*, SCALAR *time*, int *n1*, int *n2* ) [virtual]**

Calculates boundary condition.

The velocity and water height are fixed to have the same behavior at each bound of the domain.

Parameters

in	<i>hin</i>	water height of the first cell inside the domain (unused).
in	<i>unorm_in</i>	normal velocity of the first cell inside the domain (unused).
in	<i>utan_in</i>	tangential velocity of the first cell inside the domain (unused).
in	<i>hfix</i>	fixed (imposed) value of the water height (unused).
in	<i>qfix</i>	fixed (imposed) value of the discharge (unused).
in	<i>hin_oppbound</i>	value of the water height of the first cell inside the domain at the opposite bound.
in	<i>unorm_in</i> ↔ <i>oppbound</i>	value of the normal velocity of the first cell inside the domain at the opposite bound.
in	<i>utan_in</i> ↔ <i>oppbound</i>	value of the tangential velocity of the first cell inside the domain at the opposite bound.
in	<i>time</i>	current time (unused).
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary (unused).
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary (unused).

Modifies

[Boundary\\_condition::hbound](#) water height on the fictive cell.

[Boundary\\_condition::unormbound](#) normal velocity on the fictive cell.

[Boundary\\_condition::utanbound](#) tangential velocity on the fictive cell.

Implements [Boundary\\_condition](#).

Definition at line 104 of file `bc_periodic.cpp`.

The documentation for this class was generated from the following files:

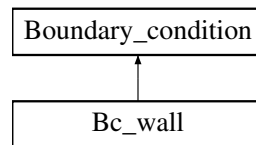
- [Headers/libboundaryconditions/bc\\_periodic.hpp](#)
- [Sources/libboundaryconditions/bc\\_periodic.cpp](#)

## 5.5 Bc\_wall Class Reference

Wall condition.

```
#include <bc_wall.hpp>
```

Inheritance diagram for `Bc_wall`:



### Public Member Functions

- [Bc\\_wall](#) ([Parameters](#) &, [TAB](#) &, int, int)  
*Constructor.*
- void [calcul](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), int, int)  
*Calculates the boundary condition.*
- virtual [~Bc\\_wall](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.5.1 Detailed Description

Wall condition.

Class that computes the wall boundary condition.

Definition at line 71 of file `bc_wall.hpp`.

#### 5.5.2 Constructor & Destructor Documentation

**`Bc_wall::Bc_wall ( Parameters & par, TAB & z, int n1, int n2 )`**

Constructor.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
<code>in</code>	<code>n1</code>	integer to specify whether it is the left (-1) or the right (1) boundary.
<code>in</code>	<code>n2</code>	integer to specify whether it is the bottom (-1) or the top (1) boundary.

<code>in, out</code>	<code>z</code>	vector that represents the topography with suitable values on the fictive cells.
----------------------	----------------	--

Definition at line 61 of file `bc_wall.cpp`.

**Bc\_wall::~Bc\_wall( ) [virtual]**

Destructor.

Definition at line 137 of file `bc_wall.cpp`.

### 5.5.3 Member Function Documentation

**void Bc\_wall::calcul ( SCALAR *hin*, SCALAR *unorm\_in*, SCALAR *utan\_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin\_oppbound*, SCALAR *unorm\_in\_oppbound*, SCALAR *utan\_in\_oppbound*, SCALAR *time*, int *n1*, int *n2* ) [virtual]**

Calculates the boundary condition.

Parameters

<code>in</code>	<code>hin</code>	water height of the first cell inside the domain.
<code>in</code>	<code>unorm_in</code>	normal velocity of the first cell inside the domain.
<code>in</code>	<code>utan_in</code>	tangential velocity of the first cell inside the domain.
<code>in</code>	<code>hfix</code>	fixed (imposed) value of the water height (unused).
<code>in</code>	<code>qfix</code>	fixed (imposed) value of the discharge (unused).
<code>in</code>	<code>hin_oppbound</code>	value of the water height of the first cell inside the domain at the opposite bound (unused).
<code>in</code>	<code>unorm_in↔ oppbound</code>	value of the normal velocity of the first cell inside the domain at the opposite bound (unused).
<code>in</code>	<code>utan_in↔ oppbound</code>	value of the tangential velocity of the first cell inside the domain at the opposite bound (unused).
<code>in</code>	<code>time</code>	current time (unused).
<code>in</code>	<code>n1</code>	integer to specify whether it is the left (-1) or the right (1) boundary (unused).
<code>in</code>	<code>n2</code>	integer to specify whether it is the bottom (-1) or the top (1) boundary (unused).

Modifies

[Boundary\\_condition::hbound](#) water height on the fictive cell.

[Boundary\\_condition::unormbound](#) normal velocity on the fictive cell.

[Boundary\\_condition::utanbound](#) tangential velocity on the fictive cell.

Implements [Boundary\\_condition](#).

Definition at line 102 of file `bc_wall.cpp`.

The documentation for this class was generated from the following files:

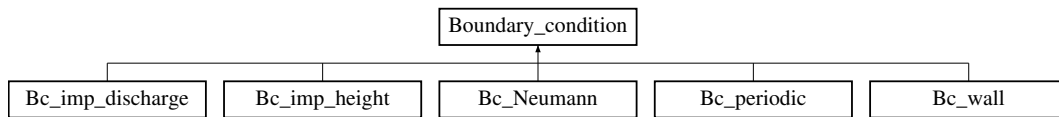
- [Headers/libboundaryconditions/bc\\_wall.hpp](#)
- [Sources/libboundaryconditions/bc\\_wall.cpp](#)

## 5.6 Boundary\_condition Class Reference

Boundary condition.

```
#include <boundary_condition.hpp>
```

Inheritance diagram for `Boundary_condition`:



## Public Member Functions

- `Boundary_condition` (Parameters &)  
*Constructor.*
- virtual void `calcul` (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int, int)=0  
*Function to be specified in each boundary condition.*
- `SCALAR get_hbound` () const  
*Gives the water height on the fictive cell.*
- `SCALAR get_unormbound` () const  
*Gives the normal velocity of the flow on the fictive cell.*
- `SCALAR get_utanbound` () const  
*Gives the tangential velocity of the flow on the fictive cell.*
- virtual `~Boundary_condition` ()  
*Destructor.*

## Protected Attributes

- const int `NXCELL`
- const int `NYCELL`
- `SCALAR hbound`
- `SCALAR unormbound`
- `SCALAR utanbound`
- `SCALAR unormfix`

### 5.6.1 Detailed Description

Boundary condition.

Class that contains all the common declarations for the boundary conditions.

**Todo** Add time and space dependency in the boundary conditions.

Improve boundary conditions at the second order for the wall and periodic conditions.

Take into account source terms (friction and topography) in the boundary conditions, see [Le Roux \[2001\]](#), [Bristeau and Coussin \[2001\]](#).

Definition at line 74 of file `boundary_condition.hpp`.

### 5.6.2 Constructor & Destructor Documentation

**Boundary\_condition::Boundary\_condition** ( Parameters & *par* )

Constructor.

Defines the number of cells.

**Parameters**

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 60 of file boundary\_condition.cpp.

**Boundary\_condition::~Boundary\_condition ( ) [virtual]**

Destructor.

Definition at line 100 of file boundary\_condition.cpp.

**5.6.3 Member Function Documentation****virtual void Boundary\_condition::calcul ( SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , int , int ) [pure virtual]**

Function to be specified in each boundary condition.

Implemented in [Bc\\_imp\\_discharge](#), [Bc\\_imp\\_height](#), [Bc\\_periodic](#), [Bc\\_Neumann](#), and [Bc\\_wall](#).

**SCALAR Boundary\_condition::get\_hbound ( ) const**

Gives the water height on the fictive cell.

Returns

[Boundary\\_condition::hbound](#) water height on the fictive cell.

Definition at line 69 of file boundary\_condition.cpp.

**SCALAR Boundary\_condition::get\_unormbound ( ) const**

Gives the normal velocity of the flow on the fictive cell.

Returns

[Boundary\\_condition::unormbound](#) normal velocity on the fictive cell.

Definition at line 79 of file boundary\_condition.cpp.

**SCALAR Boundary\_condition::get\_utanbound ( ) const**

Gives the tangential velocity of the flow on the fictive cell.

Returns

[Boundary\\_condition::utanbound](#) tangential velocity on the fictive cell.

Definition at line 89 of file boundary\_condition.cpp.

**5.6.4 Member Data Documentation****SCALAR Boundary\_condition::hbound [protected]**

Water height on the fictive cell, to be specified in each boundary condition.

Definition at line 102 of file boundary\_condition.hpp.

**const int Boundary\_condition::NXCELL [protected]**

Number of cells (in space) in the x direction.

Definition at line 98 of file boundary\_condition.hpp.

**const int Boundary\_condition::NYCELL [protected]**

Number of cells (in space) in the y direction.

Definition at line 100 of file boundary\_condition.hpp.

**SCALAR Boundary\_condition::unormbound [protected]**

Normal velocity on the fictive cell, to be specified in each boundary condition.

Definition at line 104 of file boundary\_condition.hpp.

**SCALAR Boundary\_condition::unormfix [protected]**

Imposed value of the velocity from [Parameters::left\\_imp\\_discharge](#) (or [Parameters::right\\_imp\\_discharge](#), [Parameters::bottom\\_imp\\_discharge](#), [Parameters::top\\_imp\\_discharge](#)) and [Parameters::left\\_imp\\_h](#) (or [Parameters::right\\_imp\\_h](#), [Parameters::bottom\\_imp\\_h](#), [Parameters::top\\_imp\\_h](#)).

Definition at line 108 of file boundary\_condition.hpp.

**SCALAR Boundary\_condition::utanbound [protected]**

Tangential velocity on the fictive cell, to be specified in each boundary condition.

Definition at line 106 of file boundary\_condition.hpp.

The documentation for this class was generated from the following files:

- [Headers/libboundaryconditions/boundary\\_condition.hpp](#)
- [Sources/libboundaryconditions/boundary\\_condition.cpp](#)

## 5.7 Choice\_condition Class Reference

Choice of boundary condition.

```
#include <choice_condition.hpp>
```

### Public Member Functions

- [Choice\\_condition](#) (int, [Parameters](#) &, [TAB](#) &, int, int)  
*Constructor.*
- void [calcul](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), int, int)  
*Calculates the boundary condition.*
- [SCALAR get\\_hbound](#) ()  
*Gives the water height on the fictive cell.*
- [SCALAR get\\_unormbound](#) ()  
*Gives the normal velocity of the flow on the fictive cell.*
- [SCALAR get\\_utanbound](#) ()  
*Gives the tangential velocity of the flow on the fictive cell.*
- virtual [~Choice\\_condition](#) ()  
*Destructor.*

### 5.7.1 Detailed Description

Choice of boundary condition.

Class that calls the boundary condition chosen in the parameters file.

Definition at line 94 of file choice\_condition.hpp.

### 5.7.2 Constructor & Destructor Documentation

**Choice\_condition::Choice\_condition ( int *choice*, Parameters & *par*, TAB & *z*, int *n1*, int *n2* )**

Constructor.

Defines the boundary condition from the value given in the parameters file.

Parameters

in	<i>choice</i>	integer that correspond to the chosen boundary condition.
in	<i>par</i>	parameter, contains all the values from the parameters file.
in	<i>z</i>	array that represents the topography.
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.

Definition at line 61 of file choice\_condition.cpp.

**Choice\_condition::~~Choice\_condition ( ) [virtual]**

Destructor.

Definition at line 146 of file choice\_condition.cpp.

### 5.7.3 Member Function Documentation

**void Choice\_condition::calcul ( SCALAR *hin*, SCALAR *unorm\_in*, SCALAR *utan\_in*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin\_oppbound*, SCALAR *unorm\_in\_oppbound*, SCALAR *utan\_in\_oppbound*, SCALAR *time*, int *n1*, int *n2* )**

Calculates the boundary condition.

Calls the calculation of the boundary condition.

Parameters

in	<i>hin</i>	water height of the first cell inside the domain.
in	<i>unorm_in</i>	normal velocity of the first cell inside the domain.
in	<i>utan_in</i>	tangential velocity of the first cell inside the domain.
in	<i>hfix</i>	fixed (imposed) value of the water height.
in	<i>qfix</i>	fixed (imposed) value of the discharge.
in	<i>hin_oppbound</i>	value of the water height of the first cell inside the domain at the opposite bound.
in	<i>unorm_in_↔ oppbound</i>	value of the normal velocity of the first cell inside the domain at the opposite bound.
in	<i>utan_in_↔ oppbound</i>	value of the tangential velocity of the first cell inside the domain at the opposite bound.
in	<i>time</i>	current time.
in	<i>n1</i>	integer to specify whether it is the left (-1) or the right (1) boundary.
in	<i>n2</i>	integer to specify whether it is the bottom (-1) or the top (1) boundary.

Definition at line 92 of file choice\_condition.cpp.

**SCALAR Choice\_condition::get\_hbound ( )**

Gives the water height on the fictive cell.

Calls the function to get the water height on the fictive cell.

Returns

[Boundary\\_condition::hbound](#) water height on the fictive cell for the chosen boundary condition.

Definition at line 113 of file choice\_condition.cpp.

**SCALAR Choice\_condition::get\_unormbound ( )**

Gives the normal velocity of the flow on the fictive cell.

Calls the function to get the normal velocity on the fictive cell.

Returns

[Boundary\\_condition::unormbound](#) normal velocity on the fictive cell for the chosen boundary condition.

Definition at line 124 of file choice\_condition.cpp.

**SCALAR Choice\_condition::get\_utanbound ( )**

Gives the tangential velocity of the flow on the fictive cell.

Calls the function to get the tangential velocity on the fictive cell.

Returns

[Boundary\\_condition::utanbound](#) tangential velocity on the fictive cell for the chosen boundary condition.

Definition at line 135 of file choice\_condition.cpp.

The documentation for this class was generated from the following files:

- Headers/libboundaryconditions/[choice\\_condition.hpp](#)
- Sources/libboundaryconditions/[choice\\_condition.cpp](#)

## 5.8 Choice\_flux Class Reference

Choice of numerical flux.

```
#include <choice_flux.hpp>
```

### Public Member Functions

- [Choice\\_flux](#) (int)  
*Constructor.*
- void [calcul](#) (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR)  
*Calculates the numerical flux.*
- void [set\\_tx](#) (SCALAR)  
*Sets the variable Flux::tx.*
- [SCALAR get\\_f1](#) ()  
*Gives the first component of the numerical flux.*
- [SCALAR get\\_f2](#) ()  
*Gives the second component of the numerical flux.*
- [SCALAR get\\_f3](#) ()  
*Gives the third component of the numerical flux.*
- [SCALAR get\\_cfl](#) ()  
*Gives the CFL value.*
- virtual [~Choice\\_flux](#) ()  
*Destructor.*

### 5.8.1 Detailed Description

Choice of numerical flux.

Class that calls the numerical flux chosen in the parameters file.

Definition at line 85 of file choice\_flux.hpp.



## 5.8.2 Constructor & Destructor Documentation

### Choice\_flux::Choice\_flux ( int *choice* )

Constructor.

Defines the numerical flux from the value given in the parameters file.

Parameters

<i>in</i>	<i>choice</i>	integer that correspond to the chosen numerical flux.
-----------	---------------	---

Definition at line 61 of file choice\_flux.cpp.

### Choice\_flux::~~Choice\_flux ( ) [virtual]

Destructor.

Definition at line 155 of file choice\_flux.cpp.

## 5.8.3 Member Function Documentation

### void Choice\_flux::calcul ( SCALAR *h\_L*, SCALAR *u\_L*, SCALAR *v\_L*, SCALAR *h\_R*, SCALAR *u\_R*, SCALAR *v\_R* )

Calculates the numerical flux.

Calls the calculation of the numerical flux.

Parameters

<i>in</i>	<i>h_L</i>	water height at the left of the interface where the flux is calculated.
<i>in</i>	<i>u_L</i>	velocity (in the x direction) at the left of the interface where the flux is calculated.
<i>in</i>	<i>v_L</i>	velocity (in the y direction) at the left of the interface where the flux is calculated.
<i>in</i>	<i>h_R</i>	water height at the right of the interface where the flux is calculated.
<i>in</i>	<i>u_R</i>	velocity (in the x direction) at the right of the interface where the flux is calculated.
<i>in</i>	<i>v_R</i>	velocity (in the y direction) at the right of the interface where the flux is calculated.

Definition at line 84 of file choice\_flux.cpp.

### SCALAR Choice\_flux::get\_cfl ( )

Gives the CFL value.

Calls the function to get the value of the CFL.

Returns

[Flux::cfl](#) value of the CFL.

Definition at line 144 of file choice\_flux.cpp.

### SCALAR Choice\_flux::get\_f1 ( )

Gives the first component of the numerical flux.

Calls the function to get the first component of the numerical flux.

Returns

[Flux::f1](#) first component of the numerical flux.

Definition at line 111 of file choice\_flux.cpp.

**SCALAR Choice\_flux::get\_f2 ( )**

Gives the second component of the numerical flux.

Calls the function to get the second component of the numerical flux.

Returns

[Flux::f2](#) second component of the numerical flux.

Definition at line 122 of file choice\_flux.cpp.

**SCALAR Choice\_flux::get\_f3 ( )**

Gives the third component of the numerical flux.

Calls the function to get the third component of the numerical flux.

Returns

[Flux::f3](#) third component of the numerical flux.

Definition at line 133 of file choice\_flux.cpp.

**void Choice\_flux::set\_tx ( SCALAR tx )**

Sets the variable [Flux::tx](#).

Calls the setting of the value given in parameter to the variable **tx**.

Parameters

in	tx	value of dt/dx.
----	----	-----------------

Definition at line 100 of file choice\_flux.cpp.

The documentation for this class was generated from the following files:

- [Headers/libflux/choice\\_flux.hpp](#)
- [Sources/libflux/choice\\_flux.cpp](#)

**5.9 Choice\_friction Class Reference**

Choice of friction law.

```
#include <choice_friction.hpp>
```

**Public Member Functions**

- [Choice\\_friction](#) ([Parameters](#) &)  
*Constructor.*
- void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, [SCALAR](#))  
*Calculates the friction term.*
- [TAB get\\_q1mod](#) ()  
*Gives the discharge in the first direction modified by the friction term.*
- [TAB get\\_q2mod](#) ()  
*Gives the discharge in the second direction modified by the friction term.*
- void [calculSf](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &)  
*Calculates the explicit friction term. It will be used for computations with erosion.*
- [TAB get\\_Sf1](#) ()  
*Gives the explicit friction term in the first direction.*
- [TAB get\\_Sf2](#) ()

*Gives the explicit friction term in the second direction.*

- virtual `~Choice_friction ()`

*Destructor.*

### 5.9.1 Detailed Description

Choice of friction law.

Class that calls the friction law chosen in the parameters file.

Definition at line 89 of file `choice_friction.hpp`.

### 5.9.2 Constructor & Destructor Documentation

#### **Choice\_friction::Choice\_friction ( Parameters & *par* )**

Constructor.

Defines the friction law from the value given in the parameters file.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 60 of file `choice_friction.cpp`.

#### **Choice\_friction::~~Choice\_friction ( ) [virtual]**

Destructor.

Definition at line 161 of file `choice_friction.cpp`.

### 5.9.3 Member Function Documentation

#### **void Choice\_friction::calcul ( const TAB & *uold*, const TAB & *vold*, const TAB & *hnew*, const TAB & *q1new*, const TAB & *q2new*, SCALAR *dt* )**

Calculates the friction term.

Calls the calculation of the friction law.

Parameters

<code>in</code>	<code>uold</code>	velocity in the first direction at the previous time ( <i>n</i> if you are calculating the <i>n</i> + 1th time step).
<code>in</code>	<code>vold</code>	velocity in the second direction at the previous time ( <i>n</i> if you are calculating the <i>n</i> + 1th time step).
<code>in</code>	<code>hnew</code>	water height after the Shallow-Water computation (without friction).
<code>in</code>	<code>q1new</code>	discharge in the first direction after the Shallow-Water computation (without friction).
<code>in</code>	<code>q2new</code>	discharge in the second direction after the Shallow-Water computation (without friction).
<code>in</code>	<code>dt</code>	time step.

Note

The friction only affects the discharge.

Definition at line 85 of file `choice_friction.cpp`.

#### **void Choice\_friction::calculSf ( const TAB & *h*, const TAB & *u*, const TAB & *v* )**

Calculates the explicit friction term. It will be used for computations with erosion.

Calls the calculation of the explicit friction law.

**Parameters**

<code>in</code>	<code>h</code>	water height.
<code>in</code>	<code>u</code>	velocity in the first direction.
<code>in</code>	<code>v</code>	velocity in the second direction.

**Note**

This term will be used to compute erosion.

Definition at line 125 of file `choice_friction.cpp`.

**TAB Choice\_friction::get\_q1mod ( )**

Gives the discharge in the first direction modified by the friction term.

Calls the function to get the discharge in the first direction modified by the friction term.

**Returns**

[Friction::q1mod](#) discharge in the first direction modified by the friction term.

Definition at line 102 of file `choice_friction.cpp`.

**TAB Choice\_friction::get\_q2mod ( )**

Gives the discharge in the second direction modified by the friction term.

Calls the function to get the discharge in the second direction modified by the friction term.

**Returns**

[Friction::q2mod](#) discharge in the second direction modified by the friction term.

Definition at line 113 of file `choice_friction.cpp`.

**TAB Choice\_friction::get\_Sf1 ( )**

Gives the explicit friction term in the first direction.

Calls the function to get the explicit friction term in the first direction.

**Returns**

[Friction::Sf1](#) explicit friction term in the first direction.

Definition at line 139 of file `choice_friction.cpp`.

**TAB Choice\_friction::get\_Sf2 ( )**

Gives the explicit friction term in the second direction.

Calls the function to get the explicit friction term in the second direction.

**Returns**

[Friction::Sf2](#) explicit friction term in the second direction.

Definition at line 150 of file `choice_friction.cpp`.

The documentation for this class was generated from the following files:

- Headers/libfrictions/[choice\\_friction.hpp](#)
- Sources/libfrictions/[choice\\_friction.cpp](#)

## 5.10 Choice\_infiltration Class Reference

Choice of infiltration law.

```
#include <choice_infiltration.hpp>
```

### Public Member Functions

- [Choice\\_infiltration](#) ([Parameters](#) &)  
*Constructor.*
- void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))  
*Performs the computation of the modified water height and the infiltrated volume.*
- virtual [~Choice\\_infiltration](#) ()  
*Destructor.*
- [TAB](#) [get\\_hmod](#) ()  
*Gives the value of the modified water height.*
- [TAB](#) [get\\_Vin](#) ()  
*Gives the value of the infiltrated volume.*

### 5.10.1 Detailed Description

Choice of infiltration law.

Class that calls the infiltration chosen in the parameters file.

Definition at line 82 of file choice\_infiltration.hpp.

### 5.10.2 Constructor & Destructor Documentation

#### [Choice\\_infiltration::Choice\\_infiltration](#) ( [Parameters](#) & *par* )

Constructor.

Defines the friction law from the value given in the parameters file.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 61 of file choice\_infiltration.cpp.

#### [Choice\\_infiltration::~~Choice\\_infiltration](#) ( ) [[virtual](#)]

Destructor.

Definition at line 112 of file choice\_infiltration.cpp.

### 5.10.3 Member Function Documentation

#### [void Choice\\_infiltration::calcul](#) ( const [TAB](#) & *h*, const [TAB](#) & *Vin*, const [SCALAR](#) *dt* )

Performs the computation of the modified water height and the infiltrated volume.

Calls the computation of infiltration.

Parameters

<code>in</code>	<code>h</code>	water height.
<code>in</code>	<code>Vin</code>	infiltrated volume.

in	dt	time step.
----	----	------------

Definition at line 80 of file choice\_infiltration.cpp.

### **TAB Choice\_infiltration::get\_hmod ( )**

Gives the value of the modified water height.

Returns

The value hmod [Infiltration::hmod](#).

Definition at line 92 of file choice\_infiltration.cpp.

### **TAB Choice\_infiltration::get\_Vin ( )**

Gives the value of the infiltrated volume.

Returns

The value Vin [Infiltration::Vin](#).

Definition at line 102 of file choice\_infiltration.cpp.

The documentation for this class was generated from the following files:

- Headers/librain\_infiltration/[choice\\_infiltration.hpp](#)
- Sources/librain\_infiltration/[choice\\_infiltration.cpp](#)

## **5.11 Choice\_init\_huv Class Reference**

Choice of initialization for h and U=(u,v)

```
#include <choice_init_huv.hpp>
```

### **Public Member Functions**

- [Choice\\_init\\_huv](#) ([Parameters](#) &)  
*Constructor.*
- void [initialization](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &)  
*Performs the initialization.*
- virtual [~Choice\\_init\\_huv](#) ()  
*Destructor.*

#### **5.11.1 Detailed Description**

Choice of initialization for h and U=(u,v)

Class that calls the initialization of the water height and of the velocity chosen in the parameters file.

Definition at line 93 of file choice\_init\_huv.hpp.

#### **5.11.2 Constructor & Destructor Documentation**

##### **Choice\_init\_huv::Choice\_init\_huv ( [Parameters](#) & *par* )**

Constructor.

Defines the initialization of the water height and of the velocity from the value given in the parameters file.

**Parameters**

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 60 of file choice\_init\_huv.cpp.

**Choice\_init\_huv::~~Choice\_init\_huv ( ) [virtual]**

Destructor.

Definition at line 101 of file choice\_init\_huv.cpp.

**5.11.3 Member Function Documentation****void Choice\_init\_huv::initialization ( TAB & h, TAB & u, TAB & v )**

Performs the initialization.

Calls the initialization of the water height and of the velocity.

**Parameters**

<code>in</code>	<code>h</code>	water height.
<code>in</code>	<code>u</code>	first component of the velocity.
<code>in</code>	<code>v</code>	second component of the velocity.

Definition at line 88 of file choice\_init\_huv.cpp.

The documentation for this class was generated from the following files:

- Headers/libinitializations/[choice\\_init\\_huv.hpp](#)
- Sources/libinitializations/[choice\\_init\\_huv.cpp](#)

**5.12 Choice\_init\_topo Class Reference**

Choice of initialization for the topography.

```
#include <choice_init_topo.hpp>
```

**Public Member Functions**

- [Choice\\_init\\_topo \(Parameters &\)](#)

*Constructor.*

- void [initialization \(TAB &\)](#)

*Performs the initialization.*

- virtual [~Choice\\_init\\_topo \(\)](#)

*Destructor.*

**5.12.1 Detailed Description**

Choice of initialization for the topography.

Class that calls the initialization of the topography chosen in the parameters file.

Definition at line 84 of file choice\_init\_topo.hpp.

**5.12.2 Constructor & Destructor Documentation****Choice\_init\_topo::Choice\_init\_topo ( Parameters & par )**

Constructor.

Defines the initialization of the topography from the value given in the parameters file.

**Parameters**

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 60 of file choice\_init\_topo.cpp.

**Choice\_init\_topo::~Choice\_init\_topo ( ) [virtual]**

Destructor.

Definition at line 93 of file choice\_init\_topo.cpp.

**5.12.3 Member Function Documentation****void Choice\_init\_topo::initialization ( TAB & topo )**

Performs the initialization.

Calls the initialization of the topography.

**Parameters**

<code>in</code>	<code>topo</code>	topography.
-----------------	-------------------	-------------

Definition at line 82 of file choice\_init\_topo.cpp.

The documentation for this class was generated from the following files:

- Headers/libinitializations/[choice\\_init\\_topo.hpp](#)
- Sources/libinitializations/[choice\\_init\\_topo.cpp](#)

**5.13 Choice\_limiter Class Reference**

Choice of slope limiter.

```
#include <choice_limiter.hpp>
```

**Public Member Functions**

- [Choice\\_limiter](#) (int)  
*Constructor.*
- void [calcul](#) (SCALAR, SCALAR)  
*Calculates the slope limiter.*
- [SCALAR get\\_rec](#) () const  
*Gives the reconstructed value.*
- virtual [~Choice\\_limiter](#) ()  
*Destructor.*

**5.13.1 Detailed Description**

Choice of slope limiter.

Class that calls the slope limiter chosen in the parameters file.

Definition at line 84 of file choice\_limiter.hpp.

**5.13.2 Constructor & Destructor Documentation****Choice\_limiter::Choice\_limiter ( int choice )**

Constructor.

Defines the slope limiter from the value given in the parameters file.



**Parameters**

<i>in</i>	<i>choice</i>	integer that corresponds to the chosen slope limiter.
-----------	---------------	---

Definition at line 60 of file choice\_limiter.cpp.

**Choice\_limiter::~~Choice\_limiter ( ) [virtual]**

Destructor.

Definition at line 104 of file choice\_limiter.cpp.

**5.13.3 Member Function Documentation****void Choice\_limiter::calcul ( SCALAR *a*, SCALAR *b* )**

Calculates the slope limiter.

Calls the calculation of the slope limiter.

**Parameters**

<i>in</i>	<i>a</i>	slope on the left of the cell.
<i>in</i>	<i>b</i>	slope on the right of the cell.

Definition at line 81 of file choice\_limiter.cpp.

**SCALAR Choice\_limiter::get\_rec ( ) const**

Gives the reconstructed value.

Calls the function to get the reconstructed value.

Returns

[Limiter::rec](#) reconstructed value for the chosen slope limiter.

Definition at line 93 of file choice\_limiter.cpp.

The documentation for this class was generated from the following files:

- Headers/liblimitations/[choice\\_limiter.hpp](#)
- Sources/liblimitations/[choice\\_limiter.cpp](#)

**5.14 Choice\_output Class Reference**

Choice of output format.

```
#include <choice_output.hpp>
```

**Public Member Functions**

- [Choice\\_output](#) ([Parameters](#) &)  
*Constructor.*
- void [write](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#), [SCALAR](#))  
*Save the current time.*
- void [check\\_vol](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))  
*Saves the infiltrated and rain volumes.*
- void [result](#) ([SCALAR](#), const [clock\\_t](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), const [SCALAR](#), const int, [SCALAR](#))  
*Saves global values.*
- void [initial](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#))  
*Saves the initial time.*
- void [final](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#))

*Saves the final time.*

- `SCALAR boundaries_flux ( SCALAR, TAB &, TAB &, SCALAR, SCALAR, int, int )`

*Saves the cumulated fluxes on the boundaries.*

- `void boundaries_flux_LR ( SCALAR, TAB )`

*Saves the fluxes on the left and right boundaries.*

- `void boundaries_flux_BT ( SCALAR, TAB )`

*Saves the fluxes on the bottom and top boundaries.*

- `virtual ~Choice_output ( )`

*Destructor.*

### 5.14.1 Detailed Description

Choice of output format.

From the value of the corresponding parameter, calls the savings in the chosen format.

Definition at line 84 of file choice\_output.hpp.

### 5.14.2 Constructor & Destructor Documentation

#### Choice\_output::Choice\_output ( Parameters & par )

Constructor.

Defines the output format from the value given in the parameters file.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 59 of file choice\_output.cpp.

#### Choice\_output::~~Choice\_output ( ) [virtual]

Destructor.

Definition at line 202 of file choice\_output.cpp.

### 5.14.3 Member Function Documentation

#### SCALAR Choice\_output::boundaries\_flux ( SCALAR time, TAB & flux\_u, TAB & flux\_v, SCALAR dt, SCALAR dt\_first, int ORDER, int verif )

Saves the cumulated fluxes on the boundaries.

Calls the saving of the cumulative flux on the boundaries.

Parameters

<code>in</code>	<code>time</code>	current time.
<code>in</code>	<code>flux_u</code>	flux on the left and right boundaries ( $m^2/s$ ).
<code>in</code>	<code>flux_v</code>	flux on the bottom and top boundaries ( $m^2/s$ ).
<code>in</code>	<code>dt</code>	current time step.
<code>in</code>	<code>dt_first</code>	previous time step.
<code>in</code>	<code>ORDER</code>	order of scheme.
<code>in</code>	<code>verif</code>	parameter to know if we removed the computation with the previous time step ( <code>dt_first</code> ).

Definition at line 161 of file choice\_output.cpp.

#### void Choice\_output::boundaries\_flux\_BT ( SCALAR time, TAB BT\_flux )

Saves the fluxes on the bottom and top boundaries.

Calls the saving of the fluxes on the top and bottom boundaries.

**Parameters**

<i>in</i>	<i>time</i>	current time.
<i>in</i>	<i>BT_flux</i>	flux on the bottom and tom boundaries ( $m^2/s$ ).

Definition at line 190 of file choice\_output.cpp.

**void Choice\_output::boundaries\_flux\_LR ( SCALAR *time*, TAB *LR\_flux* )**

Saves the fluxes on the left and right boundaries.

Calls the saving of the fluxes on the left and right boundaries.

**Parameters**

<i>in</i>	<i>time</i>	current time.
<i>in</i>	<i>LR_flux</i>	flux on the left and right boundaries ( $m^2/s$ ).

Definition at line 178 of file choice\_output.cpp.

**void Choice\_output::check\_vol ( SCALAR *time*, SCALAR *dt*, SCALAR *Vol\_rain\_tot*, SCALAR *Vol\_inf*, SCALAR *Vol\_of*, SCALAR *Vol\_bound\_tot* )**

Saves the infiltrated and rain volumes.

Calls the saving of the infiltrated and rain volumes.

**Parameters**

<i>in</i>	<i>time</i>	current time.
<i>in</i>	<i>dt</i>	time step.
<i>in</i>	<i>Vol_rain_tot</i>	total rain volume.
<i>in</i>	<i>Vol_inf</i>	volume of infiltrated water.
<i>in</i>	<i>Vol_of</i>	volume of overland flow.
<i>in</i>	<i>Vol_bound_tot</i>	total volume of water at the boundary.

Definition at line 97 of file choice\_output.cpp.

**void Choice\_output::final ( TAB *z*, TAB *h*, TAB *u*, TAB *v* )**

Saves the final time.

Calls the saving of the final time.

**Parameters**

<i>in</i>	<i>z</i>	topography.
<i>in</i>	<i>h</i>	water height.
<i>in</i>	<i>u</i>	first component of the velocity.
<i>in</i>	<i>v</i>	second component of the velocity.

Definition at line 146 of file choice\_output.cpp.

**void Choice\_output::initial ( TAB *z*, TAB *h*, TAB *u*, TAB *v* )**

Saves the initial time.

Calls the saving of the initial time.

**Parameters**

<i>in</i>	<i>z</i>	topography.
<i>in</i>	<i>h</i>	water height.

in	<i>u</i>	first component of the velocity.
in	<i>v</i>	second component of the velocity.

Definition at line 131 of file choice\_output.cpp.

**void Choice\_output::result ( SCALAR *time*, const clock\_t *cpu*, SCALAR *Vol\_rain*, SCALAR *Vol\_inf*, SCALAR *Vol\_of*, const SCALAR *FROUDE*, const int *NBITER*, SCALAR *vol\_output* )**

Saves global values.

Calls the saving of the global values.

#### Parameters

in	<i>time</i>	elapsed time.
in	<i>cpu</i>	CPU time.
in	<i>Vol_rain</i>	total rain volume.
in	<i>Vol_inf</i>	total volume of infiltrated water.
in	<i>Vol_of</i>	total volume of overland flow.
in	<i>FROUDE</i>	mean Froude number (in space) at the final time.
in	<i>NBITER</i>	number of time steps.
in	<i>vol_output</i>	total outflow volume at the boundary.

Definition at line 113 of file choice\_output.cpp.

**void Choice\_output::write ( TAB *h*, TAB *u*, TAB *v*, TAB *z*, SCALAR *time* )**

Save the current time.

Calls the saving of the current time.

#### Parameters

in	<i>h</i>	water height.
in	<i>u</i>	first component of the velocity.
in	<i>v</i>	second component of the velocity.
in	<i>z</i>	topography.
in	<i>time</i>	value of the current time.

Definition at line 82 of file choice\_output.cpp.

The documentation for this class was generated from the following files:

- Headers/libsave/[choice\\_output.hpp](#)
- Sources/libsave/[choice\\_output.cpp](#)

## 5.15 Choice\_rain Class Reference

Choice of initialization for the rain.

```
#include <choice_rain.hpp>
```

### Public Member Functions

- [Choice\\_rain](#) (Parameters &)  
*Constructor.*
- void [rain\\_func](#) (SCALAR, TAB &)  
*Performs the initialization filling up the table of the rain intensity.*
- virtual [~Choice\\_rain](#) ()  
*Destructor.*

### 5.15.1 Detailed Description

Choice of initialization for the rain.

Class that calls the initialization of the rain chosen in the parameters file.

Definition at line 84 of file choice\_rain.hpp.

### 5.15.2 Constructor & Destructor Documentation

#### Choice\_rain::Choice\_rain ( Parameters & *par* )

Constructor.

Defines the initialization of the rain from the value given in the parameters file.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 59 of file choice\_rain.cpp.

#### Choice\_rain::~~Choice\_rain ( ) [virtual]

Destructor.

Definition at line 94 of file choice\_rain.cpp.

### 5.15.3 Member Function Documentation

#### void Choice\_rain::rain\_func ( SCALAR *time*, TAB & *Tab\_rain* )

Performs the initialization filling up the table of the rain intensity.

Calls the initialization of the rain.

Parameters

<i>in</i>	<i>time</i>	current time.
<i>in</i>	<i>Tab_rain</i>	rain.

Definition at line 83 of file choice\_rain.cpp.

The documentation for this class was generated from the following files:

- Headers/librain\_infiltration/[choice\\_rain.hpp](#)
- Sources/librain\_infiltration/[choice\\_rain.cpp](#)

## 5.16 Choice\_reconstruction Class Reference

Choice of reconstruction.

```
#include <choice_reconstruction.hpp>
```

### Public Member Functions

- [Choice\\_reconstruction](#) (Parameters &, TAB &)

*Constructor.*

- void [calcul](#) (TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &)

*Calculates the second order reconstruction in space.*

- virtual [~Choice\\_reconstruction](#) ()

*Destructor.*

### 5.16.1 Detailed Description

Choice of reconstruction.

Class that calls the reconstruction chosen in the parameters file.

Definition at line 86 of file choice\_reconstruction.hpp.

### 5.16.2 Constructor & Destructor Documentation

#### Choice\_reconstruction::Choice\_reconstruction ( Parameters & *par*, TAB & *z* )

Constructor.

Defines the reconstruction from the value given in the parameters file.

Parameters

in	<i>par</i>	parameter, contains all the values from the parameters file.
in	<i>z</i>	array that represents the topography.

Definition at line 60 of file choice\_reconstruction.cpp.

#### Choice\_reconstruction::~~Choice\_reconstruction ( ) [virtual]

Destructor.

Definition at line 112 of file choice\_reconstruction.cpp.

### 5.16.3 Member Function Documentation

**void Choice\_reconstruction::calcul ( TAB & *h*, TAB & *u*, TAB & *v*, TAB & *z*, TAB & *delzc1*, TAB & *delzc2*, TAB & *delz1*, TAB & *delz2*, TAB & *h1r*, TAB & *u1r*, TAB & *v1r*, TAB & *h1l*, TAB & *u1l*, TAB & *v1l*, TAB & *h2r*, TAB & *u2r*, TAB & *v2r*, TAB & *h2l*, TAB & *u2l*, TAB & *v2l* )**

Calculates the second order reconstruction in space.

Calls the calculation of the second order reconstruction in space.

Parameters

in	<i>h</i>	water height.
in	<i>u</i>	velocity of the flow in the first direction.
in	<i>v</i>	velocity of the flow in the second direction.
in	<i>z</i>	topography.
out	<i>delzc1</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the first direction.
out	<i>delzc2</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the second direction.
out	<i>delz1</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the first direction.
out	<i>delz2</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the second direction.
out	<i>h1r</i>	reconstructed water height on the right of the cell in the first direction.
out	<i>u1r</i>	first component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>v1r</i>	second component of the reconstructed velocity on the right of the cell in the first direction.

out	<i>h1l</i>	reconstructed water height on the left of the cell in the first direction.
out	<i>u1l</i>	first component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>v1l</i>	second component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>h2r</i>	reconstructed water height on the right of the cell in the second direction.
out	<i>u2r</i>	first component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>v2r</i>	second component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>h2l</i>	reconstructed water height on the left of the cell in the second direction.
out	<i>u2l</i>	first component of the reconstructed velocity on the left of the cell in the second direction.
out	<i>v2l</i>	second component of the reconstructed velocity on the left of the cell in the second direction.

Definition at line 82 of file `choice_reconstruction.cpp`.

The documentation for this class was generated from the following files:

- Headers/libreconstructions/[choice\\_reconstruction.hpp](#)
- Sources/libreconstructions/[choice\\_reconstruction.cpp](#)

## 5.17 Choice\_scheme Class Reference

Choice of numerical scheme.

```
#include <choice_scheme.hpp>
```

### Public Member Functions

- [Choice\\_scheme](#) ([Parameters](#) &)  
*Constructor.*
- void [calcul](#) ()  
*Performs the scheme.*
- virtual [~Choice\\_scheme](#) ()  
*Destructor.*

#### 5.17.1 Detailed Description

Choice of numerical scheme.

Class that calls the numerical scheme chosen in the parameters file.

Definition at line 81 of file `choice_scheme.hpp`.

#### 5.17.2 Constructor & Destructor Documentation

##### **Choice\_scheme::Choice\_scheme ( [Parameters](#) & *par* )**

Constructor.

Defines the numerical scheme from the value given in the parameters file.

**Parameters**

---

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 60 of file `choice_scheme.cpp`.

### **Choice\_scheme::~Choice\_scheme ( ) [virtual]**

Destructor.

Definition at line 88 of file `choice_scheme.cpp`.

## 5.17.3 Member Function Documentation

### **void Choice\_scheme::calcul ( )**

Performs the scheme.

Calls the computation of the solution.

Definition at line 78 of file `choice_scheme.cpp`.

The documentation for this class was generated from the following files:

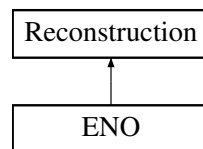
- Headers/libschemas/[choice\\_scheme.hpp](#)
- Sources/libschemas/[choice\\_scheme.cpp](#)

## 5.18 ENO Class Reference

ENO reconstruction

```
#include <eno.hpp>
```

Inheritance diagram for ENO:



### Public Member Functions

- [ENO \(Parameters &, TAB &\)](#)  
*Constructor.*
- void [calcul \(TAB &, TAB &\)](#)  
*Calculates the reconstruction in space.*
- [~ENO \(\)](#)  
*Destructor.*

### Additional Inherited Members

#### 5.18.1 Detailed Description

ENO reconstruction

Class that computes ENO reconstruction in space.

Definition at line 72 of file `eno.hpp`.



### 5.18.2 Constructor & Destructor Documentation

**ENO::ENO ( Parameters & *par*, TAB & *z* )**

Constructor.

Initializations.

**Parameters**

in	<i>par</i>	parameter, contains all the values from the parameters file.
in	<i>z</i>	topography.

Definition at line 59 of file eno.cpp.

**ENO::~~ENO ( )**

Destructor.

Definition at line 370 of file eno.cpp.

**5.18.3 Member Function Documentation**

**void ENO::calcul ( TAB & *h*, TAB & *u*, TAB & *v*, TAB & *z*, TAB & *delzc1*, TAB & *delzc2*, TAB & *delz1*, TAB & *delz2*, TAB & *h1r*, TAB & *u1r*, TAB & *v1r*, TAB & *h1l*, TAB & *u1l*, TAB & *v1l*, TAB & *h2r*, TAB & *u2r*, TAB & *v2r*, TAB & *h2l*, TAB & *u2l*, TAB & *v2l* ) [virtual]**

Calculates the reconstruction in space.

Calls the calculation of the second order reconstruction in space, with ENO formulation, see [Harten et al. \[1986\]](#), [Harten et al. \[1987\]](#), [Shu and Osher \[1988\]](#), [Bouchut \[2004\]](#), [Bouchut \[2007\]](#) .

**Parameters**

in	<i>h</i>	water height.
in	<i>u</i>	velocity of the flow in the first direction.
in	<i>v</i>	velocity of the flow in the second direction.
in	<i>z</i>	topography.
out	<i>delzc1</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the first direction.
out	<i>delzc2</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the second direction.
out	<i>delz1</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the first direction.
out	<i>delz2</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the second direction.
out	<i>h1r</i>	reconstructed water height on the right of the cell in the first direction.
out	<i>u1r</i>	first component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>v1r</i>	second component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>h1l</i>	reconstructed water height on the left of the cell in the first direction.
out	<i>u1l</i>	first component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>v1l</i>	second component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>h2r</i>	reconstructed water height on the right of the cell in the second direction.
out	<i>u2r</i>	first component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>v2r</i>	second component of the reconstructed velocity on the right of the cell in the second direction.

out	$h2l$	reconstructed water height on the left of the cell in the second direction.
out	$u2l$	first component of the reconstructed velocity on the left of the cell in the second direction.
out	$v2l$	second component of the reconstructed velocity on the left of the cell in the second direction.

Implements [Reconstruction](#).

Definition at line 88 of file eno.cpp.

The documentation for this class was generated from the following files:

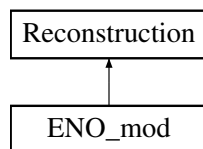
- Headers/libreconstructions/[eno.hpp](#)
- Sources/libreconstructions/[eno.cpp](#)

## 5.19 ENO\_mod Class Reference

Modified ENO reconstruction.

```
#include <eno_mod.hpp>
```

Inheritance diagram for ENO\_mod:



### Public Member Functions

- [ENO\\_mod](#) ([Parameters](#) &, [TAB](#) &)  
*Constructor.*
- void [calcul](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &)  
*Calculates the reconstruction in space.*
- [~ENO\\_mod](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.19.1 Detailed Description

Modified ENO reconstruction.

Class that computes the modified ENO reconstruction in space.

Definition at line 73 of file eno\_mod.hpp.

#### 5.19.2 Constructor & Destructor Documentation

**ENO\_mod::ENO\_mod ( [Parameters](#) & *par*, [TAB](#) & *z* )**

Constructor.

Initializations.

[Parameters](#)

---

in	<i>par</i>	parameter, contains all the values from the parameters file.
in	<i>z</i>	topography.

Definition at line 60 of file eno\_mod.cpp.

### ENO\_mod::~~ENO\_mod ( )

Destructor.

Definition at line 398 of file eno\_mod.cpp.

### 5.19.3 Member Function Documentation

**void ENO\_mod::calcul ( TAB & *h*, TAB & *u*, TAB & *v*, TAB & *z*, TAB & *delzc1*, TAB & *delzc2*, TAB & *delz1*, TAB & *delz2*, TAB & *h1r*, TAB & *u1r*, TAB & *v1r*, TAB & *h1l*, TAB & *u1l*, TAB & *v1l*, TAB & *h2r*, TAB & *u2r*, TAB & *v2r*, TAB & *h2l*, TAB & *u2l*, TAB & *v2l* ) [virtual]**

Calculates the reconstruction in space.

Calls the calculation of the second order reconstruction in space, with a modified ENO formulation, see [Bouchut \[2004\]](#), [Bouchut \[2007\]](#).

Parameters

in	<i>h</i>	water height.
in	<i>u</i>	velocity of the flow in the first direction.
in	<i>v</i>	velocity of the flow in the second direction.
in	<i>z</i>	topography.
out	<i>delzc1</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the first direction.
out	<i>delzc2</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the second direction.
out	<i>delz1</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the first direction.
out	<i>delz2</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the second direction.
out	<i>h1r</i>	reconstructed water height on the right of the cell in the first direction.
out	<i>u1r</i>	first component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>v1r</i>	second component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>h1l</i>	reconstructed water height on the left of the cell in the first direction.
out	<i>u1l</i>	first component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>v1l</i>	second component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>h2r</i>	reconstructed water height on the right of the cell in the second direction.
out	<i>u2r</i>	first component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>v2r</i>	second component of the reconstructed velocity on the right of the cell in the second direction.

out	$h2l$	reconstructed water height on the left of the cell in the second direction.
out	$u2l$	first component of the reconstructed velocity on the left of the cell in the second direction.
out	$v2l$	second component of the reconstructed velocity on the left of the cell in the second direction.

Implements [Reconstruction](#).

Definition at line 85 of file eno\_mod.cpp.

The documentation for this class was generated from the following files:

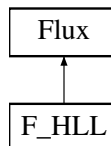
- Headers/libreconstructions/eno\_mod.hpp
- Sources/libreconstructions/eno\_mod.cpp

## 5.20 F\_HLL Class Reference

HLL flux.

```
#include <f_hll.hpp>
```

Inheritance diagram for F\_HLL:



### Public Member Functions

- [F\\_HLL \(\)](#)  
*Constructor.*
- void [calcul \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\)](#)  
*Calculates the numerical flux.*
- virtual [~F\\_HLL \(\)](#)  
*Destructor.*

### Additional Inherited Members

#### 5.20.1 Detailed Description

HLL flux.

Class that computes HLL numerical flux.

Definition at line 72 of file f\_hll.hpp.

#### 5.20.2 Constructor & Destructor Documentation

##### F\_HLL::F\_HLL ( )

Constructor.

Definition at line 59 of file f\_hll.cpp.

##### F\_HLL::~~F\_HLL ( ) [virtual]

Destructor.

Definition at line 134 of file f\_hll.cpp.

### 5.20.3 Member Function Documentation

```
void F_HLL::calcul ( SCALAR h_L, SCALAR u_L, SCALAR v_L, SCALAR h_R, SCALAR u_R,
SCALAR v_R ) [virtual]
```

Calculates the numerical flux.

Recall that this is reduced to a one-dimensional computation along the normal of the mesh edge. If the water heights on the two sides are small or  $c_1 \approx c_2 \approx 0$ , there is no water. Else, HLL formulation is used (see [Bouchut \[2004\]](#)):

$$\mathcal{F}(U_L, U_R) = \begin{cases} F(U_L) & \text{if } 0 < c_1 (\leq c_2), \\ \frac{c_2 F(U_L) - c_1 F(U_R)}{c_2 - c_1} + \frac{c_1 c_2}{c_2 - c_1} (U_R - U_L) & \text{if } c_1 < 0 < c_2, \\ F(U_R) & \text{if } (c_1 \leq) c_2 < 0, \end{cases}$$

with

$$c_1 = \inf_{U=U_L, U_R} \left( \inf_{j \in \{1,2\}} |\lambda_j(U)| \right) \text{ and } c_2 = \sup_{U=U_L, U_R} \left( \sup_{j \in \{1,2\}} |\lambda_j(U)| \right),$$

where  $\lambda_1(U) = u - \sqrt{gh}$  and  $\lambda_2(U) = u + \sqrt{gh}$  are the eigenvalues of the Shallow Water system,  $U = {}^t(h, hu, hv)$  and  $F(U) = {}^t(hu, hu^2 + gh^2/2, hv^2)$ .

Parameters

in	$h\_L$	water height at the left of the interface where the flux is calculated.
in	$u\_L$	velocity (in the x direction) at the left of the interface where the flux is calculated.
in	$v\_L$	velocity (in the y direction) at the left of the interface where the flux is calculated.
in	$h\_R$	water height at the right of the interface where the flux is calculated.
in	$u\_R$	velocity (in the x direction) at the right of the interface where the flux is calculated.
in	$v\_R$	velocity (in the y direction) at the right of the interface where the flux is calculated.

Modifies

- [Flux::f1](#) first component of the numerical flux.
- [Flux::f2](#) second component of the numerical flux.
- [Flux::f3](#) third component of the numerical flux.
- [Flux::cfl](#) value of the CFL.

Note

Long double are used locally in the computation to avoid numerical approximations.

Implements [Flux](#).

Definition at line 62 of file `f_hll.cpp`.

The documentation for this class was generated from the following files:

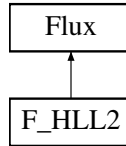
- [Headers/libflux/f\\_hll.hpp](#)
- [Sources/libflux/f\\_hll.cpp](#)

## 5.21 F\_HLL2 Class Reference

HLL flux.

```
#include <f_hll2.hpp>
```

Inheritance diagram for F\_HLL2:



## Public Member Functions

- `F_HLL2 ()`  
*Constructor.*
- `void calcul (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR)`  
*Calculates the numerical flux.*
- `virtual ~F_HLL2 ()`  
*Destructor.*

## Additional Inherited Members

### 5.21.1 Detailed Description

HLL flux.

Class that computes HLL numerical flux with a reduced formulation.

Definition at line 71 of file `f_hll2.hpp`.

### 5.21.2 Constructor & Destructor Documentation

#### `F_HLL2::F_HLL2 ( )`

Constructor.

Definition at line 60 of file `f_hll2.cpp`.

#### `F_HLL2::~~F_HLL2 ( ) [virtual]`

Destructor.

Definition at line 117 of file `f_hll2.cpp`.

### 5.21.3 Member Function Documentation

#### `void F_HLL2::calcul ( SCALAR h_L, SCALAR u_L, SCALAR v_L, SCALAR h_R, SCALAR u_R, SCALAR v_R ) [virtual]`

Calculates the numerical flux.

Recall that this is reduced to a one-dimensional computation along the normal of the mesh edge.

If the water heights on the two sides are small or  $c_1 \approx c_2 \approx 0$ , there is no water.

$$\mathcal{F}(U_L, U_R) = t_1 F(U_R) + t_2 F(U_L) - t_3 (U_R - U_L),$$

with

$$t_1 = \frac{\min(c_2, 0) - \min(c_1, 0)}{c_2 - c_1}, \quad t_2 = 1 - t_1, \quad t_3 = \frac{c_2 |c_1| - c_1 |c_2|}{2(c_2 - c_1)},$$

$$c_1 = \inf_{U=U_L, U_R} \left( \inf_{j \in \{1, 2\}} |\lambda_j(U)| \right) \quad \text{and} \quad c_2 = \sup_{U=U_L, U_R} \left( \sup_{j \in \{1, 2\}} |\lambda_j(U)| \right),$$

where  $\lambda_1(U) = u - \sqrt{gh}$  and  $\lambda_2(U) = u + \sqrt{gh}$  are the eigenvalues of the Shallow Water system,  $U = {}^t(h, hu, hv)$  and  $F(U) = {}^t(hu, hu^2 + gh^2/2, hv^2)$ .

**Parameters**

in	$h_L$	water height at the left of the interface where the flux is calculated.
in	$u_L$	velocity (in the x direction) at the left of the interface where the flux is calculated.
in	$v_L$	velocity (in the y direction) at the left of the interface where the flux is calculated.
in	$h_R$	water height at the right of the interface where the flux is calculated.
in	$u_R$	velocity (in the x direction) at the right of the interface where the flux is calculated.
in	$v_R$	velocity (in the y direction) at the right of the interface where the flux is calculated.

**Modifies**

- [Flux::f1](#) first component of the numerical flux.
- [Flux::f2](#) second component of the numerical flux.
- [Flux::f3](#) third component of the numerical flux.
- [Flux::cfl](#) value of the CFL.

**Note**

Long double are used locally in the computation to avoid numerical approximations.

Implements [Flux](#).

Definition at line 64 of file `f_hll2.cpp`.

The documentation for this class was generated from the following files:

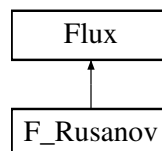
- [Headers/libflux/f\\_hll2.hpp](#)
- [Sources/libflux/f\\_hll2.cpp](#)

**5.22 F\_Rusanov Class Reference**

Rusanov flux.

```
#include <f_rusanov.hpp>
```

Inheritance diagram for `F_Rusanov`:

**Public Member Functions**

- [F\\_Rusanov \(\)](#)  
*Constructor.*
- void [calcul \(SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR\)](#)  
*Calculates the numerical flux.*
- virtual [~F\\_Rusanov \(\)](#)  
*Destructor.*



## Additional Inherited Members

### 5.22.1 Detailed Description

Rusanov flux.

Class that computes Rusanov numerical flux.

Definition at line 71 of file f\_rusanov.hpp.

### 5.22.2 Constructor & Destructor Documentation

#### F\_Rusanov::F\_Rusanov ( )

Constructor.

Definition at line 59 of file f\_rusanov.cpp.

#### F\_Rusanov::~~F\_Rusanov ( ) [virtual]

Destructor.

Definition at line 106 of file f\_rusanov.cpp.

### 5.22.3 Member Function Documentation

#### void F\_Rusanov::calcul ( SCALAR h\_L, SCALAR u\_L, SCALAR v\_L, SCALAR h\_R, SCALAR u\_R, SCALAR v\_R ) [virtual]

Calculates the numerical flux.

Recall that this is reduced to a one-dimensional computation along the normal of the mesh edge.

If the water heights on the two sides are small, there is no water. Else, Rusanov formulation is used (see [Bouchut \[2004\]](#)):

$$\mathcal{F}(U_L, U_R) = \frac{F(U_L) + F(U_R)}{2} - c \frac{U_R - U_L}{2},$$

with  $c = \max(|u_L| + \sqrt{gh_L}, |u_R| + \sqrt{gh_R})$ ,  $U = {}^t(h, hu, hv)$  and  $F(U) = {}^t(hu, hu^2 + gh^2/2, hv^2)$ .

Parameters

in	$h_L$	water height at the left of the interface where the flux is calculated.
in	$u_L$	velocity (in the x direction) at the left of the interface where the flux is calculated.
in	$v_L$	velocity (in the y direction) at the left of the interface where the flux is calculated.
in	$h_R$	water height at the right of the interface where the flux is calculated.
in	$u_R$	velocity (in the x direction) at the right of the interface where the flux is calculated.
in	$v_R$	velocity (in the y direction) at the right of the interface where the flux is calculated.

Modifies

[Flux::f1](#) first component of the numerical flux.

[Flux::f2](#) second component of the numerical flux.

[Flux::f3](#) third component of the numerical flux.

[Flux::cfl](#) value of the CFL.

Implements [Flux](#).

Definition at line 63 of file f\_rusanov.cpp.

The documentation for this class was generated from the following files:

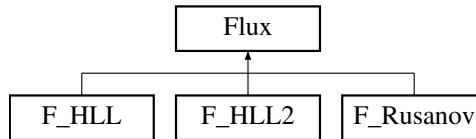
- Headers/libflux/f\_rusanov.hpp
- Sources/libflux/f\_rusanov.cpp

## 5.23 Flux Class Reference

Numerical flux.

```
#include <flux.hpp>
```

Inheritance diagram for Flux:



### Public Member Functions

- `Flux ()`  
*Constructor.*
- virtual void `calcul (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR)=0`  
*Function to be specified in each numerical flux.*
- void `set_tx (SCALAR)`  
*Sets the variable `Flux::tx`.*
- `SCALAR get_f1 () const`  
*Gives the first component of the numerical flux.*
- `SCALAR get_f2 () const`  
*Gives the second component of the numerical flux.*
- `SCALAR get_f3 () const`  
*Gives the third component of the numerical flux.*
- `SCALAR get_cfl () const`  
*Gives the CFL value.*
- virtual `~Flux ()`  
*Destructor.*

### Protected Attributes

- `SCALAR f1`
- `SCALAR f2`
- `SCALAR f3`
- `SCALAR cfl`
- `SCALAR tx`

#### 5.23.1 Detailed Description

Numerical flux.

Class that contains all the common declarations for the numerical fluxes.

Definition at line 68 of file flux.hpp.

#### 5.23.2 Constructor & Destructor Documentation

##### `Flux::Flux ( )`

Constructor.

Definition at line 59 of file flux.cpp.

**Flux::~~Flux( ) [virtual]**

Destructor.

Definition at line 116 of file flux.cpp.

**5.23.3 Member Function Documentation****virtual void Flux::calcul ( SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR ) [pure virtual]**

Function to be specified in each numerical flux.

Implemented in [F\\_HLL](#), [F\\_HLL2](#), and [F\\_Rusanov](#).

**SCALAR Flux::get\_cfl ( ) const**

Gives the CFL value.

Returns

[Flux::cfl](#) value of the CFL.

Definition at line 106 of file flux.cpp.

**SCALAR Flux::get\_f1 ( ) const**

Gives the first component of the numerical flux.

Returns

[Flux::f1](#) first component of the numerical flux.

Definition at line 76 of file flux.cpp.

**SCALAR Flux::get\_f2 ( ) const**

Gives the second component of the numerical flux.

Returns

[Flux::f2](#) second component of the numerical flux.

Definition at line 86 of file flux.cpp.

**SCALAR Flux::get\_f3 ( ) const**

Gives the third component of the numerical flux.

Returns

[Flux::f3](#) third component of the numerical flux.

Definition at line 96 of file flux.cpp.

**void Flux::set\_tx ( SCALAR tx )**

Sets the variable [Flux::tx](#).

Sets the value given in parameter to the variable **tx**.

**Parameters**

<i>in</i>	<i>tx</i>	value of dt/dx.
-----------	-----------	-----------------

Definition at line 66 of file flux.cpp.

**5.23.4 Member Data Documentation****SCALAR Flux::cfl** [**protected**]

CFL value.

Definition at line 105 of file flux.hpp.

**SCALAR Flux::f1** [**protected**]

First component of the numerical flux.

Definition at line 99 of file flux.hpp.

**SCALAR Flux::f2** [**protected**]

Second component of the numerical flux.

Definition at line 101 of file flux.hpp.

**SCALAR Flux::f3** [**protected**]

Third component of the numerical flux.

Definition at line 103 of file flux.hpp.

**SCALAR Flux::tx** [**protected**]

Value of dt/dx.

Definition at line 107 of file flux.hpp.

The documentation for this class was generated from the following files:

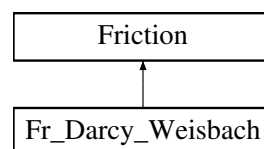
- [Headers/libflux/flux.hpp](#)
- [Sources/libflux/flux.cpp](#)

**5.24 Fr\_Darcy\_Weisbach Class Reference**

Darcy-Weisbach law.

```
#include <fr_darcy_weisbach.hpp>
```

Inheritance diagram for Fr\_Darcy\_Weisbach:

**Public Member Functions**

- [Fr\\_Darcy\\_Weisbach](#) ([Parameters](#) &)  
*Constructor.*
- void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, [SCALAR](#))  
*Calculates the Darcy-Weisbach friction term.*

- void `calculSf` (const `TAB` &, const `TAB` &, const `TAB` &)  
Calculates the explicit Darcy-Weisbach friction term.
- virtual `~Fr_Darcy_Weisbach` ()  
Destructor.

## Additional Inherited Members

### 5.24.1 Detailed Description

Darcy-Weisbach law.

General formulation:  $S_f = \frac{f|U|}{8gh}$ . This term is integrated in the code thanks to a semi-implicit method.

Definition at line 71 of file `fr_darcy_weisbach.hpp`.

### 5.24.2 Constructor & Destructor Documentation

#### `Fr_Darcy_Weisbach::Fr_Darcy_Weisbach ( Parameters & par )`

Constructor.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 59 of file `fr_darcy_weisbach.cpp`.

#### `Fr_Darcy_Weisbach::~~Fr_Darcy_Weisbach ( ) [virtual]`

Destructor.

Definition at line 125 of file `fr_darcy_weisbach.cpp`.

### 5.24.3 Member Function Documentation

#### `void Fr_Darcy_Weisbach::calcul ( const TAB & uold, const TAB & vold, const TAB & hnew, const TAB & q1new, const TAB & q2new, SCALAR dt ) [virtual]`

Calculates the Darcy-Weisbach friction term.

General formulation (see [Smith et al. \[2007\]](#)):  $S_f = \frac{f|U|}{8gh}$ . This term is integrated in the code thanks to a semi-implicit method:

$$q_{1/2_i}^{n+1} = \frac{q_{1/2_i}^*}{1 + dt \frac{f|U_i^n|}{8h_i^{n+1}}}$$

where  $f$  is the friction coefficient.

Parameters

<code>in</code>	<code>uold</code>	velocity in the first direction at the previous time ( $n$ if you are calculating the $n + 1$ th time step), first component of $U_i^n$ in the above formula.
<code>in</code>	<code>vold</code>	velocity in the second direction at the previous time ( $n$ if you are calculating the $n + 1$ th time step), second component of $U_i^n$ in the above formula.
<code>in</code>	<code>hnew</code>	water height after the Shallow-Water computation (without friction), denoted by $h_i^{n+1}$ in the above formula.
<code>in</code>	<code>q1new</code>	discharge in the first direction after the Shallow-Water computation (without friction), denoted by $q_{1_i}^*$ in the above formula.

<code>in</code>	<code>q2new</code>	discharge in the second direction after the Shallow-Water computation (without friction), denoted by $q_{2i}^*$ in the above formula.
<code>in</code>	<code>dt</code>	time step.

#### Modifies

[Friction::q1mod](#) discharge in the first direction modified by the friction term,

[Friction::q2mod](#) discharge in the second direction modified by the friction term.

#### Note

The friction only affects the discharge ( $h^{n+1} = h^*$ ).

Implements [Friction](#).

Definition at line 68 of file `fr_darcy_weisbach.cpp`.

**void Fr\_Darcy\_Weisbach::calculSf ( const TAB & h, const TAB & u, const TAB & v ) [virtual]**

Calculates the explicit Darcy-Weisbach friction term.

Explicit friction term:  $S_f = \frac{fU|U|}{8gh}$  where  $f$  is the friction coefficient.

#### Parameters

<code>in</code>	<code>h</code>	water height.
<code>in</code>	<code>u</code>	velocity in the first direction, first component of $U$ in the above formula.
<code>in</code>	<code>v</code>	velocity in the second direction, second component of $U$ in the above formula.

#### Modifies

[Friction::Sf1](#) explicit friction term in the first direction,

[Friction::Sf2](#) explicit friction term in the second direction.

#### Note

This explicit friction term will be used for erosion.

Implements [Friction](#).

Definition at line 96 of file `fr_darcy_weisbach.cpp`.

The documentation for this class was generated from the following files:

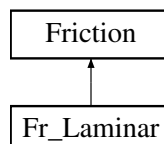
- Headers/libfrictions/[fr\\_darcy\\_weisbach.hpp](#)
- Sources/libfrictions/[fr\\_darcy\\_weisbach.cpp](#)

## 5.25 Fr\_Laminar Class Reference

Laminar law.

```
#include <fr_laminar.hpp>
```

Inheritance diagram for `Fr_Laminar`:



## Public Member Functions

- `Fr_Laminar (Parameters &)`  
*Constructor.*
- void `calcul (const TAB &, const TAB &, const TAB &, const TAB &, const TAB &, SCALAR)`  
*Calculates the laminar friction term.*
- void `calculSf (const TAB &, const TAB &, const TAB &)`  
*Calculates the explicit Manning friction term.*
- virtual `~Fr_Laminar ()`  
*Destructor.*

## Additional Inherited Members

### 5.25.1 Detailed Description

Laminar law.

General formulation:  $S_f = \nu \frac{1}{gh} \frac{U}{h}$ . This term is integrated in the code thanks to an implicit method.

Definition at line 71 of file `fr_laminar.hpp`.

### 5.25.2 Constructor & Destructor Documentation

#### `Fr_Laminar::Fr_Laminar ( Parameters & par )`

Constructor.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 58 of file `fr_laminar.cpp`.

#### `Fr_Laminar::~~Fr_Laminar ( ) [virtual]`

Destructor.

Definition at line 128 of file `fr_laminar.cpp`.

### 5.25.3 Member Function Documentation

#### `void Fr_Laminar::calcul ( const TAB & uold, const TAB & vold, const TAB & hnew, const TAB & q1new, const TAB & q2new, SCALAR dt ) [virtual]`

Calculates the laminar friction term.

General formulation:  $S_f = \nu \frac{1}{gh} \frac{U}{h}$ . This term is integrated in the code thanks to an implicit method:

$$q_{1/2_i}^{n+1} = \frac{q_{1/2_i}^*}{1 + \nu dt \frac{1}{(h_i^{n+1})^2}}$$

where  $\nu$  is the friction coefficient.

Parameters

<code>in</code>	<code>uold</code>	velocity in the first direction at the previous time ( $n$ if you are calculating the $n + 1$ th time step), first component of $U_i^n$ in the above formula.
-----------------	-------------------	---

in	<i>vold</i>	velocity in the second direction at the previous time ( <i>n</i> if you are calculating the <i>n</i> + 1th time step), second component of $U_i^n$ in the above formula.
in	<i>hnew</i>	water height after the Shallow-Water computation (without friction), denoted by $h_i^{n+1}$ in the above formula.
in	<i>q1new</i>	discharge in the first direction after the Shallow-Water computation (without friction), denoted by $q1_i^*$ in the above formula.
in	<i>q2new</i>	discharge in the second direction after the Shallow-Water computation (without friction), denoted by $q2_i^*$ in the above formula.
in	<i>dt</i>	time step.

#### Modifies

[Friction::q1mod](#) discharge in the first direction modified by the friction term,  
[Friction::q2mod](#) discharge in the second direction modified by the friction term.

#### Note

The friction only affects the discharge (  $h^{n+1} = h^*$  ).

Implements [Friction](#).

Definition at line 67 of file `fr_laminar.cpp`.

**void Fr\_Laminar::calculSf ( const TAB & h, const TAB & u, const TAB & v ) [virtual]**

Calculates the explicit Manning friction term.

Explicit friction term:  $S_f = \nu \frac{1}{gh} \frac{U}{h}$  where *nu* is the friction coefficient.

#### Parameters

in	<i>h</i>	water height.
in	<i>u</i>	velocity in the first direction, first component of $U$ in the above formula.
in	<i>v</i>	velocity in the second direction, second component of $U$ in the above formula.

#### Modifies

[Friction::Sf1](#) explicit friction term in the first direction,  
[Friction::Sf2](#) explicit friction term in the second direction.

#### Note

This explicit friction term will be used for erosion.

Implements [Friction](#).

Definition at line 98 of file `fr_laminar.cpp`.

The documentation for this class was generated from the following files:

- Headers/libfrictions/[fr\\_laminar.hpp](#)
- Sources/libfrictions/[fr\\_laminar.cpp](#)

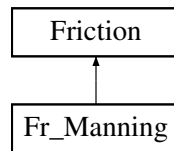
## 5.26 Fr\_Manning Class Reference

Manning law.

```
#include <fr_manning.hpp>
```

Inheritance diagram for `Fr_Manning`:





## Public Member Functions

- `Fr_Manning (Parameters &)`  
*Constructor.*
- `void calcul (const TAB &, const TAB &, const TAB &, const TAB &, const TAB &, SCALAR)`  
*Calculates the Manning friction term.*
- `void calculSf (const TAB &, const TAB &, const TAB &)`  
*Calculates the explicit Manning friction term.*
- `virtual ~Fr_Manning ()`  
*Destructor.*

## Additional Inherited Members

### 5.26.1 Detailed Description

Manning law.

General formulation:  $S_f = c^2 \frac{U|U|}{h^{4/3}}$ . This term is integrated in the code thanks to a semi-implicit method.  
Definition at line 72 of file `fr_manning.hpp`.

### 5.26.2 Constructor & Destructor Documentation

#### `Fr_Manning::Fr_Manning ( Parameters & par )`

Constructor.  
Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 59 of file `fr_manning.cpp`.

#### `Fr_Manning::~Fr_Manning ( ) [virtual]`

Destructor.

Definition at line 126 of file `fr_manning.cpp`.

### 5.26.3 Member Function Documentation

#### `void Fr_Manning::calcul ( const TAB & uold, const TAB & vold, const TAB & hnew, const TAB & q1new, const TAB & q2new, SCALAR dt ) [virtual]`

Calculates the Manning friction term.

General formulation (see [Smith et al. \[2007\]](#)):  $S_f = c^2 \frac{U|U|}{h^{4/3}}$ . This term is integrated in the code thanks to a semi-implicit method:

$$q_{1/2_i}^{n+1} = \frac{q_{1/2_i}^*}{1 + dt \frac{c^2 g |U_i^n|}{(h_i^{n+1})^{4/3}}}$$

where  $c$  is the friction coefficient.

## Parameters

in	<i>uold</i>	velocity in the first direction at the previous time ( <i>n</i> if you are calculating the <i>n</i> + 1th time step), first component of $U_i^n$ in the above formula.
in	<i>vold</i>	velocity in the second direction at the previous time ( <i>n</i> if you are calculating the <i>n</i> + 1th time step), second component of $U_i^n$ in the above formula.
in	<i>hnew</i>	water height after the Shallow-Water computation (without friction), denoted by $h_i^{n+1}$ in the above formula.
in	<i>q1new</i>	discharge in the first direction after the Shallow-Water computation (without friction), denoted by $q_{1i}^*$ in the above formula.
in	<i>q2new</i>	discharge in the second direction after the Shallow-Water computation (without friction), denoted by $q_{2i}^*$ in the above formula.
in	<i>dt</i>	time step.

## Modifies

[Friction::q1mod](#) discharge in the first direction modified by the friction term,  
[Friction::q2mod](#) discharge in the second direction modified by the friction term.

## Note

The friction only affects the discharge (  $h^{n+1} = h^*$  ).

Implements [Friction](#).

Definition at line 68 of file `fr_manning.cpp`.

**void Fr\_Manning::calculSf ( const TAB & h, const TAB & u, const TAB & v ) [virtual]**

Calculates the explicit Manning friction term.

Explicit friction term:  $S_f = c^2 \frac{U|U|}{h^{4/3}}$  where *c* is the friction coefficient.

## Parameters

in	<i>h</i>	water height.
in	<i>u</i>	velocity in the first direction, first component of $U$ in the above formula.
in	<i>v</i>	velocity in the second direction, second component of $U$ in the above formula.

## Modifies

[Friction::Sf1](#) explicit friction term in the first direction,  
[Friction::Sf2](#) explicit friction term in the second direction.

## Note

This explicit friction term will be used for erosion.

Implements [Friction](#).

Definition at line 97 of file `fr_manning.cpp`.

The documentation for this class was generated from the following files:

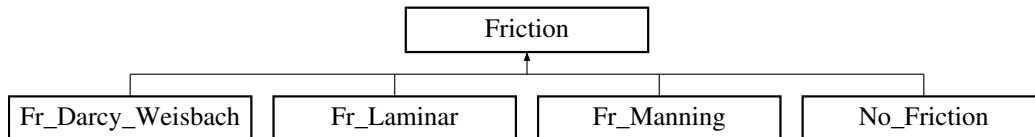
- [Headers/libfrictions/fr\\_manning.hpp](#)
- [Sources/libfrictions/fr\\_manning.cpp](#)

## 5.27 Friction Class Reference

Friction law

```
#include <friction.hpp>
```

Inheritance diagram for Friction:



### Public Member Functions

- [Friction](#) (Parameters &)

*Constructor.*

- virtual void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, const [TAB](#) &, [SCALAR](#))=0

*Function to be specified in each friction law.*

- virtual [TAB](#) [get\\_q1mod](#) () const

*Gives the discharge in the first direction modified by the friction term.*

- virtual [TAB](#) [get\\_q2mod](#) () const

*Gives the discharge in the second direction modified by the friction term.*

- virtual void [calculSf](#) (const [TAB](#) &, const [TAB](#) &, const [TAB](#) &)=0

*Calculates the explicit friction term. It will be used for computations with erosion.*

- virtual [TAB](#) [get\\_Sf1](#) () const

*Gives the explicit friction term in the first direction.*

- virtual [TAB](#) [get\\_Sf2](#) () const

*Gives the explicit friction term in the second direction.*

- virtual [~Friction](#) ()

*Destructor.*

### Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)
- [TAB](#) [q1mod](#)
- [TAB](#) [q2mod](#)
- [TAB](#) [Sf1](#)
- [TAB](#) [Sf2](#)
- [TAB](#) [Fric\\_tab](#)

#### 5.27.1 Detailed Description

Friction law

Class that contains all the common declarations for the friction law. The friction is computed with a semi-implicit method.

Definition at line 72 of file friction.hpp.

### 5.27.2 Constructor & Destructor Documentation

#### Friction::Friction ( Parameters & *par* )

Constructor.

Defines the number of cells, the space steps and initializes [Friction::Fric\\_tab](#), [Friction::q1mod](#), [Friction::q2mod](#), [Friction::Sf1](#), [Friction::Sf2](#).

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Warning

\*\*\*: ERROR: the value at the point \*\*\*.

Initialization of [Friction](#)

Definition at line 59 of file friction.cpp.

#### Friction::~~Friction ( ) [virtual]

Destructor.

Deallocation of [Friction::Fric\\_tab](#), [Friction::q1mod](#), [Friction::q2mod](#), [Friction::Sf1](#), [Friction::Sf2](#)

Definition at line 152 of file friction.cpp.

### 5.27.3 Member Function Documentation

#### virtual void Friction::calcul ( const TAB & , const TAB & , const TAB & , const TAB & , const TAB & , SCALAR ) [pure virtual]

Function to be specified in each friction law.

Implemented in [Fr\\_Manning](#), [Fr\\_Darcy\\_Weisbach](#), [Fr\\_Laminar](#), and [No\\_Friction](#).

#### virtual void Friction::calculSf ( const TAB & , const TAB & , const TAB & ) [pure virtual]

Calculates the explicit friction term. It will be used for computations with erosion.

Implemented in [Fr\\_Manning](#), [Fr\\_Darcy\\_Weisbach](#), [Fr\\_Laminar](#), and [No\\_Friction](#).

#### TAB Friction::get\_q1mod ( ) const [virtual]

Gives the discharge in the first direction modified by the friction term.

Returns

[Friction::q1mod](#) discharge in the first direction modified by the friction term.

Definition at line 112 of file friction.cpp.

#### TAB Friction::get\_q2mod ( ) const [virtual]

Gives the discharge in the second direction modified by the friction term.

Returns

[Friction::q2mod](#) discharge in the second direction modified by the friction term.

Definition at line 122 of file friction.cpp.

**TAB Friction::get\_Sf1 ( ) const [virtual]**

Gives the explicit friction term in the first direction.

Returns

[Friction::Sf1](#) explicit friction term in the first direction.

Definition at line 132 of file friction.cpp.

**TAB Friction::get\_Sf2 ( ) const [virtual]**

Gives the explicit friction term in the second direction.

Returns

[Friction::Sf2](#) explicit friction term in the second direction.

Definition at line 142 of file friction.cpp.

**5.27.4 Member Data Documentation****const SCALAR Friction::DX [protected]**

Space step in the first (x) direction.

Definition at line 106 of file friction.hpp.

**const SCALAR Friction::DY [protected]**

Space step in the second (y) direction.

Definition at line 108 of file friction.hpp.

**TAB Friction::Fric\_tab [protected]**

Array that contains the friction coefficient by cell.

Definition at line 119 of file friction.hpp.

**const int Friction::NXCELL [protected]**

Number of cells in space in the first (x) direction.

Definition at line 102 of file friction.hpp.

**const int Friction::NYCELL [protected]**

Number of cells in space in the second (y) direction.

Definition at line 104 of file friction.hpp.

**TAB Friction::q1mod [protected]**

Discharge in the first direction modified by the friction term.

Definition at line 111 of file friction.hpp.

**TAB Friction::q2mod [protected]**

Discharge in the second direction modified by the friction term.

Definition at line 113 of file friction.hpp.

**TAB Friction::Sf1** [protected]

Explicit friction term in the first direction.

Definition at line 115 of file friction.hpp.

**TAB Friction::Sf2** [protected]

Explicit friction term in the second direction.

Definition at line 117 of file friction.hpp.

The documentation for this class was generated from the following files:

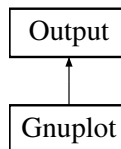
- Headers/libfrictions/[friction.hpp](#)
- Sources/libfrictions/[friction.cpp](#)

## 5.28 Gnuplot Class Reference

Gnuplot output

```
#include <gnuplot.hpp>
```

Inheritance diagram for Gnuplot:



### Public Member Functions

- [Gnuplot](#) ([Parameters](#) &)  
*Constructor.*
- void [write](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#), [SCALAR](#))  
*Saves one time step.*
- virtual [~Gnuplot](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.28.1 Detailed Description

Gnuplot output

Class that writes the result in the output file with a structure optimized for Gnuplot.

Definition at line 73 of file gnuplot.hpp.

#### 5.28.2 Constructor & Destructor Documentation

##### Gnuplot::Gnuplot ( [Parameters](#) & *par* )

Constructor.

Writes the header of the file 'huz\_evolution.dat'.

[Parameters](#)

---

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

**Warning**

Impossible to open the `***` file. Verify if the directory `***` exists.

**Note**

If `huz_evolution.dat` cannot be opened, the code will exit with failure termination code.

Definition at line 60 of file `gnuplot.cpp`.

**Gnuplot::~Gnuplot ( ) [virtual]****Destructor.**

Definition at line 125 of file `gnuplot.cpp`.

**5.28.3 Member Function Documentation****void Gnuplot::write ( TAB *h*, TAB *u*, TAB *v*, TAB *z*, SCALAR *time* ) [virtual]**

Saves one time step.

Writes the values of [Scheme::h](#), [Scheme::u](#) ( $=q1/h$ ), [Scheme::v](#) ( $=q2/h$ ), [Scheme::h](#) + [Scheme::z](#) (free surface), [Scheme::z](#),  $|U| = \sqrt{u^2 + v^2}$ , the Froude number  $\frac{|U|}{\sqrt{gh}}$ , [Scheme::q1](#), [Scheme::q2](#), and  $h|U|$  at the current time in `huz_evolution.dat`.

If the water height is too small, we replace it by 0, the velocity is null and the Froude number does not exist.

**Parameters**

<code>in</code>	<code>h</code>	the water height.
<code>in</code>	<code>u</code>	first component of the velocity.
<code>in</code>	<code>v</code>	second component of the velocity.
<code>in</code>	<code>z</code>	the topography.
<code>in</code>	<code>time</code>	the current time.

Implements [Output](#).

Definition at line 89 of file `gnuplot.cpp`.

The documentation for this class was generated from the following files:

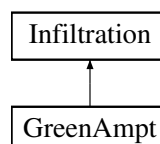
- [Headers/libsave/gnuplot.hpp](#)
- [Sources/libsave/gnuplot.cpp](#)

**5.29 GreenAmpt Class Reference**

Green-Ampt law.

```
#include <greenampt.hpp>
```

Inheritance diagram for `GreenAmpt`:



## Public Member Functions

- [GreenAmpt](#) ([Parameters](#) &)  
*Constructor.*
- [SCALAR capacity](#) (const [SCALAR](#), const [SCALAR](#), const [SCALAR](#), const [SCALAR](#) Kc, const [SCALAR](#) Ks, const [SCALAR](#) dtheta, const [SCALAR](#) Psi, const [SCALAR](#) zcrust)  
*Calculates the infiltration capacity.*
- void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))  
*Calculates the infiltrated volume.*
- virtual [~GreenAmpt](#) ()  
*Destructor.*

## Additional Inherited Members

### 5.29.1 Detailed Description

Green-Ampt law.

Class that computes the infiltrated volume and modified water height with Green-Ampt 1d law.

Definition at line 72 of file greenampt.hpp.

### 5.29.2 Constructor & Destructor Documentation

#### [GreenAmpt::GreenAmpt](#) ( [Parameters](#) & *par* )

Constructor.

Initializes the values for Green-Ampt infiltration.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Warning

\*\*\*: ERROR: the value at the point \*\*\*.

Definition at line 59 of file greenampt.cpp.

#### [GreenAmpt::~~GreenAmpt](#) ( ) [[virtual](#)]

Destructor.

Definition at line 306 of file greenampt.cpp.

### 5.29.3 Member Function Documentation

#### [void GreenAmpt::calcul](#) ( const [TAB](#) & *h*, const [TAB](#) & *Vin\_tot*, const [SCALAR](#) *dt* ) [[virtual](#)]

Calculates the infiltrated volume.

Parameters

<code>in</code>	<code>h</code>	water height.
<code>in</code>	<code>Vin_tot</code>	total infiltrated volume.
<code>in</code>	<code>dt</code>	time step.

Modifies

`infiltration::hmod` modified water height.

`infiltration::Vin` total infiltrated volume containing the current time step.

Implements [Infiltration](#).

Definition at line 260 of file greenampt.cpp.



**SCALAR GreenAmp::capacity** ( const SCALAR *h*, const SCALAR *Vin\_tot*, const SCALAR *dt*, const SCALAR *Kc*, const SCALAR *Ks*, const SCALAR *dtheta*, const SCALAR *Psi*, const SCALAR *zcrust* )

Calculates the infiltration capacity.

the infiltration capacity is given by:

$$I_C = \begin{cases} K_s \left(1 + \frac{Psi + h}{Z_f}\right) & \text{if } z_{crust} = 0 \\ K_c \left(1 + \frac{Psi + h}{Z_f}\right) & \text{if } Z_f \leq z_{crust} , \\ K_e \left(1 + \frac{Psi + h}{Z_f}\right) & \end{cases}$$

with the effective hydraulic conductivity

$$K_e = \frac{1}{\frac{1}{K_s} * \left(1 - \frac{z_{crust} * dtheta}{V_{in,tot}}\right) + z_{crust} * \frac{dtheta}{V_{in,tot}} * \frac{1}{K_c}}$$

#### Parameters

in	<i>h</i>	water height.
in	<i>Vin_tot</i>	total infiltrated volume.
in	<i>dt</i>	time step.
in	<i>Kc</i>	hydraulic conductivity of the (upper) crust.
in	<i>Ks</i>	hydraulic conductivity of the (lower) soil.
in	<i>dtheta</i>	water content.
in	<i>Psi</i>	load pressure.
in	<i>zcrust</i>	thickness of the (upper) crust.

#### Returns

**ic**: infiltration capacity.

Definition at line 216 of file greenamp.cpp.

The documentation for this class was generated from the following files:

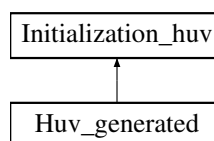
- Headers/librain\_infiltration/greenamp.hpp
- Sources/librain\_infiltration/greenamp.cpp

## 5.30 Huv\_generated Class Reference

No water configuration.

```
#include <huv_generated.hpp>
```

Inheritance diagram for Huv\_generated:



#### Public Member Functions

- [Huv\\_generated](#) (Parameters &)  
*Constructor.*

- void `initialization` ( `TAB &`, `TAB &`, `TAB &` )  
*Performs the initialization.*
- virtual `~Huv_generated` ( )  
*Destructor.*

## Additional Inherited Members

### 5.30.1 Detailed Description

No water configuration.

Class that initializes the water height and the velocity for a dry domain.

Definition at line 73 of file `huv_generated.hpp`.

### 5.30.2 Constructor & Destructor Documentation

#### `Huv_generated::Huv_generated ( Parameters & par )`

Constructor.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
-----------------	------------------	---

Definition at line 60 of file `huv_generated.cpp`.

#### `Huv_generated::~~Huv_generated ( ) [virtual]`

Destructor.

Definition at line 86 of file `huv_generated.cpp`.

### 5.30.3 Member Function Documentation

#### `void Huv_generated::initialization ( TAB & h, TAB & u, TAB & v ) [virtual]`

Performs the initialization.

Initializes the water height and the velocity at 0.

Parameters

<code>in, out</code>	<code>h</code>	water height.
<code>in, out</code>	<code>u</code>	first component of the velocity.
<code>in, out</code>	<code>v</code>	second component of the velocity.

Implements `Initialization_huv`.

Definition at line 67 of file `huv_generated.cpp`.

The documentation for this class was generated from the following files:

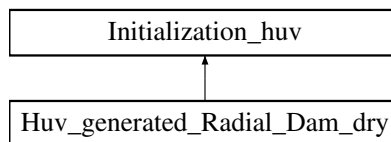
- [Headers/libinitializations/huv\\_generated.hpp](#)
- [Sources/libinitializations/huv\\_generated.cpp](#)

## 5.31 Huv\_generated\_Radial\_Dam\_dry Class Reference

Dry radial dam break configuration.

```
#include <huv_generated_radial_dam_dry.hpp>
```

Inheritance diagram for `Huv_generated_Radial_Dam_dry`:



## Public Member Functions

- [Huv\\_generated\\_Radial\\_Dam\\_dry \(Parameters &\)](#)  
*Constructor.*
- void [initialization \(TAB &, TAB &, TAB &\)](#)  
*Performs the initialization.*
- virtual [~Huv\\_generated\\_Radial\\_Dam\\_dry \(\)](#)  
*Destructor.*

## Additional Inherited Members

### 5.31.1 Detailed Description

Dry radial dam break configuration.

Class that initializes the water height and the velocity for a radial dam break on a dry domain.

Definition at line 73 of file `huv_generated_radial_dam_dry.hpp`.

### 5.31.2 Constructor & Destructor Documentation

#### **Huv\_generated\_Radial\_Dam\_dry::Huv\_generated\_Radial\_Dam\_dry ( Parameters & *par* )**

Constructor.

Defines the position of the dam (half of the domain), the water height before the dam (5 millimeters), the water height after the dam (0 meter) and the velocity (0 m/s), see [Goutal and Maurel \[1997\]](#), [Audusse et al. \[2000\]](#).

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
-----------------	------------------	---

Definition at line 60 of file `huv_generated_radial_dam_dry.cpp`.

#### **Huv\_generated\_Radial\_Dam\_dry::~~Huv\_generated\_Radial\_Dam\_dry ( ) [virtual]**

Destructor.

Definition at line 100 of file `huv_generated_radial_dam_dry.cpp`.

### 5.31.3 Member Function Documentation

#### **void Huv\_generated\_Radial\_Dam\_dry::initialization ( TAB & *h*, TAB & *u*, TAB & *v* ) [virtual]**

Performs the initialization.

Initializes the water height and the velocity, before and after the dam.

Parameters

<code>in, out</code>	<code>h</code>	water height.
<code>in, out</code>	<code>u</code>	first component of the velocity.

<code>in, out</code>	<code>v</code>	second component of the velocity.
----------------------	----------------	-----------------------------------

Implements [Initialization\\_huv](#).

Definition at line 79 of file `huv_generated_radial_dam_dry.cpp`.

The documentation for this class was generated from the following files:

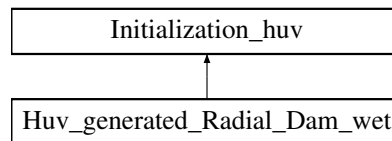
- [Headers/libinitializations/huv\\_generated\\_radial\\_dam\\_dry.hpp](#)
- [Sources/libinitializations/huv\\_generated\\_radial\\_dam\\_dry.cpp](#)

## 5.32 Huv\_generated\_Radial\_Dam\_wet Class Reference

Wet radial dam break configuration.

```
#include <huv_generated_radial_dam_wet.hpp>
```

Inheritance diagram for `Huv_generated_Radial_Dam_wet`:



### Public Member Functions

- [Huv\\_generated\\_Radial\\_Dam\\_wet](#) ([Parameters](#) &)  
*Constructor.*
- void [initialization](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &)  
*Performs the initialization.*
- virtual [~Huv\\_generated\\_Radial\\_Dam\\_wet](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.32.1 Detailed Description

Wet radial dam break configuration.

Class for the initialization of the water height and velocity for a radial dam break on a wet domain.

Definition at line 74 of file `huv_generated_radial_dam_wet.hpp`.

#### 5.32.2 Constructor & Destructor Documentation

**`Huv_generated_Radial_Dam_wet::Huv_generated_Radial_Dam_wet ( Parameters & par )`**

Constructor.

Defines the position of the dam (half of the domain), the water height before the dam (5 millimeters), the water height after the dam (4 millimeter) and the velocity (0 m/s), see [Goutal and Maurel \[1997\]](#), [Audusse et al. \[2000\]](#).

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
-----------------	------------------	---

Definition at line 60 of file `huv_generated_radial_dam_wet.cpp`.

**`Huv_generated_Radial_Dam_wet::~~Huv_generated_Radial_Dam_wet ( ) [virtual]`**

Destructor.

Definition at line 100 of file `huv_generated_radial_dam_wet.cpp`.

### 5.32.3 Member Function Documentation

**void Huv\_generated\_Radial\_Dam\_wet::initialization ( TAB & *h*, TAB & *u*, TAB & *v* ) [virtual]**

Performs the initialization.

Initializes the water height and the velocity, before and after the dam.

**Parameters**

in, out	<i>h</i>	water height.
in, out	<i>u</i>	first component of the velocity.
in, out	<i>v</i>	second component of the velocity.

Implements [Initialization\\_huv](#).

Definition at line 79 of file `huv_generated_radial_dam_wet.cpp`.

The documentation for this class was generated from the following files:

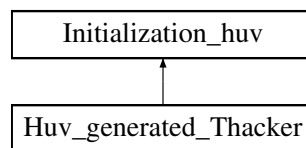
- [Headers/libinitializations/huv\\_generated\\_radial\\_dam\\_wet.hpp](#)
- [Sources/libinitializations/huv\\_generated\\_radial\\_dam\\_wet.cpp](#)

## 5.33 Huv\_generated\_Thacker Class Reference

Thacker configuration.

```
#include <huv_generated_thacker.hpp>
```

Inheritance diagram for Huv\_generated\_Thacker:



### Public Member Functions

- [Huv\\_generated\\_Thacker \(Parameters &\)](#)  
*Constructor.*
- void [initialization \(TAB &, TAB &, TAB &\)](#)  
*Performs the initialization.*
- virtual [~Huv\\_generated\\_Thacker \(\)](#)  
*Destructor.*

### Additional Inherited Members

#### 5.33.1 Detailed Description

Thacker configuration.

Class that initializes the water height and the velocity for Thacker's benchmark.

Definition at line 74 of file `huv_generated_thacker.hpp`.

#### 5.33.2 Constructor & Destructor Documentation

**Huv\_generated\_Thacker::Huv\_generated\_Thacker ( Parameters & *par* )**

Constructor.

Defines the characteristics of the paraboloid.

**Parameters**

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file (unused).
-----------	------------	---

Definition at line 60 of file `huv_generated_thacker.cpp`.

**Huv\_generated\_Thacker::~~Huv\_generated\_Thacker ( ) [virtual]**

Destructor.

Definition at line 106 of file `huv_generated_thacker.cpp`.

**5.33.3 Member Function Documentation****void Huv\_generated\_Thacker::initialization ( TAB & h, TAB & u, TAB & v ) [virtual]**

Performs the initialization.

Initializes the water height to a plane surface and the velocity to zero, see [Thacker \[1981\]](#).

**Parameters**

<i>in, out</i>	<i>h</i>	water height.
<i>in, out</i>	<i>u</i>	first component of the velocity.
<i>in, out</i>	<i>v</i>	second component of the velocity.

Implements [Initialization\\_huv](#).

Definition at line 80 of file `huv_generated_thacker.cpp`.

The documentation for this class was generated from the following files:

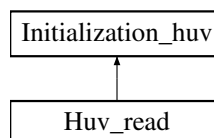
- [Headers/libinitializations/huv\\_generated\\_thacker.hpp](#)
- [Sources/libinitializations/huv\\_generated\\_thacker.cpp](#)

**5.34 Huv\_read Class Reference**

File configuration.

```
#include <huv_read.hpp>
```

Inheritance diagram for `Huv_read`:

**Public Member Functions**

- [Huv\\_read \(Parameters &\)](#)  
*Constructor.*
- void [initialization \(TAB &, TAB &, TAB &\)](#)  
*Performs the initialization.*
- virtual [~Huv\\_read \(\)](#)  
*Destructor.*

**Additional Inherited Members****5.34.1 Detailed Description**

File configuration.

Class that initializes the water height and of the velocity to the values read in a file.

Definition at line 72 of file `huv_read.hpp`.

### 5.34.2 Constructor & Destructor Documentation

#### Huv\_read::Huv\_read ( Parameters & *par* )

Constructor.

Defines the name of the file for the initialization.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 60 of file huv\_read.cpp.

#### Huv\_read::~~Huv\_read ( ) [virtual]

Destructor.

Definition at line 201 of file huv\_read.cpp.

### 5.34.3 Member Function Documentation

#### void Huv\_read::initialization ( TAB & *h*, TAB & *u*, TAB & *v* ) [virtual]

Performs the initialization.

Initializes the water height and the velocity to the values read in the corresponding file.

Parameters

<i>in, out</i>	<i>h</i>	water height.
<i>in, out</i>	<i>u</i>	first component of the velocity.
<i>in, out</i>	<i>v</i>	second component of the velocity.

Warning

(huv\_namefile): ERROR: cannot open the huv file.  
 (huv\_namefile): ERROR: the number of data in this file is too big  
 (huv\_namefile): ERROR: line \*\*\*.  
 (huv\_namefile): WARNING: line \*\*\*.  
 (huv\_namefile): ERROR: the number of data in this file is too small  
 (huv\_namefile): ERROR: the value for the point x \*\*\* y \*\*\* is missing

Note

If the file cannot be opened or if the data are not correct, the code will exit with failure termination code.

Implements [Initialization\\_huv](#).

Definition at line 72 of file huv\_read.cpp.

The documentation for this class was generated from the following files:

- Headers/libinitializations/[huv\\_read.hpp](#)
- Sources/libinitializations/[huv\\_read.cpp](#)

## 5.35 Hydrostatic Class Reference

Hydrostatic reconstruction

```
#include <hydrostatic.hpp>
```

## Public Member Functions

- [Hydrostatic \(\)](#)  
*Constructor.*
- void [calcul \(SCALAR, SCALAR, SCALAR\)](#)  
*Calculates the hydrostatic reconstruction.*
- [SCALAR get\\_hhydro\\_l \(\)](#)  
*Gives the reconstructed water height on the left.*
- [SCALAR get\\_hhydro\\_r \(\)](#)  
*Gives the reconstructed water height on the right.*
- virtual [~Hydrostatic \(\)](#)  
*Destructor.*

## Protected Attributes

- [SCALAR hl\\_rec](#)
- [SCALAR hr\\_rec](#)

### 5.35.1 Detailed Description

Hydrostatic reconstruction

Class that computes the hydrostatic reconstruction.  
Definition at line 67 of file hydrostatic.hpp.

### 5.35.2 Constructor & Destructor Documentation

#### Hydrostatic::Hydrostatic ( )

Constructor.

Definition at line 60 of file hydrostatic.cpp.

#### Hydrostatic::~~Hydrostatic ( ) [**virtual**]

Destructor.

Definition at line 102 of file hydrostatic.cpp.

### 5.35.3 Member Function Documentation

#### void Hydrostatic::calcul ( SCALAR *hl*, SCALAR *hr*, SCALAR *dz* )

Calculates the hydrostatic reconstruction.

See [Audusse et al. \[2004\]](#) for more details.

Parameters

<i>in</i>	<i>hl</i>	water height on the cell located at the left of the boundary.
<i>in</i>	<i>hr</i>	water height on the cell located at the right of the boundary.
<i>in</i>	<i>dz</i>	Difference between the values of the topography of the two adjacent cells.

Modifies

[Hydrostatic::hl\\_rec](#), set to  $(hl - \max(0, dz))_+$ .  
[Hydrostatic::hr\\_rec](#), set to  $(hr - \max(0, -dz))_+$ .

Definition at line 63 of file hydrostatic.cpp.



**SCALAR Hydrostatic::get\_hhydro\_l ( )**

Gives the reconstructed water height on the left.

Returns

[Hydrostatic::hl\\_rec](#) Hydrostatic reconstruction on the left.

Definition at line 81 of file hydrostatic.cpp.

**SCALAR Hydrostatic::get\_hhydro\_r ( )**

Gives the reconstructed water height on the right.

Returns

[Hydrostatic::hr\\_rec](#) Hydrostatic reconstruction on the right.

Definition at line 92 of file hydrostatic.cpp.

**5.35.4 Member Data Documentation****SCALAR Hydrostatic::hl\_rec [protected]**

Hydrostatic reconstruction on the left

Definition at line 87 of file hydrostatic.hpp.

**SCALAR Hydrostatic::hr\_rec [protected]**

Hydrostatic reconstruction on the right

Definition at line 89 of file hydrostatic.hpp.

The documentation for this class was generated from the following files:

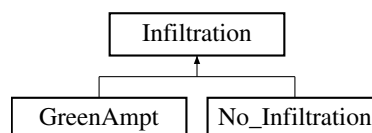
- [Headers/libreconstructions/hydrostatic.hpp](#)
- [Sources/libreconstructions/hydrostatic.cpp](#)

**5.36 Infiltration Class Reference**

Definition of infiltration law.

```
#include <infiltration.hpp>
```

Inheritance diagram for Infiltration:

**Public Member Functions**

- [Infiltration \(Parameters &\)](#)  
*Constructor.*
- virtual void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))=0  
*Function to be specified in each case.*
- [TAB get\\_hmod](#) ( ) const  
*Gives the modified valued of the water height.*

- [TAB get\\_Vin \(\)](#) const  
*Gives the infiltrated volume.*
- virtual [~Infiltration \(\)](#)  
*Destructor.*

### Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)
- [TAB hmod](#)
- [TAB Vin](#)

### 5.36.1 Detailed Description

Definition of infiltration law.

Class that contains all the common declarations for the infiltration law.

Definition at line 71 of file infiltration.hpp.

### 5.36.2 Constructor & Destructor Documentation

#### **Infiltration::Infiltration ( Parameters & *par* )**

Constructor.

Defines the number of cells, the space step and initializes [Infiltration::hmod](#) and [Infiltration::Vin](#).

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 60 of file infiltration.cpp.

#### **Infiltration::~~Infiltration ( ) [virtual]**

Destructor.

Definition at line 103 of file infiltration.cpp.

### 5.36.3 Member Function Documentation

#### **virtual void Infiltration::calcul ( const TAB & , const TAB & , const SCALAR ) [pure virtual]**

Function to be specified in each case.

Implemented in [GreenAmpt](#), and [No\\_Infiltration](#).

#### **TAB Infiltration::get\_hmod ( ) const**

Gives the modified valued of the water height.

Returns

The value of [Infiltration::hmod](#).

Definition at line 83 of file infiltration.cpp.

**TAB Infiltration::get\_Vin ( ) const**

Gives the infiltrated volume.

Returns

The value of [Infiltration::Vin](#).

Definition at line 93 of file infiltration.cpp.

**5.36.4 Member Data Documentation****const SCALAR Infiltration::DX [protected]**

Space step in the first (x) direction.

Definition at line 96 of file infiltration.hpp.

**const SCALAR Infiltration::DY [protected]**

Space step in the second (y) direction.

Definition at line 98 of file infiltration.hpp.

**TAB Infiltration::hmod [protected]**

Modified valued of the water height

Definition at line 100 of file infiltration.hpp.

**const int Infiltration::NXCELL [protected]**

Number of cells in space in the first (x) direction.

Definition at line 92 of file infiltration.hpp.

**const int Infiltration::NYCELL [protected]**

Number of cells in space in the second (y) direction.

Definition at line 94 of file infiltration.hpp.

**TAB Infiltration::Vin [protected]**

Infiltrated volume

Definition at line 102 of file infiltration.hpp.

The documentation for this class was generated from the following files:

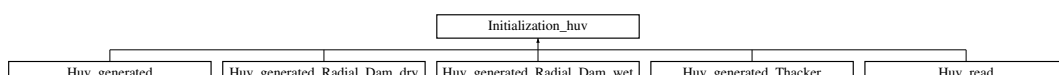
- [Headers/librain\\_infiltration/infiltration.hpp](#)
- [Sources/librain\\_infiltration/infiltration.cpp](#)

**5.37 Initialization\_huv Class Reference**

Initialization of h, u and v.

```
#include <initialization_huv.hpp>
```

Inheritance diagram for Initialization\_huv:



## Public Member Functions

- [Initialization\\_huv](#) ([Parameters](#) &)  
*Constructor.*
- virtual void [initialization](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &)=0  
*Function to be specified in each initialization.*
- virtual [~Initialization\\_huv](#) ()  
*Destructor.*

## Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)

### 5.37.1 Detailed Description

Initialization of h, u and v.

Class that contains all the common declarations for the initialization of the water height and of the velocity.  
Definition at line 71 of file `initialization_huv.hpp`.

### 5.37.2 Constructor & Destructor Documentation

#### `Initialization_huv::Initialization_huv ( Parameters & par )`

Constructor.

Defines the numbers of cells and the space steps.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 59 of file `initialization_huv.cpp`.

#### `Initialization_huv::~~Initialization_huv ( ) [virtual]`

Destructor.

Definition at line 70 of file `initialization_huv.cpp`.

### 5.37.3 Member Function Documentation

#### `virtual void Initialization_huv::initialization ( TAB &, TAB &, TAB & ) [pure virtual]`

Function to be specified in each initialization.

Implemented in [Huv\\_generated\\_Radial\\_Dam\\_wet](#), [Huv\\_generated\\_Thacker](#), [Huv\\_generated\\_Radial\\_Dam\\_dry](#), [Huv\\_generated](#), and [Huv\\_read](#).

### 5.37.4 Member Data Documentation

#### `const SCALAR Initialization_huv::DX [protected]`

Space step in the x direction.

Definition at line 90 of file `initialization_huv.hpp`.

**const SCALAR Initialization\_huv::DY [protected]**

Space step in the y direction.

Definition at line 92 of file initialization\_huv.hpp.

**const int Initialization\_huv::NXCELL [protected]**

Number of cells in space in the x direction.

Definition at line 86 of file initialization\_huv.hpp.

**const int Initialization\_huv::NYCELL [protected]**

Number of cells in space in the y direction.

Definition at line 88 of file initialization\_huv.hpp.

The documentation for this class was generated from the following files:

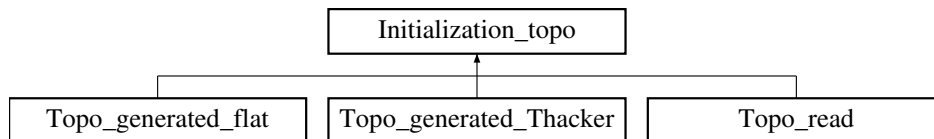
- Headers/libinitializations/[initialization\\_huv.hpp](#)
- Sources/libinitializations/[initialization\\_huv.cpp](#)

## 5.38 Initialization\_topo Class Reference

Initialization of z.

```
#include <initialization_topo.hpp>
```

Inheritance diagram for Initialization\_topo:



### Public Member Functions

- [Initialization\\_topo](#) ([Parameters](#) &)  
*Constructor.*
- virtual void [initialization](#) ([TAB](#) &)=0  
*Function to be specified in each initialization.*
- virtual [~Initialization\\_topo](#) ()  
*Destructor.*

### Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)

#### 5.38.1 Detailed Description

Initialization of z.

Class that contains all the common declarations for the initialization of the topography.

Definition at line 71 of file initialization\_topo.hpp.

### 5.38.2 Constructor & Destructor Documentation

#### Initialization\_topo::Initialization\_topo ( Parameters & *par* )

Constructor.

Defines the numbers of cells and the space steps.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 59 of file initialization\_topo.cpp.

#### Initialization\_topo::~~Initialization\_topo ( ) [virtual]

Destructor.

Definition at line 70 of file initialization\_topo.cpp.

### 5.38.3 Member Function Documentation

#### virtual void Initialization\_topo::initialization ( TAB & ) [pure virtual]

Function to be specified in each initialization.

Implemented in [Topo\\_generated\\_flat](#), [Topo\\_generated\\_Thacker](#), and [Topo\\_read](#).

### 5.38.4 Member Data Documentation

#### const SCALAR Initialization\_topo::DX [protected]

Space step in the x direction.

Definition at line 91 of file initialization\_topo.hpp.

#### const SCALAR Initialization\_topo::DY [protected]

Space step in the y direction.

Definition at line 93 of file initialization\_topo.hpp.

#### const int Initialization\_topo::NXCELL [protected]

Number of cells in space in the x direction.

Definition at line 87 of file initialization\_topo.hpp.

#### const int Initialization\_topo::NYCELL [protected]

Number of cells in space in the y direction.

Definition at line 89 of file initialization\_topo.hpp.

The documentation for this class was generated from the following files:

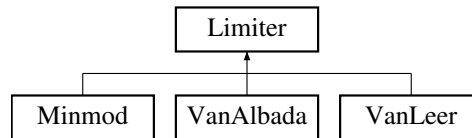
- Headers/libinitializations/[initialization\\_topo.hpp](#)
- Sources/libinitializations/[initialization\\_topo.cpp](#)

## 5.39 Limiter Class Reference

Slope limiter.

```
#include <limiter.hpp>
```

Inheritance diagram for Limiter:



## Public Member Functions

- [Limiter](#) ()  
*Constructor.*
- virtual void [calcul](#) (SCALAR, SCALAR)=0  
*Function to be specified in each slope limiter.*
- [SCALAR get\\_rec](#) () const  
*Gives the reconstructed value.*
- virtual [~Limiter](#) ()  
*Destructor.*

## Protected Attributes

- [SCALAR rec](#)

### 5.39.1 Detailed Description

Slope limiter.

Class that contains all the common declarations for the slope limiters.

Definition at line 71 of file limiter.hpp.

### 5.39.2 Constructor & Destructor Documentation

#### **Limiter::Limiter** ( )

Constructor.

Definition at line 59 of file limiter.cpp.

#### **Limiter::~~Limiter** ( ) [**virtual**]

Destructor.

Definition at line 73 of file limiter.cpp.

### 5.39.3 Member Function Documentation

#### **virtual void Limiter::calcul** ( SCALAR , SCALAR ) [**pure virtual**]

Function to be specified in each slope limiter.

Implemented in [Minmod](#), [VanAlbada](#), and [VanLeer](#).

#### **SCALAR Limiter::get\_rec** ( ) const

Gives the reconstructed value.

Returns

[Limiter::rec](#) reconstructed value.

Definition at line 63 of file limiter.cpp.

### 5.39.4 Member Data Documentation

#### SCALAR Limiter::rec [protected]

Reconstructed value

Definition at line 90 of file limiter.hpp.

The documentation for this class was generated from the following files:

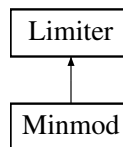
- Headers/liblimitations/limiter.hpp
- Sources/liblimitations/limiter.cpp

## 5.40 Minmod Class Reference

Minmod slope limiter

```
#include <minmod.hpp>
```

Inheritance diagram for Minmod:



### Public Member Functions

- [Minmod](#) ()  
*Constructor.*
- void [calcul](#) (SCALAR, SCALAR)  
*Calculates the value of the slope limiter.*
- virtual [~Minmod](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.40.1 Detailed Description

Minmod slope limiter

Class that calculates the minmod slope limiter.

Definition at line 71 of file minmod.hpp.

#### 5.40.2 Constructor & Destructor Documentation

##### Minmod::Minmod ( )

Constructor.

Definition at line 59 of file minmod.cpp.

##### Minmod::~~Minmod ( ) [virtual]

Destructor.

Definition at line 88 of file minmod.cpp.



### 5.40.3 Member Function Documentation

**void Minmod::calcul ( SCALAR a, SCALAR b ) [virtual]**

Calculates the value of the slope limiter.

Minmod function:

$$\text{minmod}(x,y) = \begin{cases} \min(x,y) & \text{if } x,y \geq 0, \\ \max(x,y) & \text{if } x,y \leq 0, \\ 0 & \text{else.} \end{cases}$$

#### Parameters

in	<i>a</i>	slope on the left of the cell.
in	<i>b</i>	slope on the right of the cell.

#### Modifies

[Limiter::rec](#) reconstructed value.

Implements [Limiter](#).

Definition at line 62 of file minmod.cpp.

The documentation for this class was generated from the following files:

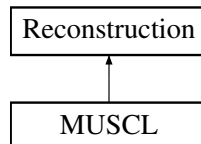
- Headers/liblimitations/[minmod.hpp](#)
- Sources/liblimitations/[minmod.cpp](#)

## 5.41 MUSCL Class Reference

MUSCL reconstruction

```
#include <muscl.hpp>
```

Inheritance diagram for MUSCL:



### Public Member Functions

- [MUSCL](#) ([Parameters](#) &, [TAB](#) &)  
*Constructor.*
- void [calcul](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &)  
*Calculates the reconstruction in space.*
- [~MUSCL](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.41.1 Detailed Description

MUSCL reconstruction

Class that computes MUSCL reconstruction in space.

Definition at line 72 of file muscl.hpp.

### 5.41.2 Constructor & Destructor Documentation

#### MUSCL::MUSCL ( Parameters & *par*, TAB & *z* )

Constructor.

Initializations.

Parameters

in	<i>par</i>	parameter, contains all the values from the parameters file.
in	<i>z</i>	topography.

Definition at line 60 of file muscl.cpp.

#### MUSCL::~~MUSCL ( )

Destructor.

Definition at line 225 of file muscl.cpp.

### 5.41.3 Member Function Documentation

**void MUSCL::calcul ( TAB & *h*, TAB & *u*, TAB & *v*, TAB & *z*, TAB & *delzc1*, TAB & *delzc2*, TAB & *delz1*, TAB & *delz2*, TAB & *h1r*, TAB & *u1r*, TAB & *v1r*, TAB & *h1l*, TAB & *u1l*, TAB & *v1l*, TAB & *h2r*, TAB & *u2r*, TAB & *v2r*, TAB & *h2l*, TAB & *u2l*, TAB & *v2l* ) [virtual]**

Calculates the reconstruction in space.

Calls the calculation of the second order reconstruction in space with MUSCL formulation, see [van Leer \[1979\]](#) [Bouchut \[2007\]](#).

Parameters

in	<i>h</i>	water height.
in	<i>u</i>	velocity of the flow in the first direction.
in	<i>v</i>	velocity of the flow in the second direction.
in	<i>z</i>	topography.
out	<i>delzc1</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the first direction.
out	<i>delzc2</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell in the second direction.
out	<i>delz1</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the first direction.
out	<i>delz2</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells) in the second direction.
out	<i>h1r</i>	reconstructed water height on the right of the cell in the first direction.
out	<i>u1r</i>	first component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>v1r</i>	second component of the reconstructed velocity on the right of the cell in the first direction.
out	<i>h1l</i>	reconstructed water height on the left of the cell in the first direction.
out	<i>u1l</i>	first component of the reconstructed velocity on the left of the cell in the first direction.
out	<i>v1l</i>	second component of the reconstructed velocity on the left of the cell in the first direction.

out	<i>h2r</i>	reconstructed water height on the right of the cell in the second direction.
out	<i>u2r</i>	first component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>v2r</i>	second component of the reconstructed velocity on the right of the cell in the second direction.
out	<i>h2l</i>	reconstructed water height on the left of the cell in the second direction.
out	<i>u2l</i>	first component of the reconstructed velocity on the left of the cell in the second direction.
out	<i>v2l</i>	second component of the reconstructed velocity on the left of the cell in the second direction.

Implements [Reconstruction](#).

Definition at line 72 of file muscl.cpp.

The documentation for this class was generated from the following files:

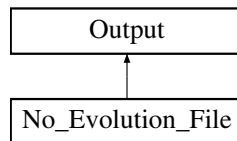
- Headers/libreconstructions/[muscl.hpp](#)
- Sources/libreconstructions/[muscl.cpp](#)

## 5.42 No\_Evolution\_File Class Reference

No output.

```
#include <no_evolution_file.hpp>
```

Inheritance diagram for No\_Evolution\_File:



### Public Member Functions

- [No\\_Evolution\\_File](#) ([Parameters](#) &)  
*Constructor.*
- void [write](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#), [SCALAR](#))  
*Saves one time step: nothing to do.*
- virtual [~No\\_Evolution\\_File](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.42.1 Detailed Description

No output.

No output files with time evolution are created.

Definition at line 69 of file no\_evolution\_file.hpp.

#### 5.42.2 Constructor & Destructor Documentation

**No\_Evolution\_File::No\_Evolution\_File** ( [Parameters](#) & *par* )

Constructor.

**Parameters**

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
-----------------	------------------	---

Definition at line 58 of file `no_evolution_file.cpp`.

**No\_Evolution\_File::~No\_Evolution\_File ( ) [virtual]**

Destructor.

Definition at line 88 of file `no_evolution_file.cpp`.

**5.42.3 Member Function Documentation****void No\_Evolution\_File::write ( TAB *h*, TAB *u*, TAB *v*, TAB *z*, SCALAR *time* ) [virtual]**

Saves one time step: nothing to do.

Does nothing.

**Parameters**

<code>in</code>	<code>h</code>	the water height (unused).
<code>in</code>	<code>u</code>	first component of the velocity (unused).
<code>in</code>	<code>v</code>	second component of the velocity (unused).
<code>in</code>	<code>z</code>	the topography (unused).
<code>in</code>	<code>time</code>	the current time (unused).

Implements [Output](#).

Definition at line 67 of file `no_evolution_file.cpp`.

The documentation for this class was generated from the following files:

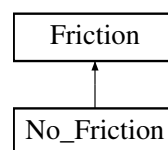
- Headers/libsave/[no\\_evolution\\_file.hpp](#)
- Sources/libsave/[no\\_evolution\\_file.cpp](#)

**5.43 No\_Friction Class Reference**

No friction.

```
#include <no_friction.hpp>
```

Inheritance diagram for `No_Friction`:

**Public Member Functions**

- [No\\_Friction \(Parameters &\)](#)  
*Constructor.*
- void [calcul](#) (const [TAB &](#), const [TAB &](#), const [TAB &](#), const [TAB &](#), const [TAB &](#), [SCALAR](#))  
*Does no calculation.*
- void [calculSf](#) (const [TAB &](#), const [TAB &](#), const [TAB &](#))  
*Return the friction term equal to zero.*
- virtual [~No\\_Friction \(\)](#)  
*Destructor.*

## Additional Inherited Members

### 5.43.1 Detailed Description

No friction.

Does no computation.

Definition at line 71 of file no\_friction.hpp.

### 5.43.2 Constructor & Destructor Documentation

#### No\_Friction::No\_Friction ( Parameters & *par* )

Constructor.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 59 of file no\_friction.cpp.

#### No\_Friction::~~No\_Friction ( ) [virtual]

Destructor.

Definition at line 123 of file no\_friction.cpp.

### 5.43.3 Member Function Documentation

#### void No\_Friction::calcul ( const TAB & *uold*, const TAB & *vold*, const TAB & *hnew*, const TAB & *q1new*, const TAB & *q2new*, SCALAR *dt* ) [virtual]

Does no calculation.

No computation (no friction).

Parameters

<i>in</i>	<i>uold</i>	velocity in the first direction at the previous time ( <i>n</i> if you are calculating the <i>n</i> + 1th time step) (unused).
<i>in</i>	<i>vold</i>	velocity in the second direction at the previous time ( <i>n</i> if you are calculating the <i>n</i> + 1th time step) (unused).
<i>in</i>	<i>hnew</i>	water height after the Shallow-Water computation (without friction) (unused).
<i>in</i>	<i>q1new</i>	discharge in the first direction after the Shallow-Water computation (without friction) (unused).
<i>in</i>	<i>q2new</i>	discharge in the second direction after the Shallow-Water computation (without friction) (unused).
<i>in</i>	<i>dt</i>	time step (unused).

Modifies

[Friction::q1mod](#) discharge in the first direction modified by the friction term,

[Friction::q2mod](#) discharge in the second direction modified by the friction term.

Implements [Friction](#).

Definition at line 68 of file no\_friction.cpp.

#### void No\_Friction::calculSf ( const TAB & *h*, const TAB & *u*, const TAB & *v* ) [virtual]

Return the friction term equal to zero.

Explicit friction term:  $S_f = 0$ .

**Parameters**

<code>in</code>	<code>h</code>	water height (unused).
<code>in</code>	<code>u</code>	velocity in the first direction (unused).
<code>in</code>	<code>v</code>	velocity in the second direction (unused).

**Modifies**

[Friction::Sf1](#) explicit friction term in the first direction,  
[Friction::Sf2](#) explicit friction term in the second direction.

**Note**

This explicit friction term will be used for erosion.

Implements [Friction](#).

Definition at line 97 of file `no_friction.cpp`.

The documentation for this class was generated from the following files:

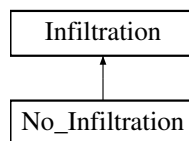
- [Headers/libfrictions/no\\_friction.hpp](#)
- [Sources/libfrictions/no\\_friction.cpp](#)

**5.44 No\_Infiltration Class Reference**

No infiltration.

```
#include <no_infiltration.hpp>
```

Inheritance diagram for `No_Infiltration`:

**Public Member Functions**

- [No\\_Infiltration](#) ([Parameters](#) &)  
*Constructor.*
- void [calcul](#) (const [TAB](#) &, const [TAB](#) &, const [SCALAR](#))  
*Does no infiltration.*
- virtual [~No\\_Infiltration](#) ()  
*Destructor.*

**Additional Inherited Members****5.44.1 Detailed Description**

No infiltration.

The water height and infiltrated volume remain unchanged.

Definition at line 70 of file `no_infiltration.hpp`.

**5.44.2 Constructor & Destructor Documentation**

**`No_Infiltration::No_Infiltration ( Parameters & par )`**

Constructor.

**Parameters**

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
-----------------	------------------	---

Definition at line 58 of file no\_infiltration.cpp.

**No\_Infiltration::~~No\_Infiltration ( ) [virtual]**

Destructor.

Definition at line 90 of file no\_infiltration.cpp.

**5.44.3 Member Function Documentation****void No\_Infiltration::calcul ( const TAB & h, const TAB & Vin\_tot, const SCALAR dt ) [virtual]**

Does no infiltration.

No computation (water height and infiltrated volume remain unchanged).

**Parameters**

<code>in</code>	<code>h</code>	water height.
<code>in</code>	<code>Vin_tot</code>	total infiltrated volume.
<code>in</code>	<code>dt</code>	time step (unused).

**Modifies**

[Infiltration::hmod](#) modified water height.

[Infiltration::Vin](#) total infiltrated volume containing the current time step.

Implements [Infiltration](#).

Definition at line 67 of file no\_infiltration.cpp.

The documentation for this class was generated from the following files:

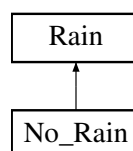
- Headers/librain\_infiltration/no\_infiltration.hpp
- Sources/librain\_infiltration/no\_infiltration.cpp

**5.45 No\_Rain Class Reference**

No rain.

```
#include <no_rain.hpp>
```

Inheritance diagram for No\_Rain:

**Public Member Functions**

- [No\\_Rain](#) ([Parameters](#) &)  
*Constructor.*
- void [rain\\_func](#) ([SCALAR](#), [TAB](#) &)  
*Sets the rain intensity to zero.*
- virtual [~No\\_Rain](#) ()  
*Destructor.*

## Additional Inherited Members

### 5.45.1 Detailed Description

No rain.

Sets the rain intensity to zero.

Definition at line 70 of file no\_rain.hpp.

### 5.45.2 Constructor & Destructor Documentation

#### No\_Rain::No\_Rain ( Parameters & *par* )

Constructor.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file (unused).
-----------	------------	---

Definition at line 59 of file no\_rain.cpp.

#### No\_Rain::~~No\_Rain ( ) [virtual]

Destructor.

Definition at line 85 of file no\_rain.cpp.

### 5.45.3 Member Function Documentation

#### void No\_Rain::rain\_func ( SCALAR *time*, TAB & *Tab\_rain* ) [virtual]

Sets the rain intensity to zero.

No computation (rain intensity set to zero).

Parameters

<i>in</i>	<i>time</i>	current time (unused).
<i>in, out</i>	<i>Tab_rain</i>	rain intensity at the current time on each cell.

Implements [Rain](#).

Definition at line 68 of file no\_rain.cpp.

The documentation for this class was generated from the following files:

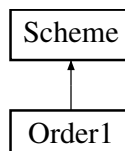
- [Headers/librain\\_infiltration/no\\_rain.hpp](#)
- [Sources/librain\\_infiltration/no\\_rain.cpp](#)

## 5.46 Order1 Class Reference

Order 1 scheme.

```
#include <order1.hpp>
```

Inheritance diagram for Order1:





## Public Member Functions

- [Order1 \(Parameters &\)](#)  
*Constructor.*
- void [calcul \(\)](#)  
*Performs the numerical scheme.*
- virtual [~Order1 \(\)](#)  
*Destructor.*

## Additional Inherited Members

### 5.46.1 Detailed Description

Order 1 scheme.

Class that computes the solution with a numerical scheme at order 1.  
Definition at line 71 of file order1.hpp.

### 5.46.2 Constructor & Destructor Documentation

#### Order1::Order1 ( Parameters & *par* )

Constructor.  
Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 59 of file order1.cpp.

#### Order1::~~Order1 ( ) [**virtual**]

Destructor.  
Definition at line 251 of file order1.cpp.

### 5.46.3 Member Function Documentation

#### void Order1::calcul ( ) [**virtual**]

Performs the numerical scheme.  
Performs the first order numerical scheme.

Note

In DEBUG mode, the programme will save four other files with boundary fluxes and volumes of water.

Warning

order1: WARNING: the computation finished because the maximum number of time steps was reached  
(see MAX\_ITER in [misc.hpp](#))

Implements [Scheme](#).

Definition at line 69 of file order1.cpp.

The documentation for this class was generated from the following files:

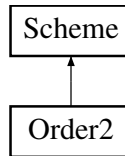
- [Headers/libschemas/order1.hpp](#)
- [Sources/libschemas/order1.cpp](#)

## 5.47 Order2 Class Reference

Order 2 scheme.

```
#include <order2.hpp>
```

Inheritance diagram for Order2:



### Public Member Functions

- [Order2 \(Parameters &\)](#)  
*Constructor.*
- void [calcul \(\)](#)  
*Performs the numerical scheme.*
- virtual [~Order2 \(\)](#)  
*Destructor.*

### Additional Inherited Members

#### 5.47.1 Detailed Description

Order 2 scheme.

Class that computes the solution with a numerical scheme at order 2.

Definition at line 71 of file order2.hpp.

#### 5.47.2 Constructor & Destructor Documentation

##### Order2::Order2 ( Parameters & *par* )

Constructor.

Initializations, definition of the reconstruction and creation of 3 vectors for this reconstruction.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 59 of file order2.cpp.

##### Order2::~~Order2 ( ) [virtual]

Destructor.

Definition at line 97 of file order2.cpp.

#### 5.47.3 Member Function Documentation

##### void Order2::calcul ( ) [virtual]

Performs the numerical scheme.

Performs the second order numerical scheme.

Note

In DEBUG mode, the programme will save another file with volumes of water.

## Warning

order2: WARNING: the computation finished because the maximum number of time steps was reached (see MAX\_ITER in [misc.hpp](#))

Implements [Scheme](#).

Definition at line 124 of file order2.cpp.

The documentation for this class was generated from the following files:

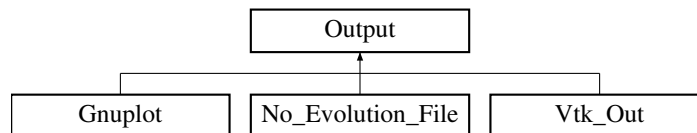
- [Headers/libschemas/order2.hpp](#)
- [Sources/libschemas/order2.cpp](#)

## 5.48 Output Class Reference

Output format

```
#include <output.hpp>
```

Inheritance diagram for Output:



### Public Member Functions

- [Output](#) ([Parameters](#) &)  
*Constructor.*
- virtual void [write](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#), [SCALAR](#))=0  
*Function to be specified in each output format.*
- void [check\\_vol](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))  
*Saves the infiltrated and rain volumes.*
- void [result](#) ([SCALAR](#), const [clock\\_t](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), const [SCALAR](#), const int, [SCALAR](#))  
*Saves global values.*
- void [initial](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#))  
*Saves the initial time.*
- void [final](#) ([TAB](#), [TAB](#), [TAB](#), [TAB](#))  
*Saves the final time.*
- [SCALAR](#) [boundaries\\_flux](#) ([SCALAR](#), [TAB](#) &, [TAB](#) &, [SCALAR](#), [SCALAR](#), int, int)  
*Saves the cumulated fluxes on the boundaries.*
- void [boundaries\\_flux\\_LR](#) ([SCALAR](#), [TAB](#))  
*Saves the fluxes on the left and right boundaries.*
- void [boundaries\\_flux\\_BT](#) ([SCALAR](#), [TAB](#))  
*Saves the fluxes on the bottom and top boundaries.*
- virtual [~Output](#) ()  
*Destructor.*

## Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)
- string [outputDirectory](#)
- string [namefile\\_check\\_volume](#)
- string [namefile\\_res](#)
- string [namefile\\_init](#)
- string [namefile\\_final](#)
- string [namefile\\_Bound\\_flux](#)
- string [namefile\\_Bound\\_flux\\_BT](#)
- string [namefile\\_Bound\\_flux\\_LR](#)

### 5.48.1 Detailed Description

Output format

Class that contains all the common declarations for the output formats.

Definition at line 70 of file output.hpp.

### 5.48.2 Constructor & Destructor Documentation

#### Output::Output ( Parameters & *par* )

Constructor.

Defines the names of the outputs.

If run in DEBUG mode, writes the header of the file 'boundaries\_flux.dat', 'check\_vol.dat', 'flux\_boundaries\_B←T.dat' and 'flux\_boundaries\_LR.dat'.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Warning

Impossible to open the \*\*\* file. Verify if the directory \*\*\* exists.

Note

If 'boundaries\_flux.dat', 'check\_vol.dat', 'flux\_boundaries\_BT.dat' or 'flux\_boundaries\_LR.dat' cannot be opened, the code will exit with failure termination code.

Definition at line 59 of file output.cpp.

#### Output::~~Output ( ) [**virtual**]

Destructor.

Definition at line 382 of file output.cpp.

### 5.48.3 Member Function Documentation

#### SCALAR Output::boundaries\_flux ( SCALAR *time*, TAB & *flux\_u*, TAB & *flux\_v*, SCALAR *dt*, SCALAR *dt\_first*, int *ORDER*, int *verif* )

Saves the cumulated fluxes on the boundaries.

**Parameters**

in	<i>time</i>	current time.
in	<i>flux_u</i>	flux on the left and right boundaries ( $m^2/s$ ).
in	<i>flux_v</i>	flux on the bottom and top boundaries ( $m^2/s$ ).
in	<i>dt</i>	current time step.
in	<i>dt_first</i>	previous time step.
in	<i>ORDER</i>	order of scheme.
in	<i>verif</i>	parameter to know if we removed the computation with the previous time step ( <i>dt_first</i> ).

Definition at line 262 of file output.cpp.

**void Output::boundaries\_flux\_BT ( SCALAR *time*, TAB *BT\_flux* )**

Saves the fluxes on the bottom and top boundaries.

**Parameters**

in	<i>time</i>	current time.
in	<i>BT_flux</i>	flux on the bottom and tom boundaries ( $m^2/s$ ).

Definition at line 323 of file output.cpp.

**void Output::boundaries\_flux\_LR ( SCALAR *time*, TAB *LR\_flux* )**

Saves the fluxes on the left and right boundaries.

**Parameters**

in	<i>time</i>	current time.
in	<i>LR_flux</i>	flux on the left and right boundaries ( $m^2/s$ ).

Definition at line 305 of file output.cpp.

**void Output::check\_vol ( SCALAR *time*, SCALAR *dt*, SCALAR *Vol\_rain\_tot*, SCALAR *Vol\_inf*, SCALAR *Vol\_of*, SCALAR *Vol\_bound\_tot* )**

Saves the infiltrated and rain volumes.

**Parameters**

in	<i>time</i>	current time (unused).
in	<i>dt</i>	time step.
in	<i>Vol_rain_tot</i>	total rain volume.
in	<i>Vol_inf</i>	volume of infiltrated water.
in	<i>Vol_of</i>	volume of overland flow.
in	<i>Vol_bound_tot</i>	total volume of water at the boundary.

Definition at line 189 of file output.cpp.

**void Output::final ( TAB *z*, TAB *h*, TAB *u*, TAB *v* )**

Saves the final time.

If the water height is too small, we replace it by 0, the velocities and discharge are null and the Froude number does not exist.

**Parameters**


---

<i>in</i>	<i>z</i>	topography.
<i>in</i>	<i>h</i>	water height.
<i>in</i>	<i>u</i>	first component of the velocity.
<i>in</i>	<i>v</i>	second component of the velocity.

**Warning**

Impossible to open the \*\*\* file. Verify if the directory \*\*\* exists.

**Note**

If huz\_final.dat cannot be opened, the code will exit with failure termination code.

Definition at line 341 of file output.cpp.

**void Output::initial ( TAB *z*, TAB *h*, TAB *u*, TAB *v* )**

Saves the initial time.

**Parameters**

<i>in</i>	<i>z</i>	topography.
<i>in</i>	<i>h</i>	water height.
<i>in</i>	<i>u</i>	first component of the velocity.
<i>in</i>	<i>v</i>	second component of the velocity.

**Warning**

Impossible to open the \*\*\* file. Verify if the directory \*\*\* exists.

**Note**

If huz\_initial.dat cannot be opened, the code will exit with failure termination code.

Definition at line 154 of file output.cpp.

**void Output::result ( SCALAR *time*, const clock\_t *cpu*, SCALAR *Vol\_rain*, SCALAR *Vol\_inf*, SCALAR *Vol\_of*, const SCALAR *FROUDE*, const int *NBITER*, SCALAR *vol\_output* )**

Saves global values.

**Parameters**

<i>in</i>	<i>time</i>	elapsed time.
<i>in</i>	<i>cpu</i>	CPU time.
<i>in</i>	<i>Vol_rain</i>	total rain volume.
<i>in</i>	<i>Vol_inf</i>	total volume of infiltrated water.
<i>in</i>	<i>Vol_of</i>	total volume of overland flow.
<i>in</i>	<i>FROUDE</i>	mean Froude number (in space) at the final time.
<i>in</i>	<i>NBITER</i>	number of time steps.
<i>in</i>	<i>vol_output</i>	total outflow volume at the boundary.

**Warning**

Impossible to open the \*\*\* file. Verify if the directory \*\*\* exists.

**Note**

If results.dat cannot be opened, the code will exit with failure termination code.

Definition at line 210 of file output.cpp.

**virtual void Output::write ( TAB , TAB , TAB , TAB , SCALAR ) [pure virtual]**

Function to be specified in each output format.

Implemented in [Gnuplot](#), [Vtk\\_Out](#), and [No\\_Evolution\\_File](#).

#### 5.48.4 Member Data Documentation

**const SCALAR Output::DX [protected]**

Space step in the first (x) direction.

Definition at line 110 of file output.hpp.

**const SCALAR Output::DY [protected]**

Space step in the second (y) direction.

Definition at line 112 of file output.hpp.

**string Output::namefile\_Bound\_flux [protected]**

Name of the file where the cumulated boundary fluxes are saved.

Definition at line 124 of file output.hpp.

**string Output::namefile\_Bound\_flux\_BT [protected]**

Name of the file where the bottom and top boundary fluxes are saved.

Definition at line 126 of file output.hpp.

**string Output::namefile\_Bound\_flux\_LR [protected]**

Name of the file where the left and right boundary fluxes are saved.

Definition at line 128 of file output.hpp.

**string Output::namefile\_check\_volume [protected]**

Name of the file where the verification of volumes is saved.

Definition at line 116 of file output.hpp.

**string Output::namefile\_final [protected]**

Name of the file where the final time is saved.

Definition at line 122 of file output.hpp.

**string Output::namefile\_init [protected]**

Name of the file where the initialization is saved.

Definition at line 120 of file output.hpp.

**string Output::namefile\_res [protected]**

Name of the file where the global results are saved.

Definition at line 118 of file output.hpp.

**const int Output::NXCELL [protected]**

Number of cells in space in the first (x) direction.

Definition at line 106 of file output.hpp.

**const int Output::NYCELL [protected]**

Number of cells in space in the second (y) direction.

Definition at line 108 of file output.hpp.

**string Output::outputDirectory [protected]**

Name of the output directory.

Definition at line 114 of file output.hpp.

The documentation for this class was generated from the following files:

- Headers/libsave/output.hpp
- Sources/libsave/output.cpp

**5.49 Parameters Class Reference**

Gets parameters.

```
#include <parameters.hpp>
```

**Public Member Functions**

- [Parameters](#) ()  
*Constructor.*
- void [setparameters](#) (const char \*)  
*Sets the parameters.*
- virtual [~Parameters](#) ()  
*Destructor.*
- int [get\\_Nxcell](#) () const  
*Gives the number of cells in space along x.*
- int [get\\_Nycell](#) () const  
*Gives the number of cells in space along y.*
- [SCALAR get\\_T](#) () const  
*Gives the final time.*
- int [get\\_nbtimes](#) () const  
*Gives the number of times saved.*
- int [get\\_scheme\\_type](#) () const  
*Gives the choice of type of scheme (fixed cfl or fixed dt)*
- [SCALAR get\\_dtfix](#) () const  
*Gives the fixed time step from the parameters.txt file.*
- [SCALAR get\\_cflfix](#) () const  
*Gives the cfl of the scheme.*
- [SCALAR get\\_dx](#) () const  
*Gives the space step along x.*
- [SCALAR get\\_dy](#) () const  
*Gives the space step along y.*
- int [get\\_Lbound](#) () const  
*Gives the value corresponding to the left boundary condition.*
- [SCALAR get\\_left\\_imp\\_discharge](#) () const  
*Gives the value of the imposed discharge in left bc.*
- [SCALAR get\\_left\\_imp\\_h](#) () const



- Gives the value of the imposed water height in left bc.*

  - int `get_Rbound ()` const
    - Gives the value corresponding to the right boundary condition.*
  - SCALAR `get_right_imp_discharge ()` const
    - Gives the value of the imposed discharge in right bc.*
  - SCALAR `get_right_imp_h ()` const
    - Gives the value of the imposed water height in right bc.*
  - int `get_Bbound ()` const
    - Gives the value corresponding to the bottom boundary condition.*
  - SCALAR `get_bottom_imp_discharge ()` const
    - Gives the value of the imposed discharge in bottom bc.*
  - SCALAR `get_bottom_imp_h ()` const
    - Gives the value of the imposed water height in bottom bc.*
  - int `get_Tbound ()` const
    - Gives the value corresponding to the top boundary condition.*
  - SCALAR `get_top_imp_discharge ()` const
    - Gives the value of the imposed discharge in top bc.*
  - SCALAR `get_top_imp_h ()` const
    - Gives the value of the imposed water height in top bc.*
  - int `get_flux ()` const
    - Gives the value corresponding to the flux.*
  - int `get_order ()` const
    - Gives the order of the scheme.*
  - int `get_rec ()` const
    - Gives the value corresponding to the reconstruction.*
  - int `get_fric ()` const
    - Gives the value corresponding to the friction law.*
  - int `get_lim ()` const
    - Gives the value corresponding to the limiter.*
  - int `get_inf ()` const
    - Gives the choice of infiltration model.*
  - SCALAR `get_amortENO ()` const
    - Gives the value of the amortENO parameter.*
  - SCALAR `get_modifENO ()` const
    - Gives the value of the modifENO parameter.*
  - SCALAR `get_friccoef ()` const
    - Gives the value of the friction coefficient.*
  - int `get_fric_init ()` const
    - Gives the value characterizing the spatialization (or not) of the friction coefficient.*
  - string `get_KcNameFile (void)` const
    - Gives the full name of the file containing the hydraulic conductivity of the crust.*
  - string `get_KcNameFileS ()` const
    - Gives the name of the file containing the hydraulic conductivity of the crust.*
  - string `get_KsNameFile (void)` const
    - Gives the full name of the file containing the hydraulic conductivity of the surface.*
  - string `get_KsNameFileS ()` const
    - Gives the name of the file containing the hydraulic conductivity of the surface.*

- string `get_dthetaNameFile` (void) const  
*Gives the full name of the file containing the water content.*
- string `get_dthetaNameFileS` () const  
*Gives the name of the file containing the water content.*
- string `get_PsiNameFile` (void) const  
*Gives the full name of the file containing the load pressure.*
- string `get_PsiNameFileS` () const  
*Gives the name of the file containing the load pressure.*
- string `get_zcrustNameFile` (void) const  
*Gives the full name of the file containing the thickness of the crust.*
- string `get_zcrustNameFileS` () const  
*Gives the name of the file containing the thickness of the crust.*
- string `get_imaxNameFile` (void) const  
*Gives the full name of the file containing the maximum infiltration rate.*
- string `get_imaxNameFileS` () const  
*Gives the name of the file containing the maximum infiltration rate.*
- int `get_Kc_init` () const  
*Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the crust.*
- SCALAR `get_Kc_coef` () const  
*Gives the value of the hydraulic conductivity of the crust.*
- int `get_Ks_init` () const  
*Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the soil.*
- SCALAR `get_Ks_coef` () const  
*Gives the value of the hydraulic conductivity of the soil.*
- int `get_dtheta_init` () const  
*Gives the value characterizing the spatialization (or not) of the water content.*
- SCALAR `get_dtheta_coef` () const  
*Gives the value of the water content.*
- int `get_Psi_init` () const  
*Gives the value characterizing the spatialization (or not) of the load pressure.*
- SCALAR `get_Psi_coef` () const  
*Gives the value of the load pressure.*
- int `get_zcrust_init` () const  
*Gives the value characterizing the spatialization (or not) of the thickness of the crust.*
- SCALAR `get_zcrust_coef` () const  
*Gives the value of the thickness of the crust.*
- int `get_imax_init` () const  
*Gives the value characterizing the spatialization (or not) of the maximum infiltration rate.*
- SCALAR `get_imax_coef` () const  
*Gives the value of the maximum infiltration rate.*
- int `get_topo` () const  
*Gives the value corresponding to the choice of topography.*
- int `get_huv` () const  
*Gives the value corresponding to the choice of initialization of h, u and v.*
- int `get_rain` () const  
*Gives the value corresponding to the choice of initialization of rain.*
- string `get_topographyNameFile` (void) const

- Gives the full name of the file containing the topography.*

  - string `get_topographyNameFileS` () const
    - Gives the name of the file containing the topography.*
  - string `get_huvNameFile` (void) const
    - Gives the full name of the file containing the water height (h) and the velocities (u and v)*
  - string `get_huvNameFileS` (void) const
    - Gives the name of the file containing the water height (h) and the velocities (u and v)*
  - string `get_rainNameFile` (void) const
    - Gives the full name of the file containing the rain.*
  - string `get_rainNameFileS` (void) const
    - Gives the name of the file containing the rain.*
  - string `get_frictionNameFile` (void) const
    - Gives the full name of the file containing the friction coefficient.*
  - string `get_frictionNameFileS` (void) const
    - Gives the name of the file containing the friction coefficient.*
  - string `get_outputDirectory` (void) const
    - Gives the output directory with the suffix.*
  - string `get_suffix` (void) const
    - Gives the suffix for the 'Outputs' directory.*
  - int `get_output` () const
    - Gives the value corresponding to the choice of the format of the [Output](#) file.*
  - void `fill_array` (TAB &, const SCALAR) const
    - Fills the TAB array with a SCALAR.*
  - void `fill_array` (TAB &, string) const
    - Fills the TAB array with the values contained in a file.*

### Protected Attributes

- SCALAR `cfl_fix`
- SCALAR `dt_fix`
- int `scheme_type`
- int `Nxcell`
- int `Nycell`
- int `nbtimes`
- SCALAR `T`
- SCALAR `dx`
- SCALAR `dy`
- SCALAR `L`
- SCALAR `I`
- int `Lbound`
- SCALAR `left_imp_discharge`
- SCALAR `left_imp_h`
- int `Rbound`
- SCALAR `right_imp_discharge`
- SCALAR `right_imp_h`
- int `Bbound`
- SCALAR `bottom_imp_discharge`
- SCALAR `bottom_imp_h`

- int Tbound
- SCALAR top\_imp\_discharge
- SCALAR top\_imp\_h
- int flux
- int order
- int rec
- int fric
- int lim
- int inf
- int topo
- int huv\_init
- int rain
- int Kc\_init
- int Ks\_init
- int dtheta\_init
- int Psi\_init
- int zcrust\_init
- int imax\_init
- int output\_format
- SCALAR amortENO
- SCALAR modifENO
- int fric\_init
- SCALAR friccoef
- SCALAR Kc\_coef
- SCALAR Ks\_coef
- SCALAR dtheta\_coef
- SCALAR Psi\_coef
- SCALAR zcrust\_coef
- SCALAR imax\_coef
- string topography\_namefile
- string topo\_NF
- string huv\_namefile
- string huv\_NF
- string fric\_namefile
- string fric\_NF
- string rain\_namefile
- string rain\_NF
- string Kc\_namefile
- string Kc\_NF
- string Ks\_namefile
- string Ks\_NF
- string dtheta\_namefile
- string dtheta\_NF
- string Psi\_namefile
- string Psi\_NF
- string zcrust\_namefile
- string zcrust\_NF
- string imax\_namefile
- string imax\_NF
- string output\_directory
- string suffix\_outputs

### 5.49.1 Detailed Description

Gets parameters.

Class that reads the parameters, checks their values and contains all the common declarations to get the values of the parameters.

Definition at line 73 of file parameters.hpp.

### 5.49.2 Constructor & Destructor Documentation

#### Parameters::Parameters ( )

Constructor.

Definition at line 2201 of file parameters.cpp.

#### Parameters::~~Parameters ( ) [virtual]

Destructor.

Definition at line 2204 of file parameters.cpp.

### 5.49.3 Member Function Documentation

#### void Parameters::fill\_array ( TAB & *myarray*, const SCALAR *myvalue* ) const

Fills the TAB array with a SCALAR.

Fills an array with a constant value.

Parameters

<i>in, out</i>	<i>myarray</i>	array to fill.
<i>in</i>	<i>myvalue</i>	value.

Definition at line 2072 of file parameters.cpp.

#### void Parameters::fill\_array ( TAB & *myarray*, string *namefile* ) const

Fills the TAB array with the values contained in a file.

Fills an array with the values given in the file

Parameters

<i>in, out</i>	<i>myarray</i>	array to fill.
<i>in</i>	<i>namefile</i>	name of the file containing the values to be inserted into the array.

Warning

\*\*\*: ERROR: cannot open the file.

\*\*\*: ERROR: the number of data in this file is too big/small.

\*\*\*: ERROR: line \*\*\*.

\*\*\*: ERROR: the value for the point x = \*\*\* y = \*\*\* is missing.

\*\*\*: WARNING: line \*\*\* ; a commentary should begin with the # symbol.

Note

If the array cannot be filled correctly, the code will exit with failure termination code.

Definition at line 2086 of file parameters.cpp.

**SCALAR Parameters::get\_amortENO ( ) const**

Gives the value of the amortENO parameter.

Returns

The value of the amortENO parameter [Parameters::amortENO](#).

Definition at line 1640 of file parameters.cpp.

**int Parameters::get\_Bbound ( ) const**

Gives the value corresponding to the bottom boundary condition.

Returns

The value corresponding to the bottom boundary condition [Parameters::Bbound](#).

Definition at line 1529 of file parameters.cpp.

**SCALAR Parameters::get\_bottom\_imp\_discharge ( ) const**

Gives the value of the imposed discharge in bottom bc.

Returns

The value of the imposed discharge per cell in the bottom boundary condition, that is [Parameters::bottom↔\\_imp\\_discharge](#) / [Parameters::L](#).

Definition at line 1539 of file parameters.cpp.

**SCALAR Parameters::get\_bottom\_imp\_h ( ) const**

Gives the value of the imposed water height in bottom bc.

Returns

The value of the imposed water height in the bottom boundary condition [Parameters::bottom\\_imp\\_h](#).

Definition at line 1549 of file parameters.cpp.

**SCALAR Parameters::get\_cflfix ( ) const**

Gives the cfl of the scheme.

Returns

The fixed cfl [Parameters::cfl\\_fix](#).

Definition at line 1439 of file parameters.cpp.

**SCALAR Parameters::get\_dtfix ( ) const**

Gives the fixed time step from the parameters.txt file.

Returns

The fixed space step [Parameters::dx\\_fix](#).

Definition at line 1429 of file parameters.cpp.

**SCALAR Parameters::get\_dtheta\_coef ( ) const**

Gives the value of the water content.

Returns

The value of dtheta [Parameters::dtheta\\_coef](#).

Definition at line 1900 of file parameters.cpp.

**int Parameters::get\_dtheta\_init ( ) const**

Gives the value characterizing the spatialization (or not) of the water content.

Returns

The value corresponding to the initialization of dtheta [Parameters::dtheta\\_init](#).

Definition at line 1890 of file parameters.cpp.

**string Parameters::get\_dthetaNameFile ( void ) const**

Gives the full name of the file containing the water content.

Returns

The dtheta path for the initialization + Input directory [Parameters::dtheta\\_namefile](#).

Definition at line 1910 of file parameters.cpp.

**string Parameters::get\_dthetaNameFileS ( ) const**

Gives the name of the file containing the water content.

Returns

The dtheta namefile for the initialization (inside the Input directory) [Parameters::dtheta\\_NF](#).

Definition at line 1920 of file parameters.cpp.

**SCALAR Parameters::get\_dx ( ) const**

Gives the space step along x.

Returns

The space step in the first (x) direction [Parameters::dx](#).

Definition at line 1449 of file parameters.cpp.

**SCALAR Parameters::get\_dy ( ) const**

Gives the space step along y.

Returns

The space step in the second (y) direction [Parameters::dy](#).

Definition at line 1459 of file parameters.cpp.

**int Parameters::get\_flux ( ) const**

Gives the value corresponding to the flux.

Returns

The value corresponding to the flux [Parameters::flux](#).

Definition at line 1589 of file parameters.cpp.

**int Parameters::get\_fric ( ) const**

Gives the value corresponding to the friction law.

Returns

The value corresponding to the friction law [Parameters::fric](#).

Definition at line 1619 of file parameters.cpp.

**int Parameters::get\_fric\_init ( ) const**

Gives the value characterizing the spatialization (or not) of the friction coefficient.

Returns

The value corresponding to the friction coefficient [Parameters::fric\\_init](#).

Definition at line 1670 of file parameters.cpp.

**SCALAR Parameters::get\_friccoef ( ) const**

Gives the value of the friction coefficient.

Returns

The value of the friction coefficient [Parameters::friccoef](#).

Definition at line 1700 of file parameters.cpp.

**string Parameters::get\_frictionNameFile ( void ) const**

Gives the full name of the file containing the friction coefficient.

Returns

The friction coefficient path + Input directory [Parameters::fric\\_namefile](#).

Definition at line 1680 of file parameters.cpp.

**string Parameters::get\_frictionNameFileS ( void ) const**

Gives the name of the file containing the friction coefficient.

Returns

The friction coefficient namefile (inside the Input directory) [Parameters::fric\\_NF](#).

Definition at line 1690 of file parameters.cpp.



**int Parameters::get\_huv ( ) const**

Gives the value corresponding to the choice of initialization of h, u and v.

Returns

The value corresponding to the initialization of h and u,v [Parameters::huv\\_init](#).

Definition at line 1740 of file parameters.cpp.

**string Parameters::get\_huvNameFile ( void ) const**

Gives the full name of the file containing the water height (h) and the velocities (u and v)

Returns

The h and u,v path for the initialization + Input directory [Parameters::huv\\_namefile](#).

Definition at line 1750 of file parameters.cpp.

**string Parameters::get\_huvNameFileS ( void ) const**

Gives the name of the file containing the water height (h) and the velocities (u and v)

Returns

The h and u namefile for the initialization (inside the Input directory) [Parameters::huv\\_NF](#).

Definition at line 1760 of file parameters.cpp.

**SCALAR Parameters::get\_imax\_coef ( ) const**

Gives the value of the maximum infiltration rate.

Returns

The value of imax [Parameters::imax\\_coef](#).

Definition at line 2020 of file parameters.cpp.

**int Parameters::get\_imax\_init ( ) const**

Gives the value characterizing the spatialization (or not) of the maximum infiltration rate.

Returns

The value corresponding to the initialization of imax [Parameters::imax\\_init](#).

Definition at line 2010 of file parameters.cpp.

**string Parameters::get\_imaxNameFile ( void ) const**

Gives the full name of the file containing the maximum infiltration rate.

Returns

The imax path for the initialization + Input directory [Parameters::imax\\_namefile](#).

Definition at line 2030 of file parameters.cpp.

**string Parameters::get\_imaxNameFileS ( ) const**

Gives the name of the file containing the maximum infiltration rate.

Returns

The imax namefile for the initialization (inside the Input directory) [Parameters::imax\\_NF](#).

Definition at line 2040 of file parameters.cpp.

**int Parameters::get\_inf ( ) const**

Gives the choice of infiltration model.

Returns

The value corresponding to the infiltration [Parameters::inf](#).

Definition at line 1660 of file parameters.cpp.

**SCALAR Parameters::get\_Kc\_coef ( ) const**

Gives the value of the hydraulic conductivity of the crust.

Returns

The value of Kc [Parameters::Kc\\_coef](#).

Definition at line 1820 of file parameters.cpp.

**int Parameters::get\_Kc\_init ( ) const**

Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the crust.

Returns

The value corresponding to the initialization of Kc [Parameters::Kc\\_init](#).

Definition at line 1810 of file parameters.cpp.

**string Parameters::get\_KcNameFile ( void ) const**

Gives the full name of the file containing the hydraulic conductivity of the crust.

Returns

The Kc path for the initialization + Input directory [Parameters::Kc\\_namefile](#).

Definition at line 1830 of file parameters.cpp.

**string Parameters::get\_KcNameFileS ( ) const**

Gives the name of the file containing the hydraulic conductivity of the crust.

Returns

The Kc namefile for the initialization (inside the Input directory) [Parameters::Kc\\_NF](#).

Definition at line 1840 of file parameters.cpp.

**SCALAR Parameters::get\_Ks\_coef ( ) const**

Gives the value of the hydraulic conductivity of the soil.

Returns

The value of Ks [Parameters::Ks\\_coef](#).

Definition at line 1860 of file parameters.cpp.

**int Parameters::get\_Ks\_init ( ) const**

Gives the value characterizing the spatialization (or not) of the hydraulic conductivity of the soil.

Returns

The value corresponding to the initialization of Ks [Parameters::Ks\\_init](#).

Definition at line 1850 of file parameters.cpp.

**string Parameters::get\_KsNameFile ( void ) const**

Gives the full name of the file containing the hydraulic conductivity of the surface.

Returns

The Ks path for the initialization + Input directory [Parameters::Ks\\_namefile](#).

Definition at line 1870 of file parameters.cpp.

**string Parameters::get\_KsNameFileS ( ) const**

Gives the name of the file containing the hydraulic conductivity of the surface.

Returns

The Ks namefile for the initialization (inside the Input directory) [Parameters::Ks\\_NF](#).

Definition at line 1880 of file parameters.cpp.

**int Parameters::get\_Lbound ( ) const**

Gives the value corresponding to the left boundary condition.

Returns

The value corresponding to the left boundary condition [Parameters::Lbound](#).

Definition at line 1469 of file parameters.cpp.

**SCALAR Parameters::get\_left\_imp\_discharge ( ) const**

Gives the value of the imposed discharge in left bc.

Returns

The value of the imposed discharge per cell in the left boundary condition, that is [Parameters::left\\_imp\\_discharge](#) / [Parameters::l](#).

Definition at line 1479 of file parameters.cpp.

**SCALAR Parameters::get\_left\_imp\_h ( ) const**

Gives the value of the imposed water height in left bc.

Returns

The value of the imposed water height in the left boundary condition [Parameters::left\\_imp\\_h](#).

Definition at line 1489 of file parameters.cpp.

**int Parameters::get\_lim ( ) const**

Gives the value corresponding to the limiter.

Returns

The value corresponding to the limiter [Parameters::lim](#).

Definition at line 1630 of file parameters.cpp.

**SCALAR Parameters::get\_modifENO ( ) const**

Gives the value of the modifENO parameter.

Returns

The value of the modifENO parameter [Parameters::modifENO](#).

Definition at line 1650 of file parameters.cpp.

**int Parameters::get\_nbtimes ( ) const**

Gives the number of times saved.

Returns

The number of times saved [Parameters::nbtimes](#).

Definition at line 1409 of file parameters.cpp.

**int Parameters::get\_Nxcell ( ) const**

Gives the number of cells in space along x.

Returns

The number of cells in space in the first (x) direction [Parameters::Nxcell](#).

Definition at line 1379 of file parameters.cpp.

**int Parameters::get\_Nycell ( ) const**

Gives the number of cells in space along y.

Returns

The number of cells in space in the second (y) direction [Parameters::Nycell](#).

Definition at line 1389 of file parameters.cpp.

**int Parameters::get\_order ( ) const**

Gives the order of the scheme.

Returns

The order of the scheme [Parameters::order](#).

Definition at line 1599 of file parameters.cpp.

**int Parameters::get\_output ( ) const**

Gives the value corresponding to the choice of the format of the [Output](#) file.

Returns

The type of output [Parameters::output\\_format](#).

Definition at line 2060 of file parameters.cpp.

**string Parameters::get\_outputDirectory ( void ) const**

Gives the output directory with the suffix.

Returns

The output directory with the suffix [Parameters::output\\_directory](#).

Definition at line 1800 of file parameters.cpp.

**SCALAR Parameters::get\_Psi\_coef ( ) const**

Gives the value of the load pressure.

Returns

The value of Psi [Parameters::Psi\\_coef](#).

Definition at line 1940 of file parameters.cpp.

**int Parameters::get\_Psi\_init ( ) const**

Gives the value characterizing the spatialization (or not) of the load pressure.

Returns

The value corresponding to the initialization of Psi [Parameters::Psi\\_init](#).

Definition at line 1930 of file parameters.cpp.

**string Parameters::get\_PsiNameFile ( void ) const**

Gives the full name of the file containing the load pressure.

Returns

The Psi path for the initialization + Input directory [Parameters::Psi\\_namefile](#).

Definition at line 1950 of file parameters.cpp.

**string Parameters::get\_PsiNameFileS ( ) const**

Gives the name of the file containing the load pressure.

Returns

The Psi namefile for the initialization (inside the Input directory) [Parameters::Psi\\_NF](#).

Definition at line 1960 of file parameters.cpp.

**int Parameters::get\_rain ( ) const**

Gives the value corresponding to the choice of initialization of rain.

Returns

The value corresponding to the initialization of the rain [Parameters::rain](#).

Definition at line 1770 of file parameters.cpp.

**string Parameters::get\_rainNameFile ( void ) const**

Gives the full name of the file containing the rain.

Returns

The rain path for the initialization + Input directory [Parameters::rain\\_namefile](#).

Definition at line 1780 of file parameters.cpp.

**string Parameters::get\_rainNameFileS ( void ) const**

Gives the name of the file containing the rain.

Returns

The rain namefile for the initialization (inside the Input directory) [Parameters::rain\\_NF](#).

Definition at line 1790 of file parameters.cpp.

**int Parameters::get\_Rbound ( ) const**

Gives the value corresponding to the right boundary condition.

Returns

The value corresponding to the right boundary condition [Parameters::Rbound](#).

Definition at line 1499 of file parameters.cpp.

**int Parameters::get\_rec ( ) const**

Gives the value corresponding to the reconstruction.

Returns

The value corresponding to the reconstruction [Parameters::rec](#).

Definition at line 1609 of file parameters.cpp.

**SCALAR Parameters::get\_right\_imp\_discharge ( ) const**

Gives the value of the imposed discharge in right bc.

Returns

The value of the imposed discharge per cell in the right boundary condition, that is [Parameters::right\\_imp\\_discharge](#) / [Parameters::l](#).

Definition at line 1509 of file parameters.cpp.

**SCALAR Parameters::get\_right\_imp\_h ( ) const**

Gives the value of the imposed water height in right bc.

Returns

The value of the imposed water height in the right boundary condition [Parameters::right\\_imp\\_h](#).

Definition at line 1519 of file parameters.cpp.

**int Parameters::get\_scheme\_type ( ) const**

Gives the choice of type of scheme (fixed cfl or fixed dt)

Returns

The type of scheme [Parameters::scheme\\_type](#).

Definition at line 1419 of file parameters.cpp.

**string Parameters::get\_suffix ( void ) const**

Gives the suffix for the 'Outputs' directory.

Returns

The suffix (for the output directory) [Parameters::suffix\\_outputs](#).

Definition at line 2050 of file parameters.cpp.

**SCALAR Parameters::get\_T ( ) const**

Gives the final time.

Returns

The final time [Parameters::T](#).

Definition at line 1399 of file parameters.cpp.

**int Parameters::get\_Tbound ( ) const**

Gives the value corresponding to the top boundary condition.

Returns

The value corresponding to the top boundary condition [Parameters::Tbound](#).

Definition at line 1559 of file parameters.cpp.

**SCALAR Parameters::get\_top\_imp\_discharge ( ) const**

Gives the value of the imposed discharge in top bc.

Returns

The value of the imposed discharge per cell in the top boundary condition, that is [Parameters::top\\_imp\\_discharge](#) / [Parameters::L](#).

Definition at line 1569 of file parameters.cpp.

**SCALAR Parameters::get\_top\_imp\_h ( ) const**

Gives the value of the imposed water height in top bc.

Returns

The value of the imposed water height in the bottom boundary condition [Parameters::top\\_imp\\_h](#).

Definition at line 1579 of file parameters.cpp.

**int Parameters::get\_topo ( ) const**

Gives the value corresponding to the choice of topography.

Returns

The value corresponding to the topography [Parameters::topo](#).

Definition at line 1730 of file parameters.cpp.

**string Parameters::get\_topographyNameFile ( void ) const**

Gives the full name of the file containing the topography.

Returns

The topography path + Input directory [Parameters::topography\\_namefile](#).

Definition at line 1710 of file parameters.cpp.

**string Parameters::get\_topographyNameFileS ( void ) const**

Gives the name of the file containing the topography.

Returns

The topography namefile (inside the Input directory) [Parameters::topo\\_NF](#).

Definition at line 1720 of file parameters.cpp.

**SCALAR Parameters::get\_zcrust\_coef ( ) const**

Gives the value of the thickness of the crust.

Returns

The value of zcrust [Parameters::zcrust\\_coef](#).

Definition at line 1980 of file parameters.cpp.



**int Parameters::get\_zcrust\_init ( ) const**

Gives the value characterizing the spatialization (or not) of the thickness of the crust.

Returns

The value corresponding to the initialization of zcrust [Parameters::zcrust\\_init](#).

Definition at line 1970 of file parameters.cpp.

**string Parameters::get\_zcrustNameFile ( void ) const**

Gives the full name of the file containing the thickness of the crust.

Returns

The zcrust path for the initialization + Input directory [Parameters::zcrust\\_namefile](#).

Definition at line 1990 of file parameters.cpp.

**string Parameters::get\_zcrustNameFileS ( ) const**

Gives the name of the file containing the thickness of the crust.

Returns

The zcrust namefile for the initialization (inside the Input directory) [Parameters::zcrust\\_NF](#).

Definition at line 2000 of file parameters.cpp.

**void Parameters::setparameters ( const char \* *FILENAME* )**

Sets the parameters.

Gets all the parameters from the file *FILENAME*, check and affect them. The values used by FullSWOF\_2D are saved in the file parameters.dat. These values are also printed in the terminal when the code is run.

Parameters

in	<i>FILENAME</i>	name of the paramters file.
----	-----------------	-----------------------------

Warning

parameters.txt: ERROR: \*\*\*.

parameters.txt: WARNING: \*\*\*.

ERROR: the \*\*\* file \*\*\* does not exists in the directory Inputs.

Impossible to open the \*\*\* file. Verify if the directory \*\*\* exists.

Note

If a value cannot be affected correctly, the code will exit with failure termination code.

If parameters.dat cannot be opened, the code will exit with failure termination code.

Definition at line 61 of file parameters.cpp.

**5.49.4 Member Data Documentation****SCALAR Parameters::amortENO [protected]**

Parameter for eno.

Definition at line 380 of file parameters.hpp.

**int Parameters::Bbound** [protected]

Bottom boundary condition.

Definition at line 336 of file parameters.hpp.

**SCALAR Parameters::bottom\_imp\_discharge** [protected]

Imposed discharge on the bottom boundary.

Definition at line 338 of file parameters.hpp.

**SCALAR Parameters::bottom\_imp\_h** [protected]

Imposed water height on the bottom boundary.

Definition at line 340 of file parameters.hpp.

**SCALAR Parameters::cfl\_fix** [protected]

Value of the fixed cfl.

Definition at line 302 of file parameters.hpp.

**SCALAR Parameters::dt\_fix** [protected]

Value of the fixed time step.

Definition at line 304 of file parameters.hpp.

**SCALAR Parameters::dtheta\_coef** [protected]

Value of dtheta.

Definition at line 392 of file parameters.hpp.

**int Parameters::dtheta\_init** [protected]

Type of initialization of dtheta.

Definition at line 370 of file parameters.hpp.

**string Parameters::dtheta\_namefile** [protected]

Name of the file for dtheta: Inputs/file.

Definition at line 424 of file parameters.hpp.

**string Parameters::dtheta\_NF** [protected]

Name of the file for dtheta without 'Inputs'.

Definition at line 426 of file parameters.hpp.

**SCALAR Parameters::dx** [protected]

Space step in the first (x) direction.

Definition at line 316 of file parameters.hpp.

**SCALAR Parameters::dy** [protected]

Space step in the second (y) direction.

Definition at line 318 of file parameters.hpp.

**int Parameters::flux [protected]**

Numerical flux.

Definition at line 348 of file parameters.hpp.

**int Parameters::fric [protected]**

Friction.

Definition at line 354 of file parameters.hpp.

**int Parameters::fric\_init [protected]**

Type of friction coefficient.

Definition at line 384 of file parameters.hpp.

**string Parameters::fric\_namefile [protected]**

Name of the file for the friction coefficient: Inputs/file.

Definition at line 408 of file parameters.hpp.

**string Parameters::fric\_NF [protected]**

Name of the file for the friction coefficient without 'Inputs'.

Definition at line 410 of file parameters.hpp.

**SCALAR Parameters::friccoef [protected]**

Friction coefficient.

Definition at line 386 of file parameters.hpp.

**int Parameters::huv\_init [protected]**

Type of initial conditions for h and u,v.

Definition at line 362 of file parameters.hpp.

**string Parameters::huv\_namefile [protected]**

Name of the file for the initialization of h and u,v: Inputs/file.

Definition at line 404 of file parameters.hpp.

**string Parameters::huv\_NF [protected]**

Name of the file for the initialization of h and u,v without 'Inputs'.

Definition at line 406 of file parameters.hpp.

**SCALAR Parameters::imax\_coef [protected]**

Value of imax.

Definition at line 398 of file parameters.hpp.

**int Parameters::imax\_init [protected]**

Type of initialization of imax.

Definition at line 376 of file parameters.hpp.

**string Parameters::imax\_namefile [protected]**

Name of the file for imax: Inputs/file.

Definition at line 436 of file parameters.hpp.

**string Parameters::imax\_NF [protected]**

Name of the file for imax without 'Inputs'.

Definition at line 438 of file parameters.hpp.

**int Parameters::inf [protected]**

Type of infiltration.

Definition at line 358 of file parameters.hpp.

**SCALAR Parameters::Kc\_coef [protected]**

Value of Kc.

Definition at line 388 of file parameters.hpp.

**int Parameters::Kc\_init [protected]**

Type of initialization of Kc.

Definition at line 366 of file parameters.hpp.

**string Parameters::Kc\_namefile [protected]**

Name of the file for Kc: Inputs/file.

Definition at line 416 of file parameters.hpp.

**string Parameters::Kc\_NF [protected]**

Name of the file for Kc without 'Inputs'.

Definition at line 418 of file parameters.hpp.

**SCALAR Parameters::Ks\_coef [protected]**

Value of Ks.

Definition at line 390 of file parameters.hpp.

**int Parameters::Ks\_init [protected]**

Type of initialization of Ks.

Definition at line 368 of file parameters.hpp.

**string Parameters::Ks\_namefile [protected]**

Name of the file for Ks: Inputs/file.

Definition at line 420 of file parameters.hpp.

**string Parameters::Ks\_NF [protected]**

Name of the file for Ks without 'Inputs'.

Definition at line 422 of file parameters.hpp.

**SCALAR Parameters::L [protected]**

Length of the domain in the first (x) direction.  
Definition at line 320 of file parameters.hpp.

**SCALAR Parameters::l [protected]**

Length of the domain in the second (y) direction.  
Definition at line 322 of file parameters.hpp.

**int Parameters::Lbound [protected]**

Left boundary condition.  
Definition at line 324 of file parameters.hpp.

**SCALAR Parameters::left\_imp\_discharge [protected]**

Imposed discharge on the left boundary.  
Definition at line 326 of file parameters.hpp.

**SCALAR Parameters::left\_imp\_h [protected]**

Imposed water height on the left boundary.  
Definition at line 328 of file parameters.hpp.

**int Parameters::lim [protected]**

Slope limiter.  
Definition at line 356 of file parameters.hpp.

**SCALAR Parameters::modifENO [protected]**

Parameter for eno\_modif.  
Definition at line 382 of file parameters.hpp.

**int Parameters::nbtimes [protected]**

Number of times saved.  
Definition at line 312 of file parameters.hpp.

**int Parameters::Nxcell [protected]**

Number of cells in space in the first (x) direction.  
Definition at line 308 of file parameters.hpp.

**int Parameters::Nycell [protected]**

Number of cells in space in the second (y) direction.  
Definition at line 310 of file parameters.hpp.

**int Parameters::order [protected]**

Order of the numerical scheme.  
Definition at line 350 of file parameters.hpp.

**string Parameters::output\_directory [protected]**

Name of the output directory Outputs+suffix.  
Definition at line 440 of file parameters.hpp.

**int Parameters::output\_format [protected]**

Type of output.  
Definition at line 378 of file parameters.hpp.

**SCALAR Parameters::Psi\_coef [protected]**

Value of Psi.  
Definition at line 394 of file parameters.hpp.

**int Parameters::Psi\_init [protected]**

Type of initialization of Psi.  
Definition at line 372 of file parameters.hpp.

**string Parameters::Psi\_namefile [protected]**

Name of the file for Psi: Inputs/file.  
Definition at line 428 of file parameters.hpp.

**string Parameters::Psi\_NF [protected]**

Name of the file for Psi without 'Inputs'.  
Definition at line 430 of file parameters.hpp.

**int Parameters::rain [protected]**

Type of rain.  
Definition at line 364 of file parameters.hpp.

**string Parameters::rain\_namefile [protected]**

Name of the file for the rain: Inputs/file.  
Definition at line 412 of file parameters.hpp.

**string Parameters::rain\_NF [protected]**

Name of the file for the rain without 'Inputs'.  
Definition at line 414 of file parameters.hpp.

**int Parameters::Rbound [protected]**

Right boundary condition.  
Definition at line 330 of file parameters.hpp.

**int Parameters::rec [protected]**

Reconstruction.  
Definition at line 352 of file parameters.hpp.

**SCALAR Parameters::right\_imp\_discharge [protected]**

Imposed discharge on the right boundary.

Definition at line 332 of file parameters.hpp.

**SCALAR Parameters::right\_imp\_h [protected]**

Imposed water height on the right boundary.

Definition at line 334 of file parameters.hpp.

**int Parameters::scheme\_type [protected]**

Type of scheme (fixed cfl or time step).

Definition at line 306 of file parameters.hpp.

**string Parameters::suffix\_outputs [protected]**

Suffix for the output directory.

Definition at line 442 of file parameters.hpp.

**SCALAR Parameters::T [protected]**

Final time.

Definition at line 314 of file parameters.hpp.

**int Parameters::Tbound [protected]**

Top boundary condition.

Definition at line 342 of file parameters.hpp.

**SCALAR Parameters::top\_imp\_discharge [protected]**

Imposed discharge on the top boundary.

Definition at line 344 of file parameters.hpp.

**SCALAR Parameters::top\_imp\_h [protected]**

Imposed water height on the top boundary.

Definition at line 346 of file parameters.hpp.

**int Parameters::topo [protected]**

Type of topography.

Definition at line 360 of file parameters.hpp.

**string Parameters::topo\_NF [protected]**

Name of the file for the topography without 'Inputs'.

Definition at line 402 of file parameters.hpp.

**string Parameters::topography\_namefile [protected]**

Name of the file for the topography: Inputs/file.

Definition at line 400 of file parameters.hpp.

**SCALAR Parameters::zcrust\_coef [protected]**

Value of zcrust.

Definition at line 396 of file parameters.hpp.

**int Parameters::zcrust\_init [protected]**

Type of initialization of zcrust.

Definition at line 374 of file parameters.hpp.

**string Parameters::zcrust\_namefile [protected]**

Name of the file for zcrust: Inputs/file.

Definition at line 432 of file parameters.hpp.

**string Parameters::zcrust\_NF [protected]**

Name of the file for zcrust without 'Inputs'.

Definition at line 434 of file parameters.hpp.

The documentation for this class was generated from the following files:

- Headers/libparameters/[parameters.hpp](#)
- Sources/libparameters/[parameters.cpp](#)

## 5.50 Parser Class Reference

Parser to read the entries

```
#include <parser.hpp>
```

### Public Member Functions

- [Parser](#) (const char \*)  
*Constructor.*
- string [getValue](#) (const char \*)  
*Returns the value of the variable.*
- virtual [~Parser](#) ()  
*Destructor.*

#### 5.50.1 Detailed Description

Parser to read the entries

Class that reads the input file written as description <variable>:: value # comment and keep the values after the "::" ignoring the comments that begin with a "#".

Definition at line 80 of file parser.hpp.

#### 5.50.2 Constructor & Destructor Documentation

**Parser::Parser ( const char \* *FILENAME* )**

Constructor.

Constructor: reads the input parameter and copy the data in a tabular.



**Parameters**

in	<i>FILENAME</i>	name of the paramters file.
----	-----------------	-----------------------------

**Warning**

Impossible to open the \*\*\* file.

**Note**

If the parameters file cannot be opened, the code will exit with failure termination code.

Definition at line 57 of file parser.cpp.

**Parser::~Parser ( ) [virtual]****Destructor.**

Definition at line 161 of file parser.cpp.

**5.50.3 Member Function Documentation****string Parser::getValue ( const char \* TAG )**

Returns the value of the variable.

Return the value corresponding to the tag.

**Parameters**

in	<i>TAG</i>	name of the variable with delimiters.
----	------------	---------------------------------------

**Warning**

No entry for the variable \*\*\*.

Bad syntax for \*\*\*. The syntax is: description <variable>:: value

**Returns**

Value of the variable as a string

**Note**

If the value cannot be read correctly, the code will exit with failure termination code.

Definition at line 113 of file parser.cpp.

The documentation for this class was generated from the following files:

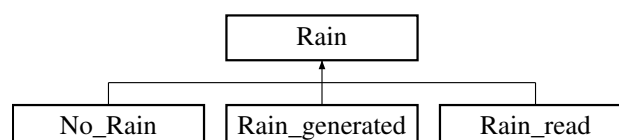
- Headers/libparser/[parser.hpp](#)
- Sources/libparser/[parser.cpp](#)

**5.51 Rain Class Reference**

Initialization of the rain.

```
#include <rain.hpp>
```

Inheritance diagram for Rain:



## Public Member Functions

- [Rain \(Parameters &\)](#)  
*Constructor.*
- virtual void [rain\\_func \(SCALAR, TAB &\)=0](#)  
*Function to be specified in each case.*
- virtual [~Rain \(\)](#)  
*Destructor.*

## Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const [SCALAR DX](#)
- const [SCALAR DY](#)

### 5.51.1 Detailed Description

Initialization of the rain.

Class that contains all the common declarations for the initialization of the rain.

Definition at line 70 of file rain.hpp.

### 5.51.2 Constructor & Destructor Documentation

#### **Rain::Rain ( Parameters & *par* )**

Constructor.

Defines the number of cells and the space steps.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 58 of file rain.cpp.

#### **Rain::~~Rain ( ) [virtual]**

Destructor.

Definition at line 68 of file rain.cpp.

### 5.51.3 Member Function Documentation

#### **virtual void Rain::rain\_func ( SCALAR , TAB & ) [pure virtual]**

Function to be specified in each case.

Implemented in [Rain\\_generated](#), [Rain\\_read](#), and [No\\_Rain](#).

### 5.51.4 Member Data Documentation

#### **const SCALAR Rain::DX [protected]**

Space step in the first (x) direction (unused).

Definition at line 89 of file rain.hpp.

#### **const SCALAR Rain::DY [protected]**

Space step in the second (y) direction (unused).

Definition at line 91 of file rain.hpp.

**const int Rain::NXCELL [protected]**

Number of cells in space in the first (x) direction.

Definition at line 85 of file rain.hpp.

**const int Rain::NYCELL [protected]**

Number of cells in space in the second (y) direction.

Definition at line 87 of file rain.hpp.

The documentation for this class was generated from the following files:

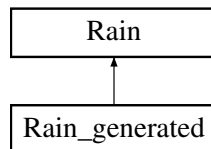
- Headers/librain\_infiltration/rain.hpp
- Sources/librain\_infiltration/rain.cpp

## 5.52 Rain\_generated Class Reference

Constant rain configuration.

```
#include <rain_generated.hpp>
```

Inheritance diagram for Rain\_generated:



### Public Member Functions

- [Rain\\_generated \(Parameters &\)](#)  
*Constructor.*
- void [rain\\_func \(SCALAR, TAB &\)](#)  
*Performs the constant initialization.*
- virtual [~Rain\\_generated \(\)](#)  
*Destructor.*

### Additional Inherited Members

#### 5.52.1 Detailed Description

Constant rain configuration.

Class that initializes a constant rain, with value 0.00001 m/s = 36 mm/h.

Definition at line 73 of file rain\_generated.hpp.

#### 5.52.2 Constructor & Destructor Documentation

##### Rain\_generated::Rain\_generated ( Parameters & par )

Constructor.

Parameters

---

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
-----------------	------------------	---

Definition at line 59 of file `rain_generated.cpp`.

### **Rain\_generated::~~Rain\_generated ( ) [virtual]**

Destructor.

Definition at line 85 of file `rain_generated.cpp`.

### 5.52.3 Member Function Documentation

#### **void Rain\_generated::rain\_func ( SCALAR *time*, TAB & *Tab\_rain* ) [virtual]**

Performs the constant initialization.

Initializes the rain to 0.00001 m/s = 36 mm/h.

Parameters

<code>in</code>	<code>time</code>	the current time (unused).
<code>in, out</code>	<code>Tab_rain</code>	rain intensity at the current time on each cell.

Implements [Rain](#).

Definition at line 67 of file `rain_generated.cpp`.

The documentation for this class was generated from the following files:

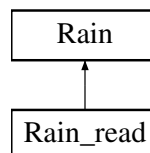
- [Headers/librain\\_infiltration/rain\\_generated.hpp](#)
- [Sources/librain\\_infiltration/rain\\_generated.cpp](#)

## 5.53 Rain\_read Class Reference

File configuration.

```
#include <rain_read.hpp>
```

Inheritance diagram for `Rain_read`:



### Public Member Functions

- [Rain\\_read \(Parameters &\)](#)  
*Constructor.*
- void [rain\\_func \(SCALAR, TAB &\)](#)  
*Performs the initialization.*
- virtual [~Rain\\_read \(\)](#)  
*Destructor.*

### Additional Inherited Members

#### 5.53.1 Detailed Description

File configuration.

Class that initializes the rain to the values read in a file.

Definition at line 71 of file `rain_read.hpp`.

### 5.53.2 Constructor & Destructor Documentation

#### Rain\_read::Rain\_read ( Parameters & *par* )

Constructor.

Defines the name of the file for the initialization and creates two tables (times and intensity) from the data read in the file.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Warning

(rain\_namefile): ERROR: cannot open the rain file.  
 (rain\_namefile): ERROR: line \*\*\*.  
 (rain\_namefile): ERROR: the first time must be t = 0.

Definition at line 59 of file rain\_read.cpp.

#### Rain\_read::~~Rain\_read ( ) [virtual]

Destructor.

Definition at line 151 of file rain\_read.cpp.

### 5.53.3 Member Function Documentation

#### void Rain\_read::rain\_func ( SCALAR *time*, TAB & *Tab\_rain* ) [virtual]

Performs the initialization.

Initializes the rain to the values read in the corresponding file.

Parameters

<i>in</i>	<i>time</i>	current time.
<i>in, out</i>	<i>Tab_rain</i>	rain intensity at the current time on each cell.

Note

As the times read in the file must start with t = 0, *Tab\_rain* is initialized.

Implements [Rain](#).

Definition at line 131 of file rain\_read.cpp.

The documentation for this class was generated from the following files:

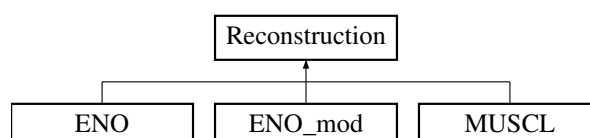
- [Headers/librain\\_infiltration/rain\\_read.hpp](#)
- [Sources/librain\\_infiltration/rain\\_read.cpp](#)

## 5.54 Reconstruction Class Reference

Reconstruction of the variables

```
#include <reconstruction.hpp>
```

Inheritance diagram for Reconstruction:



## Public Member Functions

- [Reconstruction](#) (Parameters &, TAB &)  
*Constructor.*
- virtual void [calcul](#) (TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &)=0  
*Function to be specified in each reconstruction.*
- virtual [~Reconstruction](#) ()  
*Destructor.*

## Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- [TAB z1r](#)
- [TAB z1l](#)
- [TAB z2r](#)
- [TAB z2l](#)
- [TAB delta\\_z1](#)
- [TAB delta\\_z2](#)
- [Choice\\_limiter](#) \* [limiter](#)

### 5.54.1 Detailed Description

Reconstruction of the variables

Class that contains all the common declarations for the second order reconstruction in space.  
Definition at line 74 of file reconstruction.hpp.

### 5.54.2 Constructor & Destructor Documentation

#### **Reconstruction::Reconstruction ( Parameters & *par*, TAB & *z* )**

Constructor.

Defines the number of cells, the slope limiter, and initializes [Reconstruction::z1l](#), [Reconstruction::z2l](#), [Reconstruction::z1r](#), [Reconstruction::z2r](#), [Reconstruction::delta\\_z1](#), [Reconstruction::delta\\_z2](#).

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
<i>in</i>	<i>z</i>	topography.

Definition at line 59 of file reconstruction.cpp.

#### **Reconstruction::~Reconstruction ( ) [virtual]**

Destructor.

Definition at line 109 of file reconstruction.cpp.

### 5.54.3 Member Function Documentation

**virtual void Reconstruction::calcul ( TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB &, TAB & ) [pure virtual]**

Function to be specified in each reconstruction.

Implemented in [ENO](#), [ENO\\_mod](#), and [MUSCL](#).

#### 5.54.4 Member Data Documentation

##### **TAB Reconstruction::delta\_z1** [protected]

Difference between the values of the topography on two adjacent cells (on the right) in the first direction  
Definition at line 101 of file reconstruction.hpp.

##### **TAB Reconstruction::delta\_z2** [protected]

Difference between the values of the topography on two adjacent cells (on the right) in the second direction  
Definition at line 103 of file reconstruction.hpp.

##### **Choice\_limiter\* Reconstruction::limiter** [protected]

Slope limiter  
Definition at line 105 of file reconstruction.hpp.

##### **const int Reconstruction::NXCELL** [protected]

Number of cells in space in the first (x) direction.  
Definition at line 89 of file reconstruction.hpp.

##### **const int Reconstruction::NYCELL** [protected]

Number of cells in space in the second (y) direction.  
Definition at line 91 of file reconstruction.hpp.

##### **TAB Reconstruction::z1l** [protected]

Reconstructed topography on the left boundary in the first direction  
Definition at line 95 of file reconstruction.hpp.

##### **TAB Reconstruction::z1r** [protected]

Reconstructed topography on the right boundary in the first direction  
Definition at line 93 of file reconstruction.hpp.

##### **TAB Reconstruction::z2l** [protected]

Reconstructed topography on the left boundary in the second direction  
Definition at line 99 of file reconstruction.hpp.

##### **TAB Reconstruction::z2r** [protected]

Reconstructed topography on the right boundary in the second direction  
Definition at line 97 of file reconstruction.hpp.

The documentation for this class was generated from the following files:

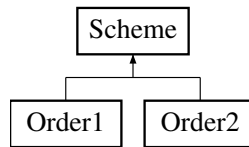
- Headers/libreconstructions/[reconstruction.hpp](#)
- Sources/libreconstructions/[reconstruction.cpp](#)

## 5.55 Scheme Class Reference

Numerical scheme.

```
#include <scheme.hpp>
```

Inheritance diagram for Scheme:



### Public Member Functions

- [Scheme](#) ([Parameters](#) &)  
*Constructor.*
- virtual void [calcul](#) ()=0  
*Function to be specified in each numerical scheme.*
- void [allocation](#) ()  
*Allocation of spatialized variables.*
- void [deallocation](#) ()  
*Deallocation of variables.*
- void [maincalcfux](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#) &)  
*Main calculation of the flux.*
- void [maincalcscheme](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [TAB](#) &, [SCALAR](#), [SCALAR](#), int)  
*Main calculation of the scheme.*
- void [boundary](#) ([TAB](#) &, [TAB](#) &, [TAB](#) &, [SCALAR](#), const int, const int)  
*Calls the boundary conditions and affects the boundary values.*
- [SCALAR](#) [froude\\_number](#) ([TAB](#), [TAB](#), [TAB](#))  
*Returns the Froude number.*
- virtual [~Scheme](#) ()  
*Destructor.*

### Protected Attributes

- const int [NXCELL](#)
- const int [NYCELL](#)
- const int [ORDER](#)
- const [SCALAR](#) [T](#)
- const int [NBTIMES](#)
- const int [SCHEME\\_TYPE](#)
- const [SCALAR](#) [DX](#)
- const [SCALAR](#) [DY](#)
- const [SCALAR](#) [CFL\\_FIX](#)
- [SCALAR](#) [DT\\_FIX](#)
- [SCALAR](#) [tx](#)
- [SCALAR](#) [ty](#)
- [SCALAR](#) [T\\_output](#)
- [SCALAR](#) [dt\\_output](#)



- const SCALAR FRICCOEF
- const SCALAR L\_IMP\_Q
- const SCALAR L\_IMP\_H
- const SCALAR R\_IMP\_Q
- const SCALAR R\_IMP\_H
- const SCALAR B\_IMP\_Q
- const SCALAR B\_IMP\_H
- const SCALAR T\_IMP\_Q
- const SCALAR T\_IMP\_H
- TAB z
- TAB h
- TAB u
- TAB v
- TAB q1
- TAB q2
- TAB hs
- TAB us
- TAB vs
- TAB qs1
- TAB qs2
- TAB f1
- TAB f2
- TAB f3
- TAB g1
- TAB g2
- TAB g3
- TAB h1left
- TAB h1right
- TAB h2left
- TAB h2right
- TAB delz1
- TAB delz2
- TAB delzc1
- TAB delzc2
- TAB h1r
- TAB u1r
- TAB v1r
- TAB h1l
- TAB u1l
- TAB v1l
- TAB h2r
- TAB u2r
- TAB v2r
- TAB h2l
- TAB u2l
- TAB v2l
- TAB Rain
- TAB Vin\_tot
- time\_t start

- `time_t end`
- `SCALAR timecomputation`
- `clock_t cpu_time`
- `int n`
- `SCALAR Fr`
- `SCALAR dt1`
- `SCALAR dt_max`
- `SCALAR cur_time`
- `SCALAR dt_first`
- `SCALAR Volrain_Tot`
- `SCALAR Total_volume_outflow`
- `SCALAR height_of_tot`
- `SCALAR height_Vinf_tot`
- `SCALAR Vol_inf_tot_cumul`
- `SCALAR Vol_of_tot`
- `Choice_condition * Lbound`
- `Choice_condition * Rbound`
- `Choice_condition * Bbound`
- `Choice_condition * Tbound`
- `Choice_output * out`
- `int verif`

### 5.55.1 Detailed Description

Numerical scheme.

Class that contains all the common declarations for the numerical schemes.

Definition at line 112 of file `scheme.hpp`.

### 5.55.2 Constructor & Destructor Documentation

#### `Scheme::Scheme ( Parameters & par )`

Constructor.

Initializations and allocations.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 60 of file `scheme.cpp`.

#### `Scheme::~Scheme ( ) [virtual]`

Destructor.

Definition at line 719 of file `scheme.cpp`.

### 5.55.3 Member Function Documentation

#### `void Scheme::allocation ( )`

Allocation of spatialized variables.

Allocation of `Scheme::z`, `Scheme::h`, `Scheme::u`, `Scheme::v`, `Scheme::q1`, `Scheme::q2`, `Scheme::Vin_tot`, `Scheme::hs`, `Scheme::us`, `Scheme::vs`, `Scheme::qs1`, `Scheme::qs2`, `Scheme::f1`, `Scheme::f2`, `Scheme::f3`, `Scheme::g1`, `Scheme::g2`, `Scheme::g3`, `Scheme::h1left`, `Scheme::h1l`, `Scheme::u1l`, `Scheme::v1l`, `Scheme::h1right`, `Scheme::h1r`, `Scheme::u1r`, `Scheme::v1r`, `Scheme::h2left`, `Scheme::h2l`, `Scheme::u2l`, `Scheme::v2l`,

[Scheme::h2right](#), [Scheme::h2r](#), [Scheme::u2r](#), [Scheme::v2r](#), [Scheme::delz1](#), [Scheme::delz2](#), [Scheme::delzc1](#), [Scheme::delzc2](#), [Scheme::Rain](#).

Definition at line 444 of file scheme.cpp.

**void Scheme::boundary ( TAB & *h\_tmp*, TAB & *u\_tmp*, TAB & *v\_tmp*, SCALAR *time\_tmp*, const int *NODEX*, const int *NODEY* )**

Calls the boundary conditions and affects the boundary values.

Parameters

in, out	<i>h_tmp</i>	water height.
in, out	<i>u_tmp</i>	first component of the velocity.
in, out	<i>v_tmp</i>	second component of the velocity.
in	<i>time_tmp</i>	current time.
in	<i>NODEX</i>	number of space cells in the first (x) direction.
in	<i>NODEY</i>	number of space cells in the second (y) direction.

Definition at line 368 of file scheme.cpp.

**virtual void Scheme::calcul ( ) [pure virtual]**

Function to be specified in each numerical scheme.

Implemented in [Order1](#), and [Order2](#).

**void Scheme::deallocation ( )**

Deallocation of variables.

Deallocation of [Scheme::z](#), [Scheme::h](#), [Scheme::u](#), [Scheme::v](#), [Scheme::q1](#), [Scheme::q2](#), [Scheme::Vin](#), [Scheme::tot](#), [Scheme::hs](#), [Scheme::us](#), [Scheme::vs](#), [Scheme::qs1](#), [Scheme::qs2](#), [Scheme::f1](#), [Scheme::f2](#), [Scheme::f3](#), [Scheme::g1](#), [Scheme::g2](#), [Scheme::g3](#), [Scheme::h1left](#), [Scheme::h1](#), [Scheme::u1l](#), [Scheme::v1l](#), [Scheme::h1right](#), [Scheme::h1r](#), [Scheme::u1r](#), [Scheme::v1r](#), [Scheme::h2left](#), [Scheme::h2l](#), [Scheme::u2l](#), [Scheme::v2l](#), [Scheme::h2right](#), [Scheme::h2r](#), [Scheme::u2r](#), [Scheme::v2r](#), [Scheme::delz1](#), [Scheme::delz2](#), [Scheme::delzc1](#), [Scheme::delzc2](#), [Scheme::Rain](#).

Definition at line 582 of file scheme.cpp.

**SCALAR Scheme::froude\_number ( TAB *h\_s*, TAB *u\_s*, TAB *v\_s* )**

Returns the Froude number.

Mean value in space of the Froude number at the final time.

Parameters

in	<i>h_s</i>	water height.
in	<i>u_s</i>	first component of the velocity.
in	<i>v_s</i>	second component of the velocity.

Returns

The mean Froude number  $\frac{\sqrt{u_s^2 + v_s^2}}{\sqrt{gh_s}}$ .

Definition at line 407 of file scheme.cpp.

**void Scheme::maincalcfux ( SCALAR *cffix*, SCALAR *T*, SCALAR *curtime*, SCALAR *dt\_max*, SCALAR *dt*, SCALAR & *dt\_cal* )**

Main calculation of the flux.

First part. Construction of variables for hydrostatic reconstruction. Fluxes in the two directions. Computation of the time step for the fixed cfl. This calculation is called once at the order 1, and twice at the second order.

**Parameters**

in	<i>cflfix</i>	fixed cfl.
in	<i>T</i>	final time (unused).
in	<i>curtime</i>	current time.
in	<i>dt_max</i>	maximum value of the time step.
in	<i>dt</i>	time step.
out	<i>dt_cal</i>	effective time step.

**Warning**

the CFL condition is not satisfied: CFL > \*\*\*

Definition at line 155 of file scheme.cpp.

**void Scheme::maincalcscheme ( TAB & *he*, TAB & *ve1*, TAB & *ve2*, TAB & *qe1*, TAB & *qe2*, TAB & *hes*, TAB & *ves1*, TAB & *ves2*, TAB & *qes1*, TAB & *qes2*, TAB & *Vin*, SCALAR *curtime*, SCALAR *dt*, int *n* )**

Main calculation of the scheme.

Second part. Computation of h, u and v. This calculation is called once at the order 1, and twice at the second order.

**Parameters**

in	<i>he</i>	water height.
in	<i>ve1</i>	first component of the velocity.
in	<i>ve2</i>	second component of the velocity.
in	<i>qe1</i>	first component of the discharge (unused).
in	<i>qe2</i>	second component of the discharge (unused).
out	<i>hes</i>	water height.
out	<i>ves1</i>	first component of the velocity.
out	<i>ves2</i>	second component of the velocity.
out	<i>qes1</i>	first component of the discharge.
out	<i>qes2</i>	second component of the discharge.
out	<i>Vin</i>	infiltrated volume
in	<i>curtime</i>	current time.
in	<i>dt</i>	time step.
in	<i>n</i>	number of iterations (unused).

**Note**

In DEBUG mode, the programme will save three other files with boundaries fluxes.

Definition at line 235 of file scheme.cpp.

**5.55.4 Member Data Documentation**

**const SCALAR Scheme::B\_IMP\_H [protected]**

Imposed water height on the bottom boundary.

Definition at line 186 of file scheme.hpp.

**const SCALAR Scheme::B\_IMP\_Q [protected]**

Imposed discharge on the bottom boundary.

Definition at line 184 of file scheme.hpp.

**Choice\_condition\* Scheme::Bbound [protected]**

The choice of the bottom boundary condition.  
Definition at line 306 of file scheme.hpp.

**const SCALAR Scheme::CFL\_FIX [protected]**

Value of the fixed cfl.  
Definition at line 162 of file scheme.hpp.

**clock\_t Scheme::cpu\_time [protected]**

CPU time.  
Definition at line 276 of file scheme.hpp.

**SCALAR Scheme::cur\_time [protected]**

The current simulation time.  
Definition at line 286 of file scheme.hpp.

**TAB Scheme::delz1 [protected]**

Variations of the topography along x.  
Definition at line 234 of file scheme.hpp.

**TAB Scheme::delz2 [protected]**

Variations of the topography along y.  
Definition at line 236 of file scheme.hpp.

**TAB Scheme::delzc1 [protected]**

Difference between the reconstructed topographies on the left and on the right boundary of a cell along x.  
Definition at line 238 of file scheme.hpp.

**TAB Scheme::delzc2 [protected]**

Difference between the reconstructed topographies on the left and on the right boundary of a cell along y.  
Definition at line 240 of file scheme.hpp.

**SCALAR Scheme::dt1 [protected]**

Time step in case of fixed cfl.  
Definition at line 282 of file scheme.hpp.

**SCALAR Scheme::dt\_first [protected]**

Space step in the first step in the method Heun.  
Definition at line 288 of file scheme.hpp.

**SCALAR Scheme::DT\_FIX [protected]**

Value of the fixed time step.  
Definition at line 164 of file scheme.hpp.

**SCALAR Scheme::dt\_max [protected]**

Maximum time step in case of fixed cfl.

Definition at line 284 of file scheme.hpp.

**SCALAR Scheme::dt\_output [protected]**

Time step to save the data (evolution file).

Definition at line 172 of file scheme.hpp.

**const SCALAR Scheme::DX [protected]**

Space step in the first (x) direction.

Definition at line 158 of file scheme.hpp.

**const SCALAR Scheme::DY [protected]**

Space step in the second (y) direction.

Definition at line 160 of file scheme.hpp.

**time\_t Scheme::end [protected]**

End of timer.

Definition at line 272 of file scheme.hpp.

**TAB Scheme::f1 [protected]**

First component of the numerical flux along x.

Definition at line 214 of file scheme.hpp.

**TAB Scheme::f2 [protected]**

Second component of the numerical flux along x.

Definition at line 216 of file scheme.hpp.

**TAB Scheme::f3 [protected]**

Third component of the numerical flux along x.

Definition at line 218 of file scheme.hpp.

**SCALAR Scheme::Fr [protected]**

Mean Froude number.

Definition at line 280 of file scheme.hpp.

**const SCALAR Scheme::FRICCOEF [protected]**

Friction coefficient.

Definition at line 174 of file scheme.hpp.

**TAB Scheme::g1 [protected]**

First component of the numerical flux along y.

Definition at line 220 of file scheme.hpp.

**TAB Scheme::g2 [protected]**

Second component of the numerical flux along y.  
Definition at line 222 of file scheme.hpp.

**TAB Scheme::g3 [protected]**

Third component of the numerical flux along y.  
Definition at line 224 of file scheme.hpp.

**TAB Scheme::h [protected]**

Water height.  
Definition at line 194 of file scheme.hpp.

**TAB Scheme::h1l [protected]**

Water height on the cell located at the left of the boundary along x.  
Definition at line 248 of file scheme.hpp.

**TAB Scheme::h1left [protected]**

Hydrostatic reconstruction on the left along x.  
Definition at line 226 of file scheme.hpp.

**TAB Scheme::h1r [protected]**

Water height on the cell located at the right of the boundary along x.  
Definition at line 242 of file scheme.hpp.

**TAB Scheme::h1right [protected]**

Hydrostatic reconstruction on the right along x.  
Definition at line 228 of file scheme.hpp.

**TAB Scheme::h2l [protected]**

Water height on the cell located at the left of the boundary along y.  
Definition at line 260 of file scheme.hpp.

**TAB Scheme::h2left [protected]**

Hydrostatic reconstruction on the left along y.  
Definition at line 230 of file scheme.hpp.

**TAB Scheme::h2r [protected]**

Water height on the cell located at the right of the boundary along y.  
Definition at line 254 of file scheme.hpp.

**TAB Scheme::h2right [protected]**

Hydrostatic reconstruction on the right along y.  
Definition at line 232 of file scheme.hpp.

**SCALAR Scheme::height\_of\_tot [protected]**

Cumulative water height on the whole domain  
Definition at line 294 of file scheme.hpp.

**SCALAR Scheme::height\_Vinf\_tot [protected]**

Cumulative height of infiltrated water on the whole domain  
Definition at line 296 of file scheme.hpp.

**TAB Scheme::hs [protected]**

Water height after one step of the scheme.  
Definition at line 204 of file scheme.hpp.

**const SCALAR Scheme::L\_IMP\_H [protected]**

Imposed water height on the left boundary.  
Definition at line 178 of file scheme.hpp.

**const SCALAR Scheme::L\_IMP\_Q [protected]**

Imposed discharge on the left boundary.  
Definition at line 176 of file scheme.hpp.

**Choice\_condition\* Scheme::Lbound [protected]**

The choice of the left boundary condition.  
Definition at line 302 of file scheme.hpp.

**int Scheme::n [protected]**

Iterator for the loop in time.  
Definition at line 278 of file scheme.hpp.

**const int Scheme::NBTIMES [protected]**

Number of times saved.  
Definition at line 154 of file scheme.hpp.

**const int Scheme::NXCELL [protected]**

Number of cells in space in the first (x) direction.  
Definition at line 146 of file scheme.hpp.

**const int Scheme::NYCELL [protected]**

Number of cells in space in the second (y) direction.  
Definition at line 148 of file scheme.hpp.

**const int Scheme::ORDER [protected]**

Order of the numerical scheme.  
Definition at line 150 of file scheme.hpp.



**Choice\_output\* Scheme::out [protected]**

The choice of output.

Definition at line 310 of file scheme.hpp.

**TAB Scheme::q1 [protected]**

First component of the discharge.

Definition at line 200 of file scheme.hpp.

**TAB Scheme::q2 [protected]**

Second component of the discharge.

Definition at line 202 of file scheme.hpp.

**TAB Scheme::qs1 [protected]**

First component of the discharge after one step of the scheme.

Definition at line 210 of file scheme.hpp.

**TAB Scheme::qs2 [protected]**

Second component of the discharge after one step of the scheme.

Definition at line 212 of file scheme.hpp.

**const SCALAR Scheme::R\_IMP\_H [protected]**

Imposed water height on the right boundary.

Definition at line 182 of file scheme.hpp.

**const SCALAR Scheme::R\_IMP\_Q [protected]**

Imposed discharge on the right boundary.

Definition at line 180 of file scheme.hpp.

**TAB Scheme::Rain [protected]**

Rain intensity.

Definition at line 266 of file scheme.hpp.

**Choice\_condition\* Scheme::Rbound [protected]**

The choice of the right boundary condition.

Definition at line 304 of file scheme.hpp.

**const int Scheme::SCHEME\_TYPE [protected]**

Type of scheme (fixed cfl or time step).

Definition at line 156 of file scheme.hpp.

**time\_t Scheme::start [protected]**

Beginning of timer.

Definition at line 270 of file scheme.hpp.

**const SCALAR Scheme::T [protected]**

Final time.

Definition at line 152 of file scheme.hpp.

**const SCALAR Scheme::T\_IMP\_H [protected]**

Imposed water height on the top boundary.

Definition at line 190 of file scheme.hpp.

**const SCALAR Scheme::T\_IMP\_Q [protected]**

Imposed discharge on the top boundary.

Definition at line 188 of file scheme.hpp.

**SCALAR Scheme::T\_output [protected]**

Time to save the data (evolution file).

Definition at line 170 of file scheme.hpp.

**Choice\_condition\* Scheme::Tbound [protected]**

The choice of the top boundary condition.

Definition at line 308 of file scheme.hpp.

**SCALAR Scheme::timecomputation [protected]**

Duration of the computation.

Definition at line 274 of file scheme.hpp.

**SCALAR Scheme::Total\_volume\_outflow [protected]**

Cumulative outflow volume at the boundary.

Definition at line 292 of file scheme.hpp.

**SCALAR Scheme::tx [protected]**

Ratio dt/dx.

Definition at line 166 of file scheme.hpp.

**SCALAR Scheme::ty [protected]**

Ratio dt/dy.

Definition at line 168 of file scheme.hpp.

**TAB Scheme::u [protected]**

First component of the velocity.

Definition at line 196 of file scheme.hpp.

**TAB Scheme::u1l [protected]**

First component of the velocity on the cell located at the left of the boundary along x.

Definition at line 250 of file scheme.hpp.

**TAB Scheme::u1r [protected]**

First component of the velocity on the cell located at the right of the boundary along x.  
Definition at line 244 of file scheme.hpp.

**TAB Scheme::u2l [protected]**

First component of the velocity on the cell located at the left of the boundary along y.  
Definition at line 262 of file scheme.hpp.

**TAB Scheme::u2r [protected]**

First component of the velocity on the cell located at the right of the boundary along y.  
Definition at line 256 of file scheme.hpp.

**TAB Scheme::us [protected]**

First component of the velocity after one step of the scheme.  
Definition at line 206 of file scheme.hpp.

**TAB Scheme::v [protected]**

Second component of the velocity.  
Definition at line 198 of file scheme.hpp.

**TAB Scheme::v1l [protected]**

Second component of the velocity on the cell located at the left of the boundary along x.  
Definition at line 252 of file scheme.hpp.

**TAB Scheme::v1r [protected]**

Second component of the velocity on the cell located at the right of the boundary along x.  
Definition at line 246 of file scheme.hpp.

**TAB Scheme::v2l [protected]**

Second component of the velocity on the cell located at the left of the boundary along y.  
Definition at line 264 of file scheme.hpp.

**TAB Scheme::v2r [protected]**

Second component of the velocity on the cell located at the right of the boundary along y.  
Definition at line 258 of file scheme.hpp.

**int Scheme::verif [protected]**

Flag for the time step.  
Definition at line 312 of file scheme.hpp.

**TAB Scheme::Vin\_tot [protected]**

Cumulative volume of infiltrated water (at each point).  
Definition at line 268 of file scheme.hpp.

**SCALAR Scheme::Vol\_inf\_tot\_cumul [protected]**

Cumulative volume of water infiltrated.

Definition at line 298 of file scheme.hpp.

**SCALAR Scheme::Vol\_of\_tot [protected]**

Cumulative streammed volume.

Definition at line 300 of file scheme.hpp.

**SCALAR Scheme::Volrain\_Tot [protected]**

Cumulative volume of rain on the whole domain.

Definition at line 290 of file scheme.hpp.

**TAB Scheme::vs [protected]**

Second componant of the velocity after one step of the scheme.

Definition at line 208 of file scheme.hpp.

**TAB Scheme::z [protected]**

Topography.

Definition at line 192 of file scheme.hpp.

The documentation for this class was generated from the following files:

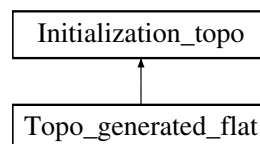
- Headers/libschemas/[scheme.hpp](#)
- Sources/libschemas/[scheme.cpp](#)

**5.56 Topo\_generated\_flat Class Reference**

Flat configuration.

```
#include <topo_generated_flat.hpp>
```

Inheritance diagram for Topo\_generated\_flat:

**Public Member Functions**

- [Topo\\_generated\\_flat](#) ([Parameters](#) &)  
*Constructor.*
- void [initialization](#) ([TAB](#) &)  
*Performs the initialization.*
- virtual [~Topo\\_generated\\_flat](#) ()  
*Destructor.*

## Additional Inherited Members

### 5.56.1 Detailed Description

Flat configuration.

Class that initializes a flat topography, with value 0.

Definition at line 73 of file topo\_generated\_flat.hpp.

### 5.56.2 Constructor & Destructor Documentation

#### Topo\_generated\_flat::Topo\_generated\_flat ( Parameters & *par* )

Constructor.

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file (unused).
-----------	------------	---

Definition at line 60 of file topo\_generated\_flat.cpp.

#### Topo\_generated\_flat::~~Topo\_generated\_flat ( ) [virtual]

Destructor.

Definition at line 84 of file topo\_generated\_flat.cpp.

### 5.56.3 Member Function Documentation

#### void Topo\_generated\_flat::initialization ( TAB & *topo* ) [virtual]

Performs the initialization.

Initializes the topography to 0.

Parameters

<i>in, out</i>	<i>topo</i>	topography.
----------------	-------------	-------------

Implements [Initialization\\_topo](#).

Definition at line 69 of file topo\_generated\_flat.cpp.

The documentation for this class was generated from the following files:

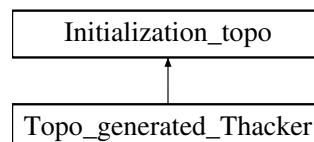
- [Headers/libinitializations/topo\\_generated\\_flat.hpp](#)
- [Sources/libinitializations/topo\\_generated\\_flat.cpp](#)

## 5.57 Topo\_generated\_Thacker Class Reference

Thacker configuration.

```
#include <topo_generated_thacker.hpp>
```

Inheritance diagram for Topo\_generated\_Thacker:



### Public Member Functions

- [Topo\\_generated\\_Thacker](#) (Parameters &)  
*Constructor.*
- void [initialization](#) (TAB &)

*Performs the initialization.*

- virtual `~Topo_generated_Thacker ()`

*Destructor.*

## Additional Inherited Members

### 5.57.1 Detailed Description

Thacker configuration.

Class that initializes a topography for Thacker's benchmark (shape of a paraboloid of revolution).

Definition at line 73 of file `topo_generated_thacker.hpp`.

### 5.57.2 Constructor & Destructor Documentation

#### `Topo_generated_Thacker::Topo_generated_Thacker ( Parameters & par )`

Constructor.

Defines the parameters of the paraboloid.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file (unused).
-----------------	------------------	---

Definition at line 60 of file `topo_generated_thacker.cpp`.

#### `Topo_generated_Thacker::~~Topo_generated_Thacker ( ) [virtual]`

Destructor.

Definition at line 98 of file `topo_generated_thacker.cpp`.

### 5.57.3 Member Function Documentation

#### `void Topo_generated_Thacker::initialization ( TAB & topo ) [virtual]`

Performs the initialization.

Initializes the topography to  $h_0 \left( \frac{(x-Lx/2)^2 + (y-Ly/2)^2}{a^2} - 1 \right)$ , see [Thacker \[1981\]](#).

Parameters

<code>in, out</code>	<code>topo</code>	topography.
----------------------	-------------------	-------------

Implements [Initialization\\_topo](#).

Definition at line 78 of file `topo_generated_thacker.cpp`.

The documentation for this class was generated from the following files:

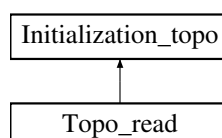
- Headers/libinitializations/[topo\\_generated\\_thacker.hpp](#)
- Sources/libinitializations/[topo\\_generated\\_thacker.cpp](#)

## 5.58 Topo\_read Class Reference

File configuration.

```
#include <topo_read.hpp>
```

Inheritance diagram for `Topo_read`:



## Public Member Functions

- [Topo\\_read](#) ([Parameters](#) &)  
*Constructor.*
- void [initialization](#) ([TAB](#) &)  
*Performs the initialization.*
- virtual [~Topo\\_read](#) ()  
*Destructor.*

## Additional Inherited Members

### 5.58.1 Detailed Description

File configuration.

Class that initializes the topography to the values read in a file.  
Definition at line 72 of file `topo_read.hpp`.

### 5.58.2 Constructor & Destructor Documentation

#### `Topo_read::Topo_read ( Parameters & par )`

Constructor.

Defines the name of the file for the initialization.

Parameters

<code>in</code>	<code>par</code>	parameter, contains all the values from the parameters file.
-----------------	------------------	--

Definition at line 60 of file `topo_read.cpp`.

#### `Topo_read::~~Topo_read ( ) [virtual]`

Destructor.

Definition at line 187 of file `topo_read.cpp`.

### 5.58.3 Member Function Documentation

#### `void Topo_read::initialization ( TAB & topo ) [virtual]`

Performs the initialization.

Initializes the topography to the values read in the corresponding file.

Parameters

<code>in, out</code>	<code>topo</code>	topography.
----------------------	-------------------	-------------

Warning

```
(huv_namefile): ERROR: cannot open the topography file.
(huv_namefile): ERROR: the number of data in this file is too big
(huv_namefile): ERROR: line ***.
(huv_namefile): WARNING: line ***.
(huv_namefile): ERROR: the number of data in this file is too small
(huv_namefile): ERROR: the value for the point x *** y *** is missing
```

## Note

If the file cannot be opened or if the data are not correct, the code will exit with failure termination code.

Implements [Initialization\\_topo](#).

Definition at line 72 of file `topo_read.cpp`.

The documentation for this class was generated from the following files:

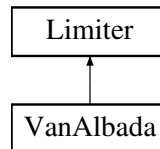
- [Headers/libinitializations/topo\\_read.hpp](#)
- [Sources/libinitializations/topo\\_read.cpp](#)

## 5.59 VanAlbada Class Reference

Van Albada slope limiter.

```
#include <vanalbada.hpp>
```

Inheritance diagram for VanAlbada:



### Public Member Functions

- [VanAlbada](#) ()  
*Constructor.*
- void [calcul](#) (SCALAR, SCALAR)  
*Calculates the value of the slope limiter.*
- virtual [~VanAlbada](#) ()  
*Destructor.*

### Additional Inherited Members

#### 5.59.1 Detailed Description

Van Albada slope limiter.

Class that calculates Van Albada slope limiter.

Definition at line 70 of file `vanalbada.hpp`.

#### 5.59.2 Constructor & Destructor Documentation

##### **VanAlbada::VanAlbada** ( )

Constructor.

Definition at line 59 of file `vanalbada.cpp`.

##### **VanAlbada::~~VanAlbada** ( ) [**virtual**]

Destructor.

Definition at line 85 of file `vanalbada.cpp`.



### 5.59.3 Member Function Documentation

**void VanAlbada::calcul ( SCALAR a, SCALAR b ) [virtual]**

Calculates the value of the slope limiter.

Van Albada function:

$$VA(x,y) = \begin{cases} 0 & \text{if } \text{sign}(x) \neq \text{sign}(y), \\ \frac{x(y^2 + \varepsilon) + y(x^2 + \varepsilon)}{x^2 + y^2 + 2\varepsilon} & \text{else,} \end{cases}$$

with  $0 \leq \varepsilon \ll 1$ .

Parameters

in	a	slope on the left of the cell.
in	b	slope on the right of the cell.

Modifies

[Limiter::rec](#) reconstructed value.

Implements [Limiter](#).

Definition at line 62 of file `vanalbada.cpp`.

The documentation for this class was generated from the following files:

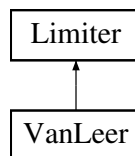
- Headers/liblimitations/[vanalbada.hpp](#)
- Sources/liblimitations/[vanalbada.cpp](#)

## 5.60 VanLeer Class Reference

Van Leer slope limiter.

```
#include <vanleer.hpp>
```

Inheritance diagram for VanLeer:



### Public Member Functions

- [VanLeer \(\)](#)  
*Constructor.*
- void [calcul \(SCALAR, SCALAR\)](#)  
*Calculates the value of the slope limiter.*
- virtual [~VanLeer \(\)](#)  
*Destructor.*

### Additional Inherited Members

#### 5.60.1 Detailed Description

Van Leer slope limiter.

Class that calculates Van Leer slope limiter.

Definition at line 70 of file `vanleer.hpp`.

### 5.60.2 Constructor & Destructor Documentation

**VanLeer::VanLeer ( )**

Constructor.

Definition at line 59 of file vanleer.cpp.

**VanLeer::~~VanLeer ( ) [virtual]**

Destructor.

Definition at line 84 of file vanleer.cpp.

### 5.60.3 Member Function Documentation

**void VanLeer::calcul ( SCALAR a, SCALAR b ) [virtual]**

Calculates the value of the slope limiter.

Van Leer function:

$$VL(x,y) = \begin{cases} 0 & \text{if } xy \leq 0, \\ \frac{2xy}{x+y} & \text{else.} \end{cases}$$

Parameters

in	<i>a</i>	slope on the left of the cell.
in	<i>b</i>	slope on the right of the cell.

Modifies

[Limiter::rec](#) reconstructed value.

Implements [Limiter](#).

Definition at line 62 of file vanleer.cpp.

The documentation for this class was generated from the following files:

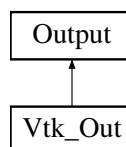
- Headers/liblimitations/[vanleer.hpp](#)
- Sources/liblimitations/[vanleer.cpp](#)

## 5.61 Vtk\_Out Class Reference

VTK output.

```
#include <vtk_out.hpp>
```

Inheritance diagram for Vtk\_Out:



### Public Member Functions

- [Vtk\\_Out \(Parameters &\)](#)  
*Constructor.*
- void [write \(TAB, TAB, TAB, TAB, SCALAR\)](#)  
*Saves one time step.*
- virtual [~Vtk\\_Out \(\)](#)  
*Destructor.*

## Additional Inherited Members

### 5.61.1 Detailed Description

VTK output.

Class that writes the result in the output file with a structure optimized for VTK.

Definition at line 71 of file vtk\_out.hpp.

### 5.61.2 Constructor & Destructor Documentation

#### Vtk\_Out::Vtk\_Out ( Parameters & *par* )

Constructor.

Defines the output name huz\_evolution.dat

Parameters

<i>in</i>	<i>par</i>	parameter, contains all the values from the parameters file.
-----------	------------	--

Definition at line 60 of file vtk\_out.cpp.

#### Vtk\_Out::~~Vtk\_Out ( ) [virtual]

Destructor.

Definition at line 259 of file vtk\_out.cpp.

### 5.61.3 Member Function Documentation

#### void Vtk\_Out::write ( TAB *h*, TAB *u*, TAB *v*, TAB *z*, SCALAR *time* ) [virtual]

Saves one time step.

Writes the values of [Scheme::z](#), [Scheme::h](#), [Scheme::u](#) (=q1/h), [Scheme::v](#) (=q2/h), [Scheme::h+ Scheme::z](#) (free surface),  $|U| = \sqrt{u^2 + v^2}$ , the Froude number  $\frac{|U|}{\sqrt{gh}}$ , [Scheme::q1](#), [Scheme::q2](#), and  $h|U|$  at the current time in huz\_evolution.dat\*\*\*.vtk.

If the water height is too small, we replace it by 0, the velocity is null and the Froude number does not exist.

Parameters

<i>in</i>	<i>h</i>	the water height.
<i>in</i>	<i>u</i>	first component of the velocity.
<i>in</i>	<i>v</i>	second component of the velocity.
<i>in</i>	<i>z</i>	the topography.
<i>in</i>	<i>time</i>	the current time.

Note

If huz\_evolution.dat\*\*\*.vtk cannot be opened, the code will exit with failure termination code.

Implements [Output](#).

Definition at line 76 of file vtk\_out.cpp.

The documentation for this class was generated from the following files:

- Headers/libsave/vtk\_out.hpp
- Sources/libsave/vtk\_out.cpp



# Chapter 6

## File Documentation

### 6.1 Headers/libboundaryconditions/bc\_imp\_discharge.hpp File Reference

Imposed discharge.

```
#include "boundary_condition.hpp"
```

#### Classes

- class [Bc\\_imp\\_discharge](#)  
*Imposed discharge.*

#### Macros

- #define [BC\\_IMP\\_DISCHARGE\\_HPP](#)

#### 6.1.1 Detailed Description

Imposed discharge.

Author

Ulrich Razafison [ulrich.razafison@math.cnrs.fr](mailto:ulrich.razafison@math.cnrs.fr) (2011)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: imposed discharge (and water height if necessary).

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

### 6.1.2 Macro Definition Documentation

```
#define BC_IMP_DISCHARGE_HPP
```

Definition at line 63 of file `bc_imp_discharge.hpp`.

## 6.2 Headers/`libboundaryconditions/bc_imp_height.hpp` File Reference

Imposed water height.

```
#include "boundary_condition.hpp"
```

### Classes

- class [Bc\\_imp\\_height](#)  
*Imposed water height.*

### 6.2.1 Detailed Description

Imposed water height.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2010-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: imposed water height (and discharge if necessary), based on the modified method of characteristics and Riemann invariants.

See also

Olivier Delestre Ph.D thesis Annexe A [Delestre \[2010\]](#) <http://tel.archives-ouvertes.fr/tel-00587197>

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.3 Headers/`libboundaryconditions/bc_neumann.hpp` File Reference

Neumann condition.

```
#include "boundary_condition.hpp"
```

## Classes

- class [Bc\\_Neumann](#)  
*Neumann condition.*

### 6.3.1 Detailed Description

Neumann condition.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: Neumann condition (the normal derivative is null).

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.4 Headers/libboundaryconditions/bc\_periodic.hpp File Reference

Periodic condition.

```
#include "boundary_condition.hpp"
```

## Classes

- class [Bc\\_periodic](#)  
*Periodic condition.*

### 6.4.1 Detailed Description

Periodic condition.

Author

Pierre-Antoine Ksinant [pierreantoine.ksinantgarcia@gmail.com](mailto:pierreantoine.ksinantgarcia@gmail.com) (2010)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: periodic condition.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.5 Headers/libboundaryconditions/bc\_wall.hpp File Reference

Wall condition.

```
#include "boundary_condition.hpp"
```

### Classes

- class [Bc\\_wall](#)  
*Wall condition.*

### 6.5.1 Detailed Description

Wall condition.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: wall condition (the discharge at the boundary is null).

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.6 Headers/libboundaryconditions/boundary\_condition.hpp File Reference

Boundary condition.

```
#include "parameters.hpp"
```



## Classes

- class [Boundary\\_condition](#)  
*Boundary condition.*

### 6.6.1 Detailed Description

Boundary condition.

#### Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

#### Version

1.06.00

#### Date

2015-02-19

Common part for all the boundary conditions.

#### Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.7 Headers/libboundaryconditions/choice\_condition.hpp File Reference

Choice of boundary condition.

```
#include "boundary_condition.hpp"  
#include "bc_imp_height.hpp"  
#include "bc_wall.hpp"  
#include "bc_neumann.hpp"  
#include "bc_periodic.hpp"  
#include "bc_imp_discharge.hpp"
```

## Classes

- class [Choice\\_condition](#)  
*Choice of boundary condition.*

## Macros

- #define [CHOICE\\_CONDITION\\_HPP](#)

### 6.7.1 Detailed Description

Choice of boundary condition.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen boundary condition.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

### 6.7.2 Macro Definition Documentation

**#define CHOICE\_CONDITION\_HPP**

Definition at line 84 of file choice\_condition.hpp.

## 6.8 Headers/libflux/choice\_flux.hpp File Reference

Choice of numerical flux.

```
#include "flux.hpp"
#include "f_rusanov.hpp"
#include "f_h11.hpp"
#include "f_h112.hpp"
```

### Classes

- class [Choice\\_flux](#)  
*Choice of numerical flux.*

### Macros

- #define [CHOICE\\_FLUX\\_HPP](#)

#### 6.8.1 Detailed Description

Choice of numerical flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

From the value of the corresponding parameter, calls the chosen numerical flux.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.8.2 Macro Definition Documentation

### #define CHOICE\_FLUX\_HPP

Definition at line 76 of file choice\_flux.hpp.

## 6.9 Headers/libflux/f\_hll.hpp File Reference

HLL flux.

```
#include "flux.hpp"
```

### Classes

- class [F\\_HLL](#)

*HLL flux.*

### 6.9.1 Detailed Description

HLL flux.

## Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Numerical flux: Harten, Lax, van Leer reduced formulation.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.10 Headers/libflux/f\_hll2.hpp File Reference

HLL flux.

```
#include "flux.hpp"
```

### Classes

- class [F\\_HLL2](#)  
*HLL flux.*

### 6.10.1 Detailed Description

HLL flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical flux: Harten, Lax, van Leer reduced formulation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.11 Headers/libflux/f\_rusanov.hpp File Reference

Rusanov flux.

```
#include "flux.hpp"
```

### Classes

- class [F\\_Rusanov](#)  
*Rusanov flux.*

### 6.11.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical flux: Rusanov formulation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.12 Headers/libflux/flux.hpp File Reference

Numerical flux.

```
#include "parameters.hpp"
```

### Classes

- class [Flux](#)  
*Numerical flux.*

### 6.12.1 Detailed Description

Numerical flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the numerical fluxes.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.13 Headers/libfrictions/choice\_friction.hpp File Reference

Choice of friction law.

```
#include "friction.hpp"  
#include "fr_manning.hpp"  
#include "fr_darcy_weisbach.hpp"  
#include "no_friction.hpp"  
#include "fr_laminar.hpp"
```

### Classes

- class [Choice\\_friction](#)  
*Choice of friction law.*

### Macros

- #define [CHOICE\\_FRICTION\\_HPP](#)

#### 6.13.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen friction law.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

#### 6.13.2 Macro Definition Documentation

**#define CHOICE\_FRICTION\_HPP**

Definition at line 80 of file choice\_friction.hpp.

## 6.14 Headers/libfrictions/fr\_darcy\_weisbach.hpp File Reference

Darcy-Weisbach law.

```
#include "friction.hpp"
```

## Classes

- class [Fr\\_Darcy\\_Weisbach](#)  
*Darcy-Weisbach law.*

### 6.14.1 Detailed Description

Darcy-Weisbach law.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Darcy-Weisbach.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.15 Headers/libfrictions/fr\_laminar.hpp File Reference

laminar law

```
#include "friction.hpp"
```

## Classes

- class [Fr\\_Laminar](#)  
*Laminar law.*

### 6.15.1 Detailed Description

laminar law

Author

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2014-2015)

Version

1.06.00

Date

2015-02-19

Friction law: laminar.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - INRA (France)

## 6.16 Headers/libfrictions/fr\_manning.hpp File Reference

Manning law.

```
#include "friction.hpp"
```

### Classes

- class [Fr\\_Manning](#)  
*Manning law.*

### 6.16.1 Detailed Description

Manning law.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Manning.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.17 Headers/libfrictions/friction.hpp File Reference

Friction law

```
#include "parameters.hpp"
```

### Classes

- class [Friction](#)  
*Friction law*



### 6.17.1 Detailed Description

Friction law

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the friction laws.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.18 Headers/libfrictions/no\_friction.hpp File Reference

No friction.

```
#include "friction.hpp"
```

### Classes

- class [No\\_Friction](#)

*No friction.*

### 6.18.1 Detailed Description

No friction.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: does no computation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.19 Headers/libinitializations/choice\_init\_huv.hpp File Reference

Choice of initialization for h, u and v.

```
#include "initialization_huv.hpp"
#include "huv_read.hpp"
#include "huv_generated.hpp"
#include "huv_generated_thacker.hpp"
#include "huv_generated_radial_dam_dry.hpp"
#include "huv_generated_radial_dam_wet.hpp"
```

### Classes

- class [Choice\\_init\\_huv](#)

*Choice of initialization for h and  $U=(u,v)$*

### Macros

- #define [CHOICE\\_INIT\\_HUV\\_HPP](#)

#### 6.19.1 Detailed Description

Choice of initialization for h, u and v.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the water height and of the velocity.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

#### 6.19.2 Macro Definition Documentation

**#define CHOICE\_INIT\_HUV\_HPP**

Definition at line 83 of file choice\_init\_huv.hpp.

## 6.20 Headers/libinitializations/choice\_init\_topo.hpp File Reference

Choice of initialization for the topography.

```
#include "initialization_topo.hpp"
#include "topo_read.hpp"
#include "topo_generated_flat.hpp"
#include "topo_generated_thacker.hpp"
```

### Classes

- class [Choice\\_init\\_topo](#)  
*Choice of initialization for the topography.*

### Macros

- #define [CHOICE\\_INIT\\_TOPO\\_HPP](#)

#### 6.20.1 Detailed Description

Choice of initialization for the topography.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the topography.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

#### 6.20.2 Macro Definition Documentation

**#define CHOICE\_INIT\_TOPO\_HPP**

Definition at line 75 of file choice\_init\_topo.hpp.

## 6.21 Headers/libinitializations/huv\_generated.hpp File Reference

No water configuration.

```
#include "initialization_huv.hpp"
```

## Classes

- class [Huv\\_generated](#)  
*No water configuration.*

### 6.21.1 Detailed Description

No water configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a dry domain.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.22 Headers/libinitializations/huv\_generated\_radial\_dam\_dry.hpp File Reference

Dry radial dam break configuration.

```
#include "initialization_huv.hpp"
```

## Classes

- class [Huv\\_generated\\_Radial\\_Dam\\_dry](#)  
*Dry radial dam break configuration.*

### 6.22.1 Detailed Description

Dry radial dam break configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a radial dam break on a dry domain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.23 Headers/libinitializations/huv\_generated\_radial\_dam\_wet.hpp File Reference

Wet radial dam break configuration.

```
#include "initialization_huv.hpp"
```

### Classes

- class [Huv\\_generated\\_Radial\\_Dam\\_wet](#)  
*Wet radial dam break configuration.*

### 6.23.1 Detailed Description

Wet radial dam break configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a radial dam break on a wet domain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.24 Headers/libinitializations/huv\_generated\_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_huv.hpp"
```

## Classes

- class [Huv\\_generated\\_Thacker](#)  
*Thacker configuration.*

### 6.24.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of Thacker's benchmark.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.25 Headers/libinitializations/huv\_read.hpp File Reference

File configuration.

```
#include "initialization_huv.hpp"
```

## Classes

- class [Huv\\_read](#)  
*File configuration.*

### 6.25.1 Detailed Description

File configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

## Date

2015-02-19

Initialization of the water height and of the velocity: the values are read in a file.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.26 Headers/libinitializations/initialization\_huv.hpp File Reference

Initialization of h, u and v

```
#include "parameters.hpp"
```

### Classes

- class [Initialization\\_huv](#)  
*Initialization of h, u and v.*

#### 6.26.1 Detailed Description

Initialization of h, u and v

## Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Common part for all the initialization of the water height and of the velocity.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.27 Headers/libinitializations/initialization\_topo.hpp File Reference

Initialization of z

```
#include "parameters.hpp"
```

## Classes

- class [Initialization\\_topo](#)  
*Initialization of z.*

### 6.27.1 Detailed Description

Initialization of z

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2010)

Version

1.06.00

Date

2015-02-19

Common part for all the initialization of the topography.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.28 Headers/libinitializations/topo\_generated\_flat.hpp File Reference

Flat configuration.

```
#include "initialization_topo.hpp"
```

## Classes

- class [Topo\\_generated\\_flat](#)  
*Flat configuration.*

### 6.28.1 Detailed Description

Flat configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00



Date

2015-02-19

Initialization of the topography: the topography is flat, its value is 0.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.29 Headers/libinitializations/topo\_generated\_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_topo.hpp"
```

### Classes

- class [Topo\\_generated\\_Thacker](#)

*Thacker configuration.*

### 6.29.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: topography with a shape of a paraboloid of revolution for Thacker's Benchmark.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.30 Headers/libinitializations/topo\_read.hpp File Reference

File configuration.

```
#include "initialization_topo.hpp"
```

**Classes**

- class [Topo\\_read](#)  
*File configuration.*

**6.30.1 Detailed Description**

File configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: the values are read in a file.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

**6.31 Headers/liblimitations/choice\_limiter.hpp File Reference**

Choice of slope limiter.

```
#include "limiter.hpp"
#include "minmod.hpp"
#include "vanalbada.hpp"
#include "vanleer.hpp"
```

**Classes**

- class [Choice\\_limiter](#)  
*Choice of slope limiter.*

**Macros**

- #define [CHOICE\\_LIMITER\\_HPP](#)

**6.31.1 Detailed Description**

Choice of slope limiter.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen slope limiter.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

### 6.31.2 Macro Definition Documentation

**#define CHOICE\_LIMITER\_HPP**

Definition at line 75 of file choice\_limiter.hpp.

## 6.32 Headers/liblimitations/limiter.hpp File Reference

Slope limiter.

```
#include "parameters.hpp"
```

### Classes

- class [Limiter](#)  
*Slope limiter.*

### 6.32.1 Detailed Description

Slope limiter.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the slope limiters.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.33 Headers/liblimitations/minmod.hpp File Reference

Minmod limiter

```
#include "limiter.hpp"
```

### Classes

- class [Minmod](#)  
*Minmod slope limiter*

### 6.33.1 Detailed Description

Minmod limiter

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: minmod.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.34 Headers/liblimitations/vanalbada.hpp File Reference

Van Albada limiter.

```
#include "limiter.hpp"
```

### Classes

- class [VanAlbada](#)  
*Van Albada slope limiter.*

### 6.34.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Slope limiter: Van Albada.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.35 Headers/liblimitations/vanleer.hpp File Reference

Van Leer limiter.

```
#include "limiter.hpp"
```

### Classes

- class [VanLeer](#)

*Van Leer slope limiter.*

### 6.35.1 Detailed Description

Van Leer limiter.

## Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Slope limiter: Van Leer.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.36 Headers/libparameters/misc.hpp File Reference

Definitions.

```
#include <vector>
#include <iostream>
#include <cmath>
#include <fstream>
#include <string>
#include <cstring>
#include <cstdlib>
#include <iomanip>
#include <sstream>
#include <cstdio>
#include <unistd.h>
#include <ctime>
```

### Macros

- #define `max(a, b)`  $(a \geq b ? a : b)$
- #define `min(a, b)`  $(a \leq b ? a : b)$
- #define `GRAV` 9.81
- #define `GRAV_DEM` 4.905
- #define `CONST_CFL_X` 0.5
- #define `CONST_CFL_Y` 0.5
- #define `HE_CA` 1.e-12
- #define `VE_CA` 1.e-12
- #define `MAX_CFL_X` 0.
- #define `MAX_CFL_Y` 0.
- #define `MAX_ITER` 1000000000
- #define `NB_CHAR` 256
- #define `ZERO` 0.
- #define `IE_CA` 1.e-8
- #define `EPSILON` 1.e-13
- #define `VERSION` "FullSWOF\_2D version 1.06.00, 2015-03-02"
- #define `RATIO_CLOSE_CELL` 1.e-3
- #define `MAX_SCAL` DBL\_MAX

### Typedefs

- typedef double `SCALAR`
- typedef vector< vector< `SCALAR` > > `TAB`

#### 6.36.1 Detailed Description

Definitions.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-03-02

Defines the constants, the types used in the code and contains the 'include'.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - INRA (France)

### 6.36.2 Macro Definition Documentation

**#define CONST\_CFL\_X 0.5**

Definition at line 74 of file misc.hpp.

**#define CONST\_CFL\_Y 0.5**

Definition at line 75 of file misc.hpp.

**#define EPSILON 1.e-13**

Definition at line 85 of file misc.hpp.

**#define GRAV 9.81**

Definition at line 72 of file misc.hpp.

**#define GRAV\_DEM 4.905**

Definition at line 73 of file misc.hpp.

**#define HE\_CA 1.e-12**

Definition at line 76 of file misc.hpp.

**#define IE\_CA 1.e-8**

Definition at line 84 of file misc.hpp.

**#define max( a, b ) (a>=b?a:b)**

Definition at line 69 of file misc.hpp.

**#define MAX\_CFL\_X 0.**

Definition at line 78 of file misc.hpp.

**#define MAX\_CFL\_Y 0.**

Definition at line 79 of file misc.hpp.

```
#define MAX_ITER 100000000
```

Definition at line 80 of file misc.hpp.

```
#define MAX_SCAL DBL_MAX
```

Definition at line 96 of file misc.hpp.

```
#define min( a, b ) (a<=b?a:b)
```

Definition at line 70 of file misc.hpp.

```
#define NB_CHAR 256
```

Definition at line 82 of file misc.hpp.

```
#define RATIO_CLOSE_CELL 1.e-3
```

Definition at line 89 of file misc.hpp.

```
#define VE_CA 1.e-12
```

Definition at line 77 of file misc.hpp.

```
#define VERSION "FullSWOF_2D version 1.06.00, 2015-03-02"
```

Definition at line 86 of file misc.hpp.

```
#define ZERO 0.
```

Definition at line 83 of file misc.hpp.

### 6.36.3 Typedef Documentation

```
typedef double SCALAR
```

Definition at line 93 of file misc.hpp.

```
typedef vector< vector< SCALAR > > TAB
```

Definition at line 98 of file misc.hpp.

## 6.37 Headers/libparameters/parameters.hpp File Reference

Gets parameters.

```
#include "misc.hpp"  
#include "parser.hpp"
```

### Classes

- class [Parameters](#)

*Gets parameters.*



### 6.37.1 Detailed Description

Gets parameters.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2011-2015)

Version

1.06.00

Date

2015-02-19

Reads the parameters, checks their values.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.38 Headers/libparser/parser.hpp File Reference

Parser

```
#include "misc.hpp"
```

### Classes

- class [Parser](#)

*Parser to read the entries*

### 6.38.1 Detailed Description

Parser

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2010-2015)

Version

1.06.00

Date

2015-02-19

Reads the input file.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - INRA (France)

## 6.39 Headers/librain\_infiltration/choice\_infiltration.hpp File Reference

Choice of infiltration law.

```
#include "infiltration.hpp"  
#include "greenampt.hpp"  
#include "no_infiltration.hpp"
```

### Classes

- class [Choice\\_infiltration](#)  
*Choice of infiltration law.*

### Macros

- #define [CHOICE\\_INFILTRATION\\_HPP](#)

#### 6.39.1 Detailed Description

Choice of infiltration law.

Author

Marie Rousseau [M.Rousseau@brgm.fr](mailto:M.Rousseau@brgm.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter: calls the chosen infiltration law.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

#### 6.39.2 Macro Definition Documentation

**#define CHOICE\_INFILTRATION\_HPP**

Definition at line 72 of file choice\_infiltration.hpp.

## 6.40 Headers/librain\_infiltration/choice\_rain.hpp File Reference

Choice of initialization for the rain.

```
#include "rain.hpp"  
#include "rain_read.hpp"  
#include "rain_generated.hpp"  
#include "no_rain.hpp"
```

## Classes

- class [Choice\\_rain](#)

*Choice of initialization for the rain.*

## Macros

- #define [CHOICE\\_RAIN\\_HPP](#)

### 6.40.1 Detailed Description

Choice of initialization for the rain.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the rain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

### 6.40.2 Macro Definition Documentation

**#define CHOICE\_RAIN\_HPP**

Definition at line 75 of file choice\_rain.hpp.

## 6.41 Headers/librain\_infiltration/greenampt.hpp File Reference

Green-Ampt law.

```
#include "infiltration.hpp"
```

## Classes

- class [GreenAmpt](#)

*Green-Ampt law.*

### 6.41.1 Detailed Description

Green-Ampt law.

Author

Marie Rousseau [M.Rousseau@brgm.fr](mailto:M.Rousseau@brgm.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration law: bi-layer Green-Ampt.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.42 Headers/librain\_infiltration/infiltration.hpp File Reference

Infiltration law

```
#include "parameters.hpp"
```

### Classes

- class [Infiltration](#)

*Definition of infiltration law.*

### 6.42.1 Detailed Description

Infiltration law

Author

Marie Rousseau [M.Rousseau@brgm.fr](mailto:M.Rousseau@brgm.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for the infiltration.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.43 Headers/librain\_infiltration/no\_infiltration.hpp File Reference

No infiltration.

```
#include "infiltration.hpp"
```

### Classes

- class [No\\_Infiltration](#)  
*No infiltration.*

#### 6.43.1 Detailed Description

No infiltration.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration: there is no infiltration.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.44 Headers/librain\_infiltration/no\_rain.hpp File Reference

No rain.

```
#include "rain.hpp"
```

### Classes

- class [No\\_Rain](#)  
*No rain.*

#### 6.44.1 Detailed Description

No rain.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Rain: there is no rain.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.45 Headers/librain\_infiltration/rain.hpp File Reference

## Rain

```
#include "parameters.hpp"
```

### Classes

- class [Rain](#)

*Initialization of the rain.*

### 6.45.1 Detailed Description

## Rain

## Author

Marie Rousseau [ma.rousseau@brgm.fr](mailto:ma.rousseau@brgm.fr) (2008)Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Common part for the initialization of the rain.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.46 Headers/librain\_infiltration/rain\_generated.hpp File Reference

Constant rain configuration.

```
#include "rain.hpp"
```

## Classes

- class [Rain\\_generated](#)  
*Constant rain configuration.*

### 6.46.1 Detailed Description

Constant rain configuration.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the value is equals to 0.00001 m/s = 36 mm/h, constant during the simulation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.47 Headers/librain\_infiltration/rain\_read.hpp File Reference

File configuration.

```
#include "rain.hpp"
```

## Classes

- class [Rain\\_read](#)  
*File configuration.*

### 6.47.1 Detailed Description

File configuration.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the values are read in a file.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.48 Headers/libreconstructions/choice\_reconstruction.hpp File Reference

Choice of reconstruction.

```
#include "reconstruction.hpp"
#include "muscl.hpp"
#include "eno.hpp"
#include "eno_mod.hpp"
```

### Classes

- class [Choice\\_reconstruction](#)  
*Choice of reconstruction.*

### Macros

- #define [CHOICE\\_RECONSTRUCTION](#)

#### 6.48.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen reconstruction.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

#### 6.48.2 Macro Definition Documentation

**#define CHOICE\_RECONSTRUCTION**

Definition at line 76 of file choice\_reconstruction.hpp.

## 6.49 Headers/libreconstructions/eno.hpp File Reference

ENO reconstruction

```
#include "reconstruction.hpp"
```



## Classes

- class [ENO](#)  
*ENO reconstruction*

### 6.49.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: ENO.

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.50 Headers/libreconstructions/eno\_mod.hpp File Reference

Modified ENO reconstruction.

```
#include "reconstruction.hpp"
```

## Classes

- class [ENO\\_mod](#)  
*Modified ENO reconstruction.*

### 6.50.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: modified ENO.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.51 Headers/libreconstructions/hydrostatic.hpp File Reference

Hydrostatic reconstruction

```
#include "misc.hpp"
```

### Classes

- class [Hydrostatic](#)

*Hydrostatic reconstruction*

### 6.51.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.52 Headers/libreconstructions/muscl.hpp File Reference

MUSCL reconstruction

```
#include "reconstruction.hpp"
```

### Classes

- class [MUSCL](#)

*MUSCL reconstruction*

### 6.52.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: MUSCL.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.53 Headers/libreconstructions/reconstruction.hpp File Reference

Reconstruction

```
#include "parameters.hpp"  
#include "choice_limiter.hpp"
```

### Classes

- class [Reconstruction](#)

*Reconstruction of the variables*

### 6.53.1 Detailed Description

Reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the reconstructions.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.54 Headers/libsave/choice\_output.hpp File Reference

Choice of output format.

```
#include "output.hpp"  
#include "gnuplot.hpp"  
#include "vtk_out.hpp"  
#include "no_evolution_file.hpp"
```

### Classes

- class [Choice\\_output](#)  
*Choice of output format.*

### Macros

- #define [CHOICE\\_OUTPUT\\_HPP](#)

#### 6.54.1 Detailed Description

Choice of output format.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the savings in the chosen format.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

#### 6.54.2 Macro Definition Documentation

**#define CHOICE\_OUTPUT\_HPP**

Definition at line 74 of file choice\_output.hpp.

## 6.55 Headers/libsave/gnuplot.hpp File Reference

Gnuplot output

```
#include "output.hpp"
```

## Classes

- class [Gnuplot](#)  
*Gnuplot output*

### 6.55.1 Detailed Description

Gnuplot output

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Output format: optimized for Gnuplot.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.56 Headers/libsave/no\_evolution\_file.hpp File Reference

No output.

```
#include "output.hpp"
```

## Classes

- class [No\\_Evolution\\_File](#)  
*No output.*

### 6.56.1 Detailed Description

No output.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

No output files with time evolution

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.57 Headers/libsave/output.hpp File Reference

Output format

```
#include "parameters.hpp"
```

### Classes

- class [Output](#)  
*Output format*

### 6.57.1 Detailed Description

Output format

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the output formats.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.58 Headers/libsave/vtk\_out.hpp File Reference

VTK output

```
#include "output.hpp"
```

### Classes

- class [Vtk\\_Out](#)  
*VTK output.*

### 6.58.1 Detailed Description

VTK output

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Output format: optimized for software compatible with vtk format (example: paraview).

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.59 Headers/libschemas/choice\_scheme.hpp File Reference

Choice of numerical scheme.

```
#include "scheme.hpp"  
#include "order1.hpp"  
#include "order2.hpp"
```

### Classes

- class [Choice\\_scheme](#)  
*Choice of numerical scheme.*

### Macros

- #define [CHOICE\\_SCHEME\\_HPP](#)

#### 6.59.1 Detailed Description

Choice of numerical scheme.

## Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

From the value of the corresponding parameter, calls the chosen numerical scheme.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

### 6.59.2 Macro Definition Documentation

**#define CHOICE\_SCHEME\_HPP**

Definition at line 72 of file choice\_scheme.hpp.

## 6.60 Headers/libschemas/order1.hpp File Reference

Order 1 scheme.

```
#include "scheme.hpp"
```

### Classes

- class [Order1](#)

*Order 1 scheme.*

### 6.60.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical scheme: at order 1.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.61 Headers/libschemas/order2.hpp File Reference

Order 2 scheme.

```
#include "scheme.hpp"
```

### Classes

- class [Order2](#)

*Order 2 scheme.*



### 6.61.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical scheme: at order 2.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.62 Headers/libschemas/scheme.hpp File Reference

Numerical scheme.

```
#include "parameters.hpp"
#include "hydrostatic.hpp"
#include "choice_condition.hpp"
#include "choice_flux.hpp"
#include "choice_friction.hpp"
#include "choice_infiltration.hpp"
#include "choice_init_topo.hpp"
#include "choice_init_huv.hpp"
#include "choice_rain.hpp"
#include "choice_output.hpp"
#include "choice_reconstruction.hpp"
```

### Classes

- class [Scheme](#)

*Numerical scheme.*

### 6.62.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Common part for all the numerical schemes.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.63 Sources/FullSWOF\_2D.cpp File Reference

Main function.

```
#include "choice_scheme.hpp"
```

### Functions

- int [main](#) (int argc, char \*\*argv)

#### 6.63.1 Detailed Description

Main function.

## Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

## Version

1.06.00

## Date

2015-02-19

Runs the programm.

## Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

### 6.63.2 Function Documentation

```
int main ( int argc, char ** argv )
```

Main function

Declare the scheme and executes the program.

Returns

0 if the program finished correctly.

Note

The name of the input file (Inputs/parameters.txt) is written here.

Definition at line 58 of file FullSWOF\_2D.cpp.

## 6.64 Sources/libboundaryconditions/bc\_imp\_discharge.cpp File Reference

Imposed discharge.

```
#include "bc_imp_discharge.hpp"
```

### 6.64.1 Detailed Description

Imposed discharge.

Author

Ulrich Razafison [ulrich.razafison@math.cnrs.fr](mailto:ulrich.razafison@math.cnrs.fr) (2011)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: imposed discharge (and water height if necessary).

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.65 Sources/libboundaryconditions/bc\_imp\_height.cpp File Reference

Imposed water height.

```
#include "bc_imp_height.hpp"
```

### 6.65.1 Detailed Description

Imposed water height.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2010-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: imposed water height (and discharge if necessary), based on the modified method of characteristics and Riemann invariants.

See also

Olivier Delestre Ph.D thesis Annexe A [Delestre \[2010\] http://tel.archives-ouvertes.fr/tel-00587197](http://tel.archives-ouvertes.fr/tel-00587197)

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.66 Sources/libboundaryconditions/bc\_neumann.cpp File Reference

Neumann condition.

```
#include "bc_neumann.hpp"
```

### 6.66.1 Detailed Description

Neumann condition.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: Neumann condition (the normal derivative is null).

Copyright

License Cecill-V2  
[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.67 Sources/libboundaryconditions/bc\_periodic.cpp File Reference

Periodic condition.

```
#include "bc_periodic.hpp"
```

### 6.67.1 Detailed Description

Periodic condition.

Author

Pierre-Antoine Ksinant [pierreantoine.ksinantgarcia@gmail.com](mailto:pierreantoine.ksinantgarcia@gmail.com) (2010)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: periodic condition.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.68 Sources/libboundaryconditions/bc\_wall.cpp File Reference

Wall condition.

```
#include "bc_wall.hpp"
```

### 6.68.1 Detailed Description

Wall condition.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)  
Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Boundary condition: wall condition (the discharge at the boundary is null).

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.69 Sources/libboundaryconditions/boundary\_condition.cpp File Reference

Boundary condition.

```
#include "boundary_condition.hpp"
```

### 6.69.1 Detailed Description

Boundary condition.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the boundary conditions.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.70 Sources/libboundaryconditions/choice\_condition.cpp File Reference

Choice of boundary condition.

```
#include "choice_condition.hpp"
```

### 6.70.1 Detailed Description

Choice of boundary condition.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen boundary condition.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.71 Sources/libflux/choice\_flux.cpp File Reference

Choice of numerical flux.

```
#include "choice_flux.hpp"
```

### 6.71.1 Detailed Description

Choice of numerical flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen numerical flux.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.72 Sources/libflux/f\_hll.cpp File Reference

HLL flux.

```
#include "f_hll.hpp"
```

### 6.72.1 Detailed Description

HLL flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical flux: Harten, Lax, van Leer formulation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.73 Sources/libflux/f\_hll2.cpp File Reference

HLL flux.

```
#include "f_hll2.hpp"
```

### 6.73.1 Detailed Description

HLL flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical flux: Harten, Lax, van Leer reduced formulation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.74 Sources/libflux/f\_rusanov.cpp File Reference

Rusanov flux.

```
#include "f_rusanov.hpp"
```

### 6.74.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical flux: Rusanov formulation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)



## 6.75 Sources/libflux/flux.cpp File Reference

Numerical flux.

```
#include "flux.hpp"
```

### 6.75.1 Detailed Description

Numerical flux.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the numerical fluxes.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.76 Sources/libfrictions/choice\_friction.cpp File Reference

Choice of friction law.

```
#include "choice_friction.hpp"
```

### 6.76.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen friction law.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.77 Sources/libfrictions/fr\_darcy\_weisbach.cpp File Reference

Darcy-Weisbach law.

```
#include "fr_darcy_weisbach.hpp"
```

### 6.77.1 Detailed Description

Darcy-Weisbach law.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Darcy-Weisbach.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.78 Sources/libfrictions/fr\_laminar.cpp File Reference

Laminar law.

```
#include "fr_laminar.hpp"
```

### 6.78.1 Detailed Description

Laminar law.

Author

Carine Lucas [carine.lucas@univ-orleans.fr](mailto:carine.lucas@univ-orleans.fr) (2014-2015)

Version

1.06.00

Date

2015-02-19

Friction law: laminar.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - INRA (France)

## 6.79 Sources/libfrictions/fr\_manning.cpp File Reference

Manning law.

```
#include "fr_manning.hpp"
```

### 6.79.1 Detailed Description

Manning law.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Friction law: Manning.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.80 Sources/libfrictions/friction.cpp File Reference

Friction law

```
#include "friction.hpp"
```

### 6.80.1 Detailed Description

Friction law

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the friction laws.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.81 Sources/libfrictions/no\_friction.cpp File Reference

No friction.

```
#include "no_friction.hpp"
```

### 6.81.1 Detailed Description

No friction.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

**Friction** law: does no computation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.82 Sources/libinitializations/choice\_init\_huv.cpp File Reference

Choice of initialization for h, u and v.

```
#include "choice_init_huv.hpp"
```

### 6.82.1 Detailed Description

Choice of initialization for h, u and v.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the water height and of the velocity.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.83 Sources/libinitializations/choice\_init\_topo.cpp File Reference

Choice of initialization for the topography.

```
#include "choice_init_topo.hpp"
```

### 6.83.1 Detailed Description

Choice of initialization for the topography.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the topography.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.84 Sources/libinitializations/huv\_generated.cpp File Reference

No water configuration.

```
#include "huv_generated.hpp"
```

### 6.84.1 Detailed Description

No water configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a dry domain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.85 Sources/libinitializations/huv\_generated\_radial\_dam\_dry.cpp File Reference

Dry radial dam break configuration.

```
#include "huv_generated_radial_dam_dry.hpp"
```

### 6.85.1 Detailed Description

Dry radial dam break configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a radial dam break on a dry domain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.86 Sources/libinitializations/huv\_generated\_radial\_dam\_wet.cpp File Reference

Wet radial dam break configuration.

```
#include "huv_generated_radial_dam_wet.hpp"
```

### 6.86.1 Detailed Description

Wet radial dam break configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of a radial dam break on a wet domain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.87 Sources/libinitializations/huv\_generated\_thacker.cpp File Reference

Thacker configuration.

```
#include "huv_generated_thacker.hpp"
```

### 6.87.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: case of Thacker's benchmark.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.88 Sources/libinitializations/huv\_read.cpp File Reference

File configuration.

```
#include "huv_read.hpp"
```

### 6.88.1 Detailed Description

File configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the water height and of the velocity: the values are read in a file.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.89 Sources/libinitializations/initialization\_huv.cpp File Reference

Initialization of h, u and v

```
#include "initialization_huv.hpp"
```

### 6.89.1 Detailed Description

Initialization of h, u and v

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the initialization of the water height and of the velocity.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.90 Sources/libinitializations/initialization\_topo.cpp File Reference

Initialization of z

```
#include "initialization_topo.hpp"
```



### 6.90.1 Detailed Description

Initialization of z

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the initialization of the topography.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.91 Sources/libinitializations/topo\_generated\_flat.cpp File Reference

Flat configuration.

```
#include "topo_generated_flat.hpp"
```

### 6.91.1 Detailed Description

Flat configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: the topography is flat, its value is 0.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.92 Sources/libinitializations/topo\_generated\_thacker.cpp File Reference

Thacker configuration.

```
#include "topo_generated_thacker.hpp"
```

### 6.92.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: topography with a shape of a paraboloid of revolution for Thacker's Benchmark.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.93 Sources/libinitializations/topo\_read.cpp File Reference

File configuration.

```
#include "topo_read.hpp"
```

### 6.93.1 Detailed Description

File configuration.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the topography: the values are read in a file.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.94 Sources/liblimitations/choice\_limiter.cpp File Reference

Choice of slope limiter.

```
#include "choice_limiter.hpp"
```

### 6.94.1 Detailed Description

Choice of slope limiter.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen slope limiter.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.95 Sources/liblimitations/limiter.cpp File Reference

Slope limiter.

```
#include "limiter.hpp"
```

### 6.95.1 Detailed Description

Slope limiter.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the slope limiters.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.96 Sources/liblimitations/minmod.cpp File Reference

Minmod limiter

```
#include "minmod.hpp"
```

### 6.96.1 Detailed Description

Minmod limiter

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: minmod.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.97 Sources/liblimitations/vanalbada.cpp File Reference

Van Albada limiter.

```
#include "vanalbada.hpp"
```

### 6.97.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: Van Albada.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.98 Sources/liblimitations/vanleer.cpp File Reference

Van Leer limiter.

```
#include "vanleer.hpp"
```

### 6.98.1 Detailed Description

Van Leer limiter.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Slope limiter: Van Leer.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.99 Sources/libparameters/parameters.cpp File Reference

Gets parameters.

```
#include "parameters.hpp"
```

### 6.99.1 Detailed Description

Gets parameters.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2011-2015)

Frederic Darboux [frederic.darboux@orleans.inra.fr](mailto:frederic.darboux@orleans.inra.fr) (2014)

Version

1.06.00

Date

2015-02-19

Reads the parameters, checks their values.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.100 Sources/libparser/parser.cpp File Reference

Parser

```
#include "parser.hpp"
```

### 6.100.1 Detailed Description

Parser

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2010-2015)

Version

1.06.00

Date

2015-02-19

Reads the input file.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - INRA (France)

## 6.101 Sources/librain\_infiltration/choice\_infiltration.cpp File Reference

Choice of infiltration law.

```
#include "choice_infiltration.hpp"
```

### 6.101.1 Detailed Description

Choice of infiltration law.

Author

Marie Rousseau [M.Rousseau@brgm.fr](mailto:M.Rousseau@brgm.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter: calls the chosen infiltration law.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.102 Sources/librain\_infiltration/choice\_rain.cpp File Reference

Choice of initialization for the rain.

```
#include "choice_rain.hpp"
```

### 6.102.1 Detailed Description

Choice of initialization for the rain.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen initialization of the rain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.103 Sources/librain\_infiltration/greenampt.cpp File Reference

Green-Ampt law.

```
#include "greenampt.hpp"
```

### 6.103.1 Detailed Description

Green-Ampt law.

Author

Marie Rousseau [M.Rousseau@brgm.fr](mailto:M.Rousseau@brgm.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration law: bi-layer Green-Ampt.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.104 Sources/librain\_infiltration/infiltration.cpp File Reference

Infiltration law

```
#include "infiltration.hpp"
```

### 6.104.1 Detailed Description

Infiltration law

Author

Marie Rousseau [M.Rousseau@brgm.fr](mailto:M.Rousseau@brgm.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for the infiltration.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.105 Sources/librain\_infiltration/no\_infiltration.cpp File Reference

No infiltration.

```
#include "no_infiltration.hpp"
```

### 6.105.1 Detailed Description

No infiltration.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Infiltration: there is no infiltration.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)



## 6.106 Sources/librain\_infiltration/no\_rain.cpp File Reference

No rain.

```
#include "no_rain.hpp"
```

### 6.106.1 Detailed Description

No rain.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Rain: there is no rain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.107 Sources/librain\_infiltration/rain.cpp File Reference

Rain

```
#include "rain.hpp"
```

### 6.107.1 Detailed Description

Rain

Author

Marie Rousseau [ma.rousseau@brgm.fr](mailto:ma.rousseau@brgm.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for the initialization of the rain.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.108 Sources/librain\_infiltration/rain\_generated.cpp File Reference

Constant rain configuration.

```
#include "rain_generated.hpp"
```

### 6.108.1 Detailed Description

Constant rain configuration.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the value is equals to 0.00001 m/s = 36 mm/h, constant during the simulation.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.109 Sources/librain\_infiltration/rain\_read.cpp File Reference

File configuration.

```
#include "rain_read.hpp"
```

### 6.109.1 Detailed Description

File configuration.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Initialization of the rain: the values are read in a file.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.110 Sources/libreconstructions/choice\_reconstruction.cpp File Reference

Choice of reconstruction.

```
#include "choice_reconstruction.hpp"
```

### 6.110.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen reconstruction.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.111 Sources/libreconstructions/eno.cpp File Reference

ENO reconstruction

```
#include "eno.hpp"
```

### 6.111.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: ENO.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.112 Sources/libreconstructions/eno\_mod.cpp File Reference

Modified ENO reconstruction.

```
#include "eno_mod.hpp"
```

### 6.112.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: modified ENO.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.113 Sources/libreconstructions/hydrostatic.cpp File Reference

Hydrostatic reconstruction

```
#include "hydrostatic.hpp"
```

### 6.113.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.114 Sources/libreconstructions/muscl.cpp File Reference

MUSCL reconstruction

```
#include "muscl.hpp"
```

### 6.114.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Linear reconstruction: MUSCL.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.115 Sources/libreconstructions/reconstruction.cpp File Reference

Reconstruction

```
#include "reconstruction.hpp"
```

### 6.115.1 Detailed Description

Reconstruction

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the reconstructions.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.116 Sources/libsave/choice\_output.cpp File Reference

Choice of output format.

```
#include "choice_output.hpp"
```

### 6.116.1 Detailed Description

Choice of output format.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the savings in the chosen format.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.117 Sources/libsave/gnuplot.cpp File Reference

Gnuplot output

```
#include "gnuplot.hpp"
```

### 6.117.1 Detailed Description

Gnuplot output

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Output format: optimized for Gnuplot (for huz\_evolution.dat).

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.118 Sources/libsave/no\_evolution\_file.cpp File Reference

No output.

```
#include "no_evolution_file.hpp"
```

### 6.118.1 Detailed Description

No output.

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

No output files with time evolution

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.119 Sources/libsave/output.cpp File Reference

Output format

```
#include "output.hpp"
```

### 6.119.1 Detailed Description

Output format

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the output formats.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.120 Sources/libsave/vtk\_out.cpp File Reference

VTK output

```
#include "vtk_out.hpp"
```

### 6.120.1 Detailed Description

VTK output

Author

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Output format: optimized for software compatible with vtk format (example: paraview).

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.121 Sources/libschemas/choice\_scheme.cpp File Reference

Choice of numerical scheme.

```
#include "choice_scheme.hpp"
```

### 6.121.1 Detailed Description

Choice of numerical scheme.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

From the value of the corresponding parameter, calls the chosen numerical scheme.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)



## 6.122 Sources/libschemas/order1.cpp File Reference

Order 1 scheme.

```
#include "order1.hpp"
```

### 6.122.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical scheme: at order 1.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.123 Sources/libschemas/order2.cpp File Reference

Order 2 scheme.

```
#include "order2.hpp"
```

### 6.123.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Numerical scheme: at order 2.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

## 6.124 Sources/libschemas/scheme.cpp File Reference

Numerical scheme.

```
#include "scheme.hpp"
```

### 6.124.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre [olivierdelestre41@yahoo.fr](mailto:olivierdelestre41@yahoo.fr) (2008)

Christian Laguerre [christian.laguerre@math.cnrs.fr](mailto:christian.laguerre@math.cnrs.fr) (2012-2015)

Version

1.06.00

Date

2015-02-19

Common part for all the numerical schemes.

Copyright

License Cecill-V2

[http://www.cecill.info/licences/Licence\\_CeCILL\\_V2-en.html](http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)

(c) CNRS - Universite d'Orleans - BRGM (France)

# Bibliography

- E. Audusse, M.-O. Bristeau, and B. Perthame. Kinetic schemes for Saint-Venant equations with source terms on unstructured grids. Technical Report 3989, INRIA, 2000. [73](#), [74](#)
- E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM J. Sci. Comput.*, 25(6):2050–2065, 2004. doi:[10.1137/S1064827503431090](#). [78](#)
- F. Bouchut. *Nonlinear stability of finite volume methods for hyperbolic conservation laws, and well-balanced schemes for sources*, volume 2/2004. Birkhäuser Basel, 2004. doi:[10.1007/b95203](#). [48](#), [50](#), [52](#), [55](#)
- F. Bouchut. Chapter 4 efficient numerical finite volume schemes for shallow water models. In V. Zeitlin, editor, *Nonlinear Dynamics of Rotating Shallow Water: Methods and Advances*, volume 2 of *Edited Series on Advances in Nonlinear Science and Complexity*, pages 189 – 256. Elsevier Science, 2007. doi:[10.1016/S1574-6909\(06\)02004-1](#). URL <http://www.sciencedirect.com/science/article/B8JG3-4PS6TNS-6/2/f4c3dbcf476626c1cb6f353e9bc66cc9>. [48](#), [50](#), [88](#)
- M.-O. Bristeau and B. Coussin. Boundary conditions for the shallow water equations solved by kinetic schemes. Technical Report RR-4282, INRIA, 2001. [1](#), [26](#)
- O. Delestre. *Simulation du ruissellement d'eau de pluie sur des surfaces agricoles*. PhD thesis, Université d'Orléans, Orléans, France, July 2010. <http://tel.archives-ouvertes.fr/tel-00531377/en/>. [156](#), [202](#)
- N. Goutal and F. Maurel. Proceedings of the 2<sup>nd</sup> workshop on dam-break wave simulation. Technical Report HE-43/97/016/B, Electricité de France, Direction des études et recherches, 1997. [73](#), [74](#)
- A. Harten, S. Osher, B. Engquist, and S. R. Chakravarthy. Some results on uniformly high-order accurate essentially nonoscillatory schemes. *Applied Numerical Mathematics*, 2(3-5):347 – 377, 1986. ISSN 0168-9274. doi:[10.1016/0168-9274\(86\)90039-5](#). URL <http://www.sciencedirect.com/science/article/B6TYD-45GVV8T-J/2/830ca2dce5e3fb34ace9fa53ae5ab76b>. Special Issue in Honor of Milt Rose's Sixtieth Birthday. [48](#)
- A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III. *Journal of Computational Physics*, 71:231–303, 1987. [48](#)
- A. Y. Le Roux. Conditions aux limites et problèmes hyperboliques : un point de vue numérique. Available in the document of the Conférence at IHP-Paris, 2001. [1](#), [26](#)
- C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439 – 471, 1988. ISSN 0021-9991. doi:[10.1016/0021-9991\(88\)90177-5](#). URL <http://www.sciencedirect.com/science/article/B6WHY-4DD1T6W-MM/2/bef99e4b67bd7132c8af1984c34ce57e>. [48](#)
- M. W. Smith, N. J. Cox, and L. J. Bracken. Applying flow resistance equations to overland flows. *Progress in Physical Geography*, 31(4):363–387, 2007. doi:[10.1177/0309133307081289](#). [59](#), [63](#)
- W. C. Thacker. Some exact solutions to the nonlinear shallow-water wave equations. *Journal of Fluid Mechanics*, 107:499–508, 1981. doi:[10.1017/S0022112081001882](#). URL <http://journals.cambridge>.

[org/action/displayAbstract?fromPage=online&aid=389055&fulltextType=RA&fileId=S0022112081001882](http://www.sciencedirect.com/science/article/B6WHY-4DD1N8T-C5/2/9b051d1cfcff715a3d0f4b7b7b0397cc). 76, 148

- B. van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101 – 136, 1979. ISSN 0021-9991. doi:[10.1016/0021-9991\(79\)90145-1](https://doi.org/10.1016/0021-9991(79)90145-1). URL <http://www.sciencedirect.com/science/article/B6WHY-4DD1N8T-C5/2/9b051d1cfcff715a3d0f4b7b7b0397cc>. 88

# Index

- ~Bc\_Neumann
  - Bc\_Neumann, 21
- ~Bc\_imp\_discharge
  - Bc\_imp\_discharge, 17
- ~Bc\_imp\_height
  - Bc\_imp\_height, 20
- ~Bc\_periodic
  - Bc\_periodic, 23
- ~Bc\_wall
  - Bc\_wall, 25
- ~Boundary\_condition
  - Boundary\_condition, 27
- ~Choice\_condition
  - Choice\_condition, 29
- ~Choice\_flux
  - Choice\_flux, 31
- ~Choice\_friction
  - Choice\_friction, 33
- ~Choice\_infiltration
  - Choice\_infiltration, 35
- ~Choice\_init\_huv
  - Choice\_init\_huv, 37
- ~Choice\_init\_topo
  - Choice\_init\_topo, 38
- ~Choice\_limiter
  - Choice\_limiter, 39
- ~Choice\_output
  - Choice\_output, 40
- ~Choice\_rain
  - Choice\_rain, 43
- ~Choice\_reconstruction
  - Choice\_reconstruction, 44
- ~Choice\_scheme
  - Choice\_scheme, 46
- ~ENO
  - ENO, 48
- ~ENO\_mod
  - ENO\_mod, 50
- ~F\_HLL
  - F\_HLL, 51
- ~F\_HLL2
  - F\_HLL2, 53
- ~F\_Rusanov
  - F\_Rusanov, 55
- ~Flux
  - Flux, 56
- ~Fr\_Darcy\_Weisbach
  - Fr\_Darcy\_Weisbach, 59
- ~Fr\_Laminar
  - Fr\_Laminar, 61
- ~Fr\_Manning
  - Fr\_Manning, 63
- ~Friction
  - Friction, 66
- ~Gnuplot
  - Gnuplot, 69
- ~GreenAmpt
  - GreenAmpt, 70
- ~Huv\_generated
  - Huv\_generated, 72
- ~Huv\_generated\_Radial\_Dam\_dry
  - Huv\_generated\_Radial\_Dam\_dry, 73
- ~Huv\_generated\_Radial\_Dam\_wet
  - Huv\_generated\_Radial\_Dam\_wet, 74
- ~Huv\_generated\_Thacker
  - Huv\_generated\_Thacker, 76
- ~Huv\_read
  - Huv\_read, 77
- ~Hydrostatic
  - Hydrostatic, 78
- ~Infiltration
  - Infiltration, 80
- ~Initialization\_huv
  - Initialization\_huv, 82
- ~Initialization\_topo
  - Initialization\_topo, 84
- ~Limiter
  - Limiter, 85
- ~MUSCL
  - MUSCL, 88
- ~Minmod
  - Minmod, 86
- ~No\_Evolution\_File
  - No\_Evolution\_File, 90
- ~No\_Friction
  - No\_Friction, 91
- ~No\_Infiltration
  - No\_Infiltration, 93
- ~No\_Rain
  - No\_Rain, 94
- ~Order1
  - Order1, 95

- ~Order2
  - Order2, 96
- ~Output
  - Output, 98
- ~Parameters
  - Parameters, 107
- ~Parser
  - Parser, 127
- ~Rain
  - Rain, 128
- ~Rain\_generated
  - Rain\_generated, 130
- ~Rain\_read
  - Rain\_read, 131
- ~Reconstruction
  - Reconstruction, 132
- ~Scheme
  - Scheme, 136
- ~Topo\_generated\_Thacker
  - Topo\_generated\_Thacker, 148
- ~Topo\_generated\_flat
  - Topo\_generated\_flat, 147
- ~Topo\_read
  - Topo\_read, 149
- ~VanAlbada
  - VanAlbada, 150
- ~VanLeer
  - VanLeer, 152
- ~Vtk\_Out
  - Vtk\_Out, 153
- allocation
  - Scheme, 136
- amortENO
  - Parameters, 119
- B\_IMP\_H
  - Scheme, 138
- B\_IMP\_Q
  - Scheme, 138
- BC\_IMP\_DISCHARGE\_HPP
  - bc\_imp\_discharge.hpp, 156
- Bbound
  - Parameters, 119
  - Scheme, 138
- Bc\_Neumann, 20
  - ~Bc\_Neumann, 21
  - Bc\_Neumann, 21
  - calcul, 21
- Bc\_imp\_discharge, 15
  - ~Bc\_imp\_discharge, 17
  - Bc\_imp\_discharge, 16
  - calcul, 17
  - getValueOfPolynomial, 18
  - getValueofDerivativeOfPolynomial, 17
  - newtonSolver, 18
- bc\_imp\_discharge.hpp
  - BC\_IMP\_DISCHARGE\_HPP, 156
- Bc\_imp\_height, 19
  - ~Bc\_imp\_height, 20
  - Bc\_imp\_height, 19
  - calcul, 20
- Bc\_periodic, 22
  - ~Bc\_periodic, 23
  - Bc\_periodic, 23
  - calcul, 23
- Bc\_wall, 24
  - ~Bc\_wall, 25
  - Bc\_wall, 24
  - calcul, 25
- bottom\_imp\_discharge
  - Parameters, 120
- bottom\_imp\_h
  - Parameters, 120
- boundaries\_flux
  - Choice\_output, 40
  - Output, 98
- boundaries\_flux\_BT
  - Choice\_output, 40
  - Output, 99
- boundaries\_flux\_LR
  - Choice\_output, 41
  - Output, 99
- boundary
  - Scheme, 137
- Boundary\_condition, 25
  - ~Boundary\_condition, 27
  - Boundary\_condition, 26
  - calcul, 27
  - get\_hbound, 27
  - get\_unormbound, 27
  - get\_utanbound, 27
  - hbound, 27
  - NXCELL, 27
  - NYCELL, 27
  - unormbound, 28
  - unormfix, 28
  - utanbound, 28
- CFL\_FIX
  - Scheme, 139
- CHOICE\_CONDITION\_HPP
  - choice\_condition.hpp, 160
- CHOICE\_FLUX\_HPP
  - choice\_flux.hpp, 161
- CHOICE\_FRICTION\_HPP
  - choice\_friction.hpp, 164
- CHOICE\_INFILTRATION\_HPP

- choice\_infiltration.hpp, 184
- CHOICE\_INIT\_HUV\_HPP
  - choice\_init\_huv.hpp, 168
- CHOICE\_INIT\_TOPO\_HPP
  - choice\_init\_topo.hpp, 169
- CHOICE\_LIMITER\_HPP
  - choice\_limiter.hpp, 177
- CHOICE\_OUTPUT\_HPP
  - choice\_output.hpp, 194
- CHOICE\_RAIN\_HPP
  - choice\_rain.hpp, 185
- CHOICE\_RECONSTRUCTION
  - choice\_reconstruction.hpp, 190
- CHOICE\_SCHEME\_HPP
  - choice\_scheme.hpp, 198
- CONST\_CFL\_X
  - misc.hpp, 181
- CONST\_CFL\_Y
  - misc.hpp, 181
- calcul
  - Bc\_Neumann, 21
  - Bc\_imp\_discharge, 17
  - Bc\_imp\_height, 20
  - Bc\_periodic, 23
  - Bc\_wall, 25
  - Boundary\_condition, 27
  - Choice\_condition, 29
  - Choice\_flux, 31
  - Choice\_friction, 33
  - Choice\_infiltration, 35
  - Choice\_limiter, 39
  - Choice\_reconstruction, 44
  - Choice\_scheme, 46
  - ENO, 48
  - ENO\_mod, 50
  - F\_HLL, 52
  - F\_HLL2, 53
  - F\_Rusanov, 55
  - Flux, 57
  - Fr\_Darcy\_Weisbach, 59
  - Fr\_Laminar, 61
  - Fr\_Manning, 63
  - Friction, 66
  - GreenAmpt, 70
  - Hydrostatic, 78
  - Infiltration, 80
  - Limiter, 85
  - MUSCL, 88
  - Minmod, 87
  - No\_Friction, 91
  - No\_Infiltration, 93
  - Order1, 95
  - Order2, 96
  - Reconstruction, 132
  - Scheme, 137
  - VanAlbada, 151
  - VanLeer, 152
- calculSf
  - Choice\_friction, 33
  - Fr\_Darcy\_Weisbach, 60
  - Fr\_Laminar, 62
  - Fr\_Manning, 64
  - Friction, 66
  - No\_Friction, 91
- capacity
  - GreenAmpt, 70
- cfl
  - Flux, 58
- cfl\_fix
  - Parameters, 120
- check\_vol
  - Choice\_output, 41
  - Output, 99
- Choice\_condition, 28
  - ~Choice\_condition, 29
  - calcul, 29
  - Choice\_condition, 29
  - get\_hbound, 29
  - get\_unormbound, 29
  - get\_utanbound, 30
- choice\_condition.hpp
  - CHOICE\_CONDITION\_HPP, 160
- Choice\_flux, 30
  - ~Choice\_flux, 31
  - calcul, 31
  - Choice\_flux, 31
  - get\_cfl, 31
  - get\_f1, 31
  - get\_f2, 31
  - get\_f3, 32
  - set\_tx, 32
- choice\_flux.hpp
  - CHOICE\_FLUX\_HPP, 161
- Choice\_friction, 32
  - ~Choice\_friction, 33
  - calcul, 33
  - calculSf, 33
  - Choice\_friction, 33
  - get\_Sf1, 34
  - get\_Sf2, 34
  - get\_q1mod, 34
  - get\_q2mod, 34
- choice\_friction.hpp
  - CHOICE\_FRICTION\_HPP, 164
- Choice\_infiltration, 35
  - ~Choice\_infiltration, 35

- calcul, [35](#)
- Choice\_infiltration, [35](#)
- get\_Vin, [36](#)
- get\_hmod, [36](#)
- choice\_infiltration.hpp
  - CHOICE\_INFILTRATION\_HPP, [184](#)
- Choice\_init\_huv, [36](#)
  - ~Choice\_init\_huv, [37](#)
  - Choice\_init\_huv, [36](#)
  - initialization, [37](#)
- choice\_init\_huv.hpp
  - CHOICE\_INIT\_HUV\_HPP, [168](#)
- Choice\_init\_topo, [37](#)
  - ~Choice\_init\_topo, [38](#)
  - Choice\_init\_topo, [37](#)
  - initialization, [38](#)
- choice\_init\_topo.hpp
  - CHOICE\_INIT\_TOPO\_HPP, [169](#)
- Choice\_limiter, [38](#)
  - ~Choice\_limiter, [39](#)
  - calcul, [39](#)
  - Choice\_limiter, [38](#)
  - get\_rec, [39](#)
- choice\_limiter.hpp
  - CHOICE\_LIMITER\_HPP, [177](#)
- Choice\_output, [39](#)
  - ~Choice\_output, [40](#)
  - boundaries\_flux, [40](#)
  - boundaries\_flux\_BT, [40](#)
  - boundaries\_flux\_LR, [41](#)
  - check\_vol, [41](#)
  - Choice\_output, [40](#)
  - final, [41](#)
  - initial, [41](#)
  - result, [42](#)
  - write, [42](#)
- choice\_output.hpp
  - CHOICE\_OUTPUT\_HPP, [194](#)
- Choice\_rain, [42](#)
  - ~Choice\_rain, [43](#)
  - Choice\_rain, [43](#)
  - rain\_func, [43](#)
- choice\_rain.hpp
  - CHOICE\_RAIN\_HPP, [185](#)
- Choice\_reconstruction, [43](#)
  - ~Choice\_reconstruction, [44](#)
  - calcul, [44](#)
  - Choice\_reconstruction, [44](#)
- choice\_reconstruction.hpp
  - CHOICE\_RECONSTRUCTION, [190](#)
- Choice\_scheme, [45](#)
  - ~Choice\_scheme, [46](#)
  - calcul, [46](#)
  - Choice\_scheme, [45](#)
- choice\_scheme.hpp
  - CHOICE\_SCHEME\_HPP, [198](#)
- cpu\_time
  - Scheme, [139](#)
- cur\_time
  - Scheme, [139](#)
- DT\_FIX
  - Scheme, [139](#)
- DX
  - Friction, [67](#)
  - Infiltration, [81](#)
  - Initialization\_huv, [82](#)
  - Initialization\_topo, [84](#)
  - Output, [101](#)
  - Rain, [128](#)
  - Scheme, [140](#)
- DY
  - Friction, [67](#)
  - Infiltration, [81](#)
  - Initialization\_huv, [82](#)
  - Initialization\_topo, [84](#)
  - Output, [101](#)
  - Rain, [128](#)
  - Scheme, [140](#)
- deallocation
  - Scheme, [137](#)
- delta\_z1
  - Reconstruction, [133](#)
- delta\_z2
  - Reconstruction, [133](#)
- delz1
  - Scheme, [139](#)
- delz2
  - Scheme, [139](#)
- delzc1
  - Scheme, [139](#)
- delzc2
  - Scheme, [139](#)
- dt1
  - Scheme, [139](#)
- dt\_first
  - Scheme, [139](#)
- dt\_fix
  - Parameters, [120](#)
- dt\_max
  - Scheme, [139](#)
- dt\_output
  - Scheme, [140](#)
- dtheta\_NF
  - Parameters, [120](#)
- dtheta\_coef
  - Parameters, [120](#)



- dtheta\_init
  - Parameters, 120
- dtheta\_namefile
  - Parameters, 120
- dx
  - Parameters, 120
- dy
  - Parameters, 120
- ENO, 46
  - ~ENO, 48
  - calcul, 48
  - ENO, 47
- ENO\_mod, 49
  - ~ENO\_mod, 50
  - calcul, 50
  - ENO\_mod, 49
- EPSILON
  - misc.hpp, 181
- end
  - Scheme, 140
- f1
  - Flux, 58
  - Scheme, 140
- f2
  - Flux, 58
  - Scheme, 140
- f3
  - Flux, 58
  - Scheme, 140
- F\_HLL, 51
  - ~F\_HLL, 51
  - calcul, 52
  - F\_HLL, 51
- F\_HLL2, 52
  - ~F\_HLL2, 53
  - calcul, 53
  - F\_HLL2, 53
- F\_Rusanov, 54
  - ~F\_Rusanov, 55
  - calcul, 55
  - F\_Rusanov, 55
- FRICCOEF
  - Scheme, 140
- fill\_array
  - Parameters, 107
- final
  - Choice\_output, 41
  - Output, 99
- Flux, 56
  - ~Flux, 56
  - calcul, 57
  - cfl, 58
  - f1, 58
  - f2, 58
  - f3, 58
  - Flux, 56
  - get\_cfl, 57
  - get\_f1, 57
  - get\_f2, 57
  - get\_f3, 57
  - set\_tx, 57
  - tx, 58
- flux
  - Parameters, 120
- Fr
  - Scheme, 140
- Fr\_Darcy\_Weisbach, 58
  - ~Fr\_Darcy\_Weisbach, 59
  - calcul, 59
  - calculSf, 60
  - Fr\_Darcy\_Weisbach, 59
- Fr\_Laminar, 60
  - ~Fr\_Laminar, 61
  - calcul, 61
  - calculSf, 62
  - Fr\_Laminar, 61
- Fr\_Manning, 62
  - ~Fr\_Manning, 63
  - calcul, 63
  - calculSf, 64
  - Fr\_Manning, 63
- fric
  - Parameters, 121
- fric\_NF
  - Parameters, 121
- fric\_init
  - Parameters, 121
- fric\_namefile
  - Parameters, 121
- Fric\_tab
  - Friction, 67
- friccoef
  - Parameters, 121
- Friction, 65
  - ~Friction, 66
  - calcul, 66
  - calculSf, 66
  - DX, 67
  - DY, 67
  - Fric\_tab, 67
  - Friction, 66
  - get\_Sf1, 66
  - get\_Sf2, 67
  - get\_q1mod, 66
  - get\_q2mod, 66

- NXCELL, [67](#)
- NYCELL, [67](#)
- q1mod, [67](#)
- q2mod, [67](#)
- Sf1, [67](#)
- Sf2, [68](#)
- froude\_number
  - Scheme, [137](#)
- FullSWOF\_2D.cpp
  - main, [201](#)
- g1
  - Scheme, [140](#)
- g2
  - Scheme, [140](#)
- g3
  - Scheme, [141](#)
- GRAV
  - misc.hpp, [181](#)
- GRAV\_DEM
  - misc.hpp, [181](#)
- get\_Bbound
  - Parameters, [108](#)
- get\_Kc\_coef
  - Parameters, [112](#)
- get\_Kc\_init
  - Parameters, [112](#)
- get\_KcNameFile
  - Parameters, [112](#)
- get\_KcNameFileS
  - Parameters, [112](#)
- get\_Ks\_coef
  - Parameters, [112](#)
- get\_Ks\_init
  - Parameters, [113](#)
- get\_KsNameFile
  - Parameters, [113](#)
- get\_KsNameFileS
  - Parameters, [113](#)
- get\_Lbound
  - Parameters, [113](#)
- get\_Nxcell
  - Parameters, [114](#)
- get\_Nycell
  - Parameters, [114](#)
- get\_Psi\_coef
  - Parameters, [115](#)
- get\_Psi\_init
  - Parameters, [115](#)
- get\_PsiNameFile
  - Parameters, [115](#)
- get\_PsiNameFileS
  - Parameters, [115](#)
- get\_Rbound
  - Parameters, [116](#)
- get\_Sf1
  - Choice\_friction, [34](#)
  - Friction, [66](#)
- get\_Sf2
  - Choice\_friction, [34](#)
  - Friction, [67](#)
- get\_T
  - Parameters, [117](#)
- get\_Tbound
  - Parameters, [117](#)
- get\_Vin
  - Choice\_infiltration, [36](#)
  - Infiltration, [80](#)
- get\_amortENO
  - Parameters, [107](#)
- get\_bottom\_imp\_discharge
  - Parameters, [108](#)
- get\_bottom\_imp\_h
  - Parameters, [108](#)
- get\_cfl
  - Choice\_flux, [31](#)
  - Flux, [57](#)
- get\_cflfix
  - Parameters, [108](#)
- get\_dtfix
  - Parameters, [108](#)
- get\_dtheta\_coef
  - Parameters, [108](#)
- get\_dtheta\_init
  - Parameters, [109](#)
- get\_dthetaNameFile
  - Parameters, [109](#)
- get\_dthetaNameFileS
  - Parameters, [109](#)
- get\_dx
  - Parameters, [109](#)
- get\_dy
  - Parameters, [109](#)
- get\_f1
  - Choice\_flux, [31](#)
  - Flux, [57](#)
- get\_f2
  - Choice\_flux, [31](#)
  - Flux, [57](#)
- get\_f3
  - Choice\_flux, [32](#)
  - Flux, [57](#)
- get\_flux
  - Parameters, [109](#)
- get\_fric
  - Parameters, [110](#)
- get\_fric\_init

- Parameters, 110
- get\_friccoef
  - Parameters, 110
- get\_frictionNameFile
  - Parameters, 110
- get\_frictionNameFileS
  - Parameters, 110
- get\_hbound
  - Boundary\_condition, 27
  - Choice\_condition, 29
- get\_hhydro\_l
  - Hydrostatic, 78
- get\_hhydro\_r
  - Hydrostatic, 79
- get\_hmod
  - Choice\_infiltration, 36
  - Infiltration, 80
- get\_huv
  - Parameters, 110
- get\_huvNameFile
  - Parameters, 111
- get\_huvNameFileS
  - Parameters, 111
- get\_imax\_coef
  - Parameters, 111
- get\_imax\_init
  - Parameters, 111
- get\_imaxNameFile
  - Parameters, 111
- get\_imaxNameFileS
  - Parameters, 111
- get\_inf
  - Parameters, 112
- get\_left\_imp\_discharge
  - Parameters, 113
- get\_left\_imp\_h
  - Parameters, 113
- get\_lim
  - Parameters, 114
- get\_modifENO
  - Parameters, 114
- get\_nbtimes
  - Parameters, 114
- get\_order
  - Parameters, 114
- get\_output
  - Parameters, 115
- get\_outputDirectory
  - Parameters, 115
- get\_q1mod
  - Choice\_friction, 34
  - Friction, 66
- get\_q2mod
  - Choice\_friction, 34
  - Friction, 66
- get\_rain
  - Parameters, 116
- get\_rainNameFile
  - Parameters, 116
- get\_rainNameFileS
  - Parameters, 116
- get\_rec
  - Choice\_limiter, 39
  - Limiter, 85
  - Parameters, 116
- get\_right\_imp\_discharge
  - Parameters, 116
- get\_right\_imp\_h
  - Parameters, 117
- get\_scheme\_type
  - Parameters, 117
- get\_suffix
  - Parameters, 117
- get\_top\_imp\_discharge
  - Parameters, 117
- get\_top\_imp\_h
  - Parameters, 118
- get\_topo
  - Parameters, 118
- get\_topographyNameFile
  - Parameters, 118
- get\_topographyNameFileS
  - Parameters, 118
- get\_unormbound
  - Boundary\_condition, 27
  - Choice\_condition, 29
- get\_utanbound
  - Boundary\_condition, 27
  - Choice\_condition, 30
- get\_zcrust\_coef
  - Parameters, 118
- get\_zcrust\_init
  - Parameters, 118
- get\_zcrustNameFile
  - Parameters, 119
- get\_zcrustNameFileS
  - Parameters, 119
- getValue
  - Parser, 127
- getValueOfPolynomial
  - Bc\_imp\_discharge, 18
- getValueofDerivativeOfPolynomial
  - Bc\_imp\_discharge, 17
- Gnuplot, 68
  - ~Gnuplot, 69
  - Gnuplot, 68

- write, 69
- GreenAmpt, 69
  - ~GreenAmpt, 70
  - calcul, 70
  - capacity, 70
  - GreenAmpt, 70
- h
  - Scheme, 141
- h1l
  - Scheme, 141
- h1left
  - Scheme, 141
- h1r
  - Scheme, 141
- h1right
  - Scheme, 141
- h2l
  - Scheme, 141
- h2left
  - Scheme, 141
- h2r
  - Scheme, 141
- h2right
  - Scheme, 141
- HE\_CA
  - misc.hpp, 181
- hbound
  - Boundary\_condition, 27
- Headers/libboundaryconditions/bc\_imp\_discharge.↔  
hpp, 155
- Headers/libboundaryconditions/bc\_imp\_height.hpp,  
156
- Headers/libboundaryconditions/bc\_neumann.hpp, 156
- Headers/libboundaryconditions/bc\_periodic.hpp, 157
- Headers/libboundaryconditions/bc\_wall.hpp, 158
- Headers/libboundaryconditions/boundary\_condition.↔  
hpp, 158
- Headers/libboundaryconditions/choice\_condition.hpp,  
159
- Headers/libflux/choice\_flux.hpp, 160
- Headers/libflux/f\_hll.hpp, 161
- Headers/libflux/f\_hll2.hpp, 162
- Headers/libflux/f\_rusanov.hpp, 162
- Headers/libflux/flux.hpp, 163
- Headers/libfrictions/choice\_friction.hpp, 164
- Headers/libfrictions/fr\_darcy\_weisbach.hpp, 164
- Headers/libfrictions/fr\_laminar.hpp, 165
- Headers/libfrictions/fr\_manning.hpp, 166
- Headers/libfrictions/friction.hpp, 166
- Headers/libfrictions/no\_friction.hpp, 167
- Headers/libinitializations/choice\_init\_huv.hpp, 168
- Headers/libinitializations/choice\_init\_topo.hpp, 169
- Headers/libinitializations/huv\_generated.hpp, 169
- Headers/libinitializations/huv\_generated\_radial\_↔  
dam\_dry.hpp, 170
- Headers/libinitializations/huv\_generated\_radial\_↔  
dam\_wet.hpp, 171
- Headers/libinitializations/huv\_generated\_thacker.hpp,  
171
- Headers/libinitializations/huv\_read.hpp, 172
- Headers/libinitializations/initialization\_huv.hpp, 173
- Headers/libinitializations/initialization\_topo.hpp, 173
- Headers/libinitializations/topo\_generated\_flat.hpp, 174
- Headers/libinitializations/topo\_generated\_thacker.hpp,  
175
- Headers/libinitializations/topo\_read.hpp, 175
- Headers/liblimitations/choice\_limiter.hpp, 176
- Headers/liblimitations/limiter.hpp, 177
- Headers/liblimitations/minmod.hpp, 178
- Headers/liblimitations/vanalbada.hpp, 178
- Headers/liblimitations/vanleer.hpp, 179
- Headers/libparameters/misc.hpp, 180
- Headers/libparameters/parameters.hpp, 182
- Headers/libparser/parser.hpp, 183
- Headers/librain\_infiltration/choice\_infiltration.hpp, 184
- Headers/librain\_infiltration/choice\_rain.hpp, 184
- Headers/librain\_infiltration/greenampt.hpp, 185
- Headers/librain\_infiltration/infiltration.hpp, 186
- Headers/librain\_infiltration/no\_infiltration.hpp, 187
- Headers/librain\_infiltration/no\_rain.hpp, 187
- Headers/librain\_infiltration/rain.hpp, 188
- Headers/librain\_infiltration/rain\_generated.hpp, 188
- Headers/librain\_infiltration/rain\_read.hpp, 189
- Headers/libreconstructions/choice\_reconstruction.↔  
hpp, 190
- Headers/libreconstructions/eno.hpp, 190
- Headers/libreconstructions/eno\_mod.hpp, 191
- Headers/libreconstructions/hydrostatic.hpp, 192
- Headers/libreconstructions/muscl.hpp, 192
- Headers/libreconstructions/reconstruction.hpp, 193
- Headers/libsave/choice\_output.hpp, 194
- Headers/libsave/gnuplot.hpp, 194
- Headers/libsave/no\_evolution\_file.hpp, 195
- Headers/libsave/output.hpp, 196
- Headers/libsave/vtk\_out.hpp, 196
- Headers/libschemas/choice\_scheme.hpp, 197
- Headers/libschemas/order1.hpp, 198
- Headers/libschemas/order2.hpp, 198
- Headers/libschemas/scheme.hpp, 199
- height\_Vinf\_tot
  - Scheme, 142
- height\_of\_tot
  - Scheme, 141
- hl\_rec
  - Hydrostatic, 79
- hmod

- Infiltration, 81
- hr\_rec
  - Hydrostatic, 79
- hs
  - Scheme, 142
- huv\_NF
  - Parameters, 121
- Huv\_generated, 71
  - ~Huv\_generated, 72
  - Huv\_generated, 72
  - initialization, 72
- Huv\_generated\_Radial\_Dam\_dry, 72
  - ~Huv\_generated\_Radial\_Dam\_dry, 73
  - Huv\_generated\_Radial\_Dam\_dry, 73
  - initialization, 73
- Huv\_generated\_Radial\_Dam\_wet, 74
  - ~Huv\_generated\_Radial\_Dam\_wet, 74
  - Huv\_generated\_Radial\_Dam\_wet, 74
  - initialization, 75
- Huv\_generated\_Thacker, 75
  - ~Huv\_generated\_Thacker, 76
  - Huv\_generated\_Thacker, 75
  - initialization, 76
- huv\_init
  - Parameters, 121
- huv\_namefile
  - Parameters, 121
- Huv\_read, 76
  - ~Huv\_read, 77
  - Huv\_read, 77
  - initialization, 77
- Hydrostatic, 77
  - ~Hydrostatic, 78
  - calcul, 78
  - get\_hhydro\_l, 78
  - get\_hhydro\_r, 79
  - hl\_rec, 79
  - hr\_rec, 79
  - Hydrostatic, 78
- IE\_CA
  - misc.hpp, 181
- imax\_NF
  - Parameters, 122
- imax\_coef
  - Parameters, 121
- imax\_init
  - Parameters, 121
- imax\_namefile
  - Parameters, 121
- inf
  - Parameters, 122
- Infiltration, 79
  - ~Infiltration, 80
  - calcul, 80
  - DX, 81
  - DY, 81
  - get\_Vin, 80
  - get\_hmod, 80
  - hmod, 81
  - Infiltration, 80
  - NXCELL, 81
  - NYCELL, 81
  - Vin, 81
- initial
  - Choice\_output, 41
  - Output, 100
- initialization
  - Choice\_init\_huv, 37
  - Choice\_init\_topo, 38
  - Huv\_generated, 72
  - Huv\_generated\_Radial\_Dam\_dry, 73
  - Huv\_generated\_Radial\_Dam\_wet, 75
  - Huv\_generated\_Thacker, 76
  - Huv\_read, 77
  - Initialization\_huv, 82
  - Initialization\_topo, 84
  - Topo\_generated\_Thacker, 148
  - Topo\_generated\_flat, 147
  - Topo\_read, 149
- Initialization\_huv, 81
  - ~Initialization\_huv, 82
  - DX, 82
  - DY, 82
  - initialization, 82
  - Initialization\_huv, 82
  - NXCELL, 83
  - NYCELL, 83
- Initialization\_topo, 83
  - ~Initialization\_topo, 84
  - DX, 84
  - DY, 84
  - initialization, 84
  - Initialization\_topo, 84
  - NXCELL, 84
  - NYCELL, 84
- Kc\_NF
  - Parameters, 122
- Kc\_coef
  - Parameters, 122
- Kc\_init
  - Parameters, 122
- Kc\_namefile
  - Parameters, 122
- Ks\_NF
  - Parameters, 122
- Ks\_coef

- Parameters, 122
- Ks\_init
  - Parameters, 122
- Ks\_namefile
  - Parameters, 122
- L
  - Parameters, 122
- I
  - Parameters, 123
- L\_IMP\_H
  - Scheme, 142
- L\_IMP\_Q
  - Scheme, 142
- Lbound
  - Parameters, 123
  - Scheme, 142
- left\_imp\_discharge
  - Parameters, 123
- left\_imp\_h
  - Parameters, 123
- lim
  - Parameters, 123
- Limiter, 84
  - ~Limiter, 85
  - calcul, 85
  - get\_rec, 85
  - Limiter, 85
  - rec, 86
- limiter
  - Reconstruction, 133
- MAX\_CFL\_X
  - misc.hpp, 181
- MAX\_CFL\_Y
  - misc.hpp, 181
- MAX\_ITER
  - misc.hpp, 181
- MAX\_SCAL
  - misc.hpp, 182
- MUSCL, 87
  - ~MUSCL, 88
  - calcul, 88
  - MUSCL, 88
- main
  - FullSWOF\_2D.cpp, 201
- maincalcflux
  - Scheme, 137
- maincalcscheme
  - Scheme, 138
- max
  - misc.hpp, 181
- min
  - misc.hpp, 182
- Minmod, 86
  - ~Minmod, 86
  - calcul, 87
  - Minmod, 86
- misc.hpp
  - CONST\_CFL\_X, 181
  - CONST\_CFL\_Y, 181
  - EPSILON, 181
  - GRAV, 181
  - GRAV\_DEM, 181
  - HE\_CA, 181
  - IE\_CA, 181
  - MAX\_CFL\_X, 181
  - MAX\_CFL\_Y, 181
  - MAX\_ITER, 181
  - MAX\_SCAL, 182
  - max, 181
  - min, 182
  - NB\_CHAR, 182
  - RATIO\_CLOSE\_CELL, 182
  - SCALAR, 182
  - TAB, 182
  - VE\_CA, 182
  - VERSION, 182
  - ZERO, 182
- modifENO
  - Parameters, 123
- n
  - Scheme, 142
- NB\_CHAR
  - misc.hpp, 182
- NBTIMES
  - Scheme, 142
- NXCELL
  - Boundary\_condition, 27
  - Friction, 67
  - Infiltration, 81
  - Initialization\_huv, 83
  - Initialization\_topo, 84
  - Output, 101
  - Rain, 128
  - Reconstruction, 133
  - Scheme, 142
- NYCELL
  - Boundary\_condition, 27
  - Friction, 67
  - Infiltration, 81
  - Initialization\_huv, 83
  - Initialization\_topo, 84
  - Output, 101
  - Rain, 129
  - Reconstruction, 133
  - Scheme, 142

- namefile\_Bound\_flux
  - Output, 101
- namefile\_Bound\_flux\_BT
  - Output, 101
- namefile\_Bound\_flux\_LR
  - Output, 101
- namefile\_check\_volume
  - Output, 101
- namefile\_final
  - Output, 101
- namefile\_init
  - Output, 101
- namefile\_res
  - Output, 101
- nbtimes
  - Parameters, 123
- newtonSolver
  - Bc\_imp\_discharge, 18
- No\_Evolution\_File, 89
  - ~No\_Evolution\_File, 90
  - No\_Evolution\_File, 89
  - write, 90
- No\_Friction, 90
  - ~No\_Friction, 91
  - calcul, 91
  - calculSf, 91
  - No\_Friction, 91
- No\_Infiltration, 92
  - ~No\_Infiltration, 93
  - calcul, 93
  - No\_Infiltration, 92
- No\_Rain, 93
  - ~No\_Rain, 94
  - No\_Rain, 94
  - rain\_func, 94
- Nxcell
  - Parameters, 123
- Nycell
  - Parameters, 123
- ORDER
  - Scheme, 142
- order
  - Parameters, 123
- Order1, 94
  - ~Order1, 95
  - calcul, 95
  - Order1, 95
- Order2, 96
  - ~Order2, 96
  - calcul, 96
  - Order2, 96
- out
  - Scheme, 142
- Output, 97
  - ~Output, 98
  - boundaries\_flux, 98
  - boundaries\_flux\_BT, 99
  - boundaries\_flux\_LR, 99
  - check\_vol, 99
  - DX, 101
  - DY, 101
  - final, 99
  - initial, 100
  - NXCELL, 101
  - NYCELL, 101
  - namefile\_Bound\_flux, 101
  - namefile\_Bound\_flux\_BT, 101
  - namefile\_Bound\_flux\_LR, 101
  - namefile\_check\_volume, 101
  - namefile\_final, 101
  - namefile\_init, 101
  - namefile\_res, 101
  - Output, 98
  - outputDirectory, 102
  - result, 100
  - write, 100
- output\_directory
  - Parameters, 123
- output\_format
  - Parameters, 124
- outputDirectory
  - Output, 102
- Parameters, 102
  - ~Parameters, 107
  - amortENO, 119
  - Bbound, 119
  - bottom\_imp\_discharge, 120
  - bottom\_imp\_h, 120
  - cfl\_fix, 120
  - dt\_fix, 120
  - dtheta\_NF, 120
  - dtheta\_coef, 120
  - dtheta\_init, 120
  - dtheta\_namefile, 120
  - dx, 120
  - dy, 120
  - fill\_array, 107
  - flux, 120
  - fric, 121
  - fric\_NF, 121
  - fric\_init, 121
  - fric\_namefile, 121
  - friccoef, 121
  - get\_Bbound, 108
  - get\_Kc\_coef, 112
  - get\_Kc\_init, 112

get\_KcNameFile, 112  
 get\_KcNameFileS, 112  
 get\_Ks\_coef, 112  
 get\_Ks\_init, 113  
 get\_KsNameFile, 113  
 get\_KsNameFileS, 113  
 get\_Lbound, 113  
 get\_Nxcell, 114  
 get\_Nycell, 114  
 get\_Psi\_coef, 115  
 get\_Psi\_init, 115  
 get\_PsiNameFile, 115  
 get\_PsiNameFileS, 115  
 get\_Rbound, 116  
 get\_T, 117  
 get\_Tbound, 117  
 get\_amortENO, 107  
 get\_bottom\_imp\_discharge, 108  
 get\_bottom\_imp\_h, 108  
 get\_cflfix, 108  
 get\_dtfix, 108  
 get\_dtheta\_coef, 108  
 get\_dtheta\_init, 109  
 get\_dthetaNameFile, 109  
 get\_dthetaNameFileS, 109  
 get\_dx, 109  
 get\_dy, 109  
 get\_flux, 109  
 get\_fric, 110  
 get\_fric\_init, 110  
 get\_friccoef, 110  
 get\_frictionNameFile, 110  
 get\_frictionNameFileS, 110  
 get\_huv, 110  
 get\_huvNameFile, 111  
 get\_huvNameFileS, 111  
 get\_imax\_coef, 111  
 get\_imax\_init, 111  
 get\_imaxNameFile, 111  
 get\_imaxNameFileS, 111  
 get\_inf, 112  
 get\_left\_imp\_discharge, 113  
 get\_left\_imp\_h, 113  
 get\_lim, 114  
 get\_modifENO, 114  
 get\_nbtimes, 114  
 get\_order, 114  
 get\_output, 115  
 get\_outputDirectory, 115  
 get\_rain, 116  
 get\_rainNameFile, 116  
 get\_rainNameFileS, 116  
 get\_rec, 116  
 get\_right\_imp\_discharge, 116  
 get\_right\_imp\_h, 117  
 get\_scheme\_type, 117  
 get\_suffix, 117  
 get\_top\_imp\_discharge, 117  
 get\_top\_imp\_h, 118  
 get\_topo, 118  
 get\_topographyNameFile, 118  
 get\_topographyNameFileS, 118  
 get\_zcrust\_coef, 118  
 get\_zcrust\_init, 118  
 get\_zcrustNameFile, 119  
 get\_zcrustNameFileS, 119  
 huv\_NF, 121  
 huv\_init, 121  
 huv\_namefile, 121  
 imax\_NF, 122  
 imax\_coef, 121  
 imax\_init, 121  
 imax\_namefile, 121  
 inf, 122  
 Kc\_NF, 122  
 Kc\_coef, 122  
 Kc\_init, 122  
 Kc\_namefile, 122  
 Ks\_NF, 122  
 Ks\_coef, 122  
 Ks\_init, 122  
 Ks\_namefile, 122  
 L, 122  
 l, 123  
 Lbound, 123  
 left\_imp\_discharge, 123  
 left\_imp\_h, 123  
 lim, 123  
 modifENO, 123  
 nbtimes, 123  
 Nxcell, 123  
 Nycell, 123  
 order, 123  
 output\_directory, 123  
 output\_format, 124  
 Parameters, 107  
 Psi\_NF, 124  
 Psi\_coef, 124  
 Psi\_init, 124  
 Psi\_namefile, 124  
 rain, 124  
 rain\_NF, 124  
 rain\_namefile, 124  
 Rbound, 124  
 rec, 124  
 right\_imp\_discharge, 124



- right\_imp\_h, 125
- scheme\_type, 125
- setparameters, 119
- suffix\_outputs, 125
- T, 125
- Tbound, 125
- top\_imp\_discharge, 125
- top\_imp\_h, 125
- topo, 125
- topo\_NF, 125
- topography\_namefile, 125
- zcrust\_NF, 126
- zcrust\_coef, 125
- zcrust\_init, 126
- zcrust\_namefile, 126
- Parser, 126
  - ~Parser, 127
  - getValue, 127
  - Parser, 126
- Psi\_NF
  - Parameters, 124
- Psi\_coef
  - Parameters, 124
- Psi\_init
  - Parameters, 124
- Psi\_namefile
  - Parameters, 124
- q1
  - Scheme, 143
- q1mod
  - Friction, 67
- q2
  - Scheme, 143
- q2mod
  - Friction, 67
- qs1
  - Scheme, 143
- qs2
  - Scheme, 143
- R\_IMP\_H
  - Scheme, 143
- R\_IMP\_Q
  - Scheme, 143
- RATIO\_CLOSE\_CELL
  - misc.hpp, 182
- Rain, 127
  - ~Rain, 128
  - DX, 128
  - DY, 128
  - NXCELL, 128
  - NYCELL, 129
  - Rain, 128
  - rain\_func, 128
  - Scheme, 143
- rain
  - Parameters, 124
- rain\_NF
  - Parameters, 124
- rain\_func
  - Choice\_rain, 43
  - No\_Rain, 94
  - Rain, 128
  - Rain\_generated, 130
  - Rain\_read, 131
- Rain\_generated, 129
  - ~Rain\_generated, 130
  - rain\_func, 130
  - Rain\_generated, 129
- rain\_namefile
  - Parameters, 124
- Rain\_read, 130
  - ~Rain\_read, 131
  - rain\_func, 131
  - Rain\_read, 131
- Rbound
  - Parameters, 124
  - Scheme, 143
- rec
  - Limiter, 86
  - Parameters, 124
- Reconstruction, 131
  - ~Reconstruction, 132
  - calcul, 132
  - delta\_z1, 133
  - delta\_z2, 133
  - limiter, 133
  - NXCELL, 133
  - NYCELL, 133
  - Reconstruction, 132
  - z1l, 133
  - z1r, 133
  - z2l, 133
  - z2r, 133
- result
  - Choice\_output, 42
  - Output, 100
- right\_imp\_discharge
  - Parameters, 124
- right\_imp\_h
  - Parameters, 125
- SCALAR
  - misc.hpp, 182
- SCHEME\_TYPE
  - Scheme, 143
- Scheme, 134

- ~Scheme, 136
- allocation, 136
- B\_IMP\_H, 138
- B\_IMP\_Q, 138
- Bbound, 138
- boundary, 137
- CFL\_FIX, 139
- calcul, 137
- cpu\_time, 139
- cur\_time, 139
- DT\_FIX, 139
- DX, 140
- DY, 140
- deallocation, 137
- delz1, 139
- delz2, 139
- delzc1, 139
- delzc2, 139
- dt1, 139
- dt\_first, 139
- dt\_max, 139
- dt\_output, 140
- end, 140
- f1, 140
- f2, 140
- f3, 140
- FRICCOEF, 140
- Fr, 140
- froude\_number, 137
- g1, 140
- g2, 140
- g3, 141
- h, 141
- h1l, 141
- h1left, 141
- h1r, 141
- h1right, 141
- h2l, 141
- h2left, 141
- h2r, 141
- h2right, 141
- height\_Vinf\_tot, 142
- height\_of\_tot, 141
- hs, 142
- L\_IMP\_H, 142
- L\_IMP\_Q, 142
- Lbound, 142
- maincalcflux, 137
- maincalcscheme, 138
- n, 142
- NBTIMES, 142
- NXCELL, 142
- NYCELL, 142
- ORDER, 142
- out, 142
- q1, 143
- q2, 143
- qs1, 143
- qs2, 143
- R\_IMP\_H, 143
- R\_IMP\_Q, 143
- Rain, 143
- Rbound, 143
- SCHEME\_TYPE, 143
- Scheme, 136
- start, 143
- T, 143
- T\_IMP\_H, 144
- T\_IMP\_Q, 144
- T\_output, 144
- Tbound, 144
- timecomputation, 144
- Total\_volume\_outflow, 144
- tx, 144
- ty, 144
- u, 144
- u1l, 144
- u1r, 144
- u2l, 145
- u2r, 145
- us, 145
- v, 145
- v1l, 145
- v1r, 145
- v2l, 145
- v2r, 145
- verif, 145
- Vin\_tot, 145
- Vol\_inf\_tot\_cumul, 145
- Vol\_of\_tot, 146
- Volrain\_Tot, 146
- vs, 146
- z, 146
- scheme\_type
  - Parameters, 125
- set\_tx
  - Choice\_flux, 32
  - Flux, 57
- setparameters
  - Parameters, 119
- Sf1
  - Friction, 67
- Sf2
  - Friction, 68
- Sources/FullSWOF\_2D.cpp, 200

- Sources/libboundaryconditions/bc\_imp\_discharge.↔  
cpp, 201
- Sources/libboundaryconditions/bc\_imp\_height.cpp,  
201
- Sources/libboundaryconditions/bc\_neumann.cpp, 202
- Sources/libboundaryconditions/bc\_periodic.cpp, 203
- Sources/libboundaryconditions/bc\_wall.cpp, 203
- Sources/libboundaryconditions/boundary\_condition.↔  
cpp, 204
- Sources/libboundaryconditions/choice\_condition.cpp,  
204
- Sources/libflux/choice\_flux.cpp, 205
- Sources/libflux/f\_hll.cpp, 205
- Sources/libflux/f\_hll2.cpp, 206
- Sources/libflux/f\_rusanov.cpp, 206
- Sources/libflux/flux.cpp, 207
- Sources/libfrictions/choice\_friction.cpp, 207
- Sources/libfrictions/fr\_darcy\_weisbach.cpp, 208
- Sources/libfrictions/fr\_laminar.cpp, 208
- Sources/libfrictions/fr\_manning.cpp, 209
- Sources/libfrictions/friction.cpp, 209
- Sources/libfrictions/no\_friction.cpp, 210
- Sources/libinitializations/choice\_init\_huv.cpp, 210
- Sources/libinitializations/choice\_init\_topo.cpp, 211
- Sources/libinitializations/huv\_generated.cpp, 211
- Sources/libinitializations/huv\_generated\_radial\_dam↔  
\_dry.cpp, 212
- Sources/libinitializations/huv\_generated\_radial\_dam↔  
\_wet.cpp, 212
- Sources/libinitializations/huv\_generated\_thacker.cpp,  
213
- Sources/libinitializations/huv\_read.cpp, 213
- Sources/libinitializations/initialization\_huv.cpp, 214
- Sources/libinitializations/initialization\_topo.cpp, 214
- Sources/libinitializations/topo\_generated\_flat.cpp, 215
- Sources/libinitializations/topo\_generated\_thacker.cpp,  
216
- Sources/libinitializations/topo\_read.cpp, 216
- Sources/liblimitations/choice\_limiter.cpp, 217
- Sources/liblimitations/limiter.cpp, 217
- Sources/liblimitations/minmod.cpp, 218
- Sources/liblimitations/vanalbada.cpp, 218
- Sources/liblimitations/vanleer.cpp, 219
- Sources/libparameters/parameters.cpp, 219
- Sources/libparser/parser.cpp, 220
- Sources/librain\_infiltration/choice\_infiltration.cpp, 220
- Sources/librain\_infiltration/choice\_rain.cpp, 221
- Sources/librain\_infiltration/greenampt.cpp, 221
- Sources/librain\_infiltration/infiltration.cpp, 222
- Sources/librain\_infiltration/no\_infiltration.cpp, 222
- Sources/librain\_infiltration/no\_rain.cpp, 223
- Sources/librain\_infiltration/rain.cpp, 223
- Sources/librain\_infiltration/rain\_generated.cpp, 224
- Sources/librain\_infiltration/rain\_read.cpp, 224
- Sources/libreconstructions/choice\_reconstruction.cpp,  
225
- Sources/libreconstructions/eno.cpp, 225
- Sources/libreconstructions/eno\_mod.cpp, 226
- Sources/libreconstructions/hydrostatic.cpp, 226
- Sources/libreconstructions/muscl.cpp, 227
- Sources/libreconstructions/reconstruction.cpp, 227
- Sources/libsave/choice\_output.cpp, 228
- Sources/libsave/gnuplot.cpp, 228
- Sources/libsave/no\_evolution\_file.cpp, 229
- Sources/libsave/output.cpp, 229
- Sources/libsave/vtk\_out.cpp, 230
- Sources/libschemas/choice\_scheme.cpp, 230
- Sources/libschemas/order1.cpp, 231
- Sources/libschemas/order2.cpp, 231
- Sources/libschemas/scheme.cpp, 232
- start
  - Scheme, 143
- suffix\_outputs
  - Parameters, 125
- T
  - Parameters, 125
  - Scheme, 143
- T\_IMP\_H
  - Scheme, 144
- T\_IMP\_Q
  - Scheme, 144
- T\_output
  - Scheme, 144
- TAB
  - misc.hpp, 182
- Tbound
  - Parameters, 125
  - Scheme, 144
- timecomputation
  - Scheme, 144
- top\_imp\_discharge
  - Parameters, 125
- top\_imp\_h
  - Parameters, 125
- topo
  - Parameters, 125
- topo\_NF
  - Parameters, 125
- Topo\_generated\_Thacker, 147
  - ~Topo\_generated\_Thacker, 148
  - initialization, 148
  - Topo\_generated\_Thacker, 148
- Topo\_generated\_flat, 146
  - ~Topo\_generated\_flat, 147
  - initialization, 147
  - Topo\_generated\_flat, 147

- Topo\_read, 148
  - ~Topo\_read, 149
  - initialization, 149
  - Topo\_read, 149
- topography\_namefile
  - Parameters, 125
- Total\_volume\_outflow
  - Scheme, 144
- tx
  - Flux, 58
  - Scheme, 144
- ty
  - Scheme, 144
- u
  - Scheme, 144
- u1l
  - Scheme, 144
- u1r
  - Scheme, 144
- u2l
  - Scheme, 145
- u2r
  - Scheme, 145
- unormbound
  - Boundary\_condition, 28
- unormfix
  - Boundary\_condition, 28
- us
  - Scheme, 145
- utanbound
  - Boundary\_condition, 28
- v
  - Scheme, 145
- v1l
  - Scheme, 145
- v1r
  - Scheme, 145
- v2l
  - Scheme, 145
- v2r
  - Scheme, 145
- VE\_CA
  - misc.hpp, 182
- VERSION
  - misc.hpp, 182
- VanAlbada, 150
  - ~VanAlbada, 150
  - calcul, 151
  - VanAlbada, 150
- VanLeer, 151
  - ~VanLeer, 152
  - calcul, 152
  - VanLeer, 152
- verif
  - Scheme, 145
- Vin
  - Infiltration, 81
- Vin\_tot
  - Scheme, 145
- Vol\_inf\_tot\_cumul
  - Scheme, 145
- Vol\_of\_tot
  - Scheme, 146
- Volrain\_Tot
  - Scheme, 146
- vs
  - Scheme, 146
- Vtk\_Out, 152
  - ~Vtk\_Out, 153
  - Vtk\_Out, 153
  - write, 153
- write
  - Choice\_output, 42
  - Gnuplot, 69
  - No\_Evolution\_File, 90
  - Output, 100
  - Vtk\_Out, 153
- z
  - Scheme, 146
- z1l
  - Reconstruction, 133
- z1r
  - Reconstruction, 133
- z2l
  - Reconstruction, 133
- z2r
  - Reconstruction, 133
- ZERO
  - misc.hpp, 182
- zcrust\_NF
  - Parameters, 126
- zcrust\_coef
  - Parameters, 125
- zcrust\_init
  - Parameters, 126
- zcrust\_namefile
  - Parameters, 126