

Documentation
of
FullSWOF_1D

v1.02.01 (2015-04-03)

Generated by Doxygen 1.8.9.1
on Fri Apr 3 2015 11:06:26

Chapter 1

Todo List

Class `Boundary_condition`

Add time 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 \[2001\]](#).

Member `Choice_friction::set_c` (SCALAR)

For future evolutions: `Friction::set_c` will not be necessary when the friction coefficient will be read as in 2D.

Member `F_HLL::calc` (SCALAR, SCALAR, SCALAR, SCALAR)

Check if the use of long double should be generalized to other fluxes.

Member `Fr_Darcy_Weisbach::calc` (SCALAR, SCALAR, SCALAR)

For future evolutions: the friction coefficient should be added as in 2D as a parameter of `Fr_Darcy_Weisbach::calc`.

Member `Fr_Laminar::calc` (SCALAR, SCALAR, SCALAR)

For future evolutions: the friction coefficient should be added as in 2D as a parameter of `Fr_Laminar::calc`.

Member `Fr_Manning::calc` (SCALAR, SCALAR, SCALAR)

For future evolutions: the friction coefficient should be added as in 2D as a parameter of `Fr_Manning::calc`.

Member `Friction::set_c` (SCALAR)

For future evolutions: `Friction::set_c` will not be necessary when the friction coefficient will be read as in 2D.

Member `MUSCL::calc` (VECT, VECT, VECT, VECT &, VECT &, VECT &, VECT &, VECT &, VECT &)

Check if the use of long double should be generalized to other reconstructions.

Member `No_Friction::calc` (SCALAR, SCALAR, SCALAR)

For future evolutions: the friction coefficient should be added as in 2D as a parameter of `Fr_Manning::calc`.

Member `Scheme::maincalc` (VECT, VECT, SCALAR, VECT &, VECT &, VECT &, SCALAR &, SCALAR &, VECT &)

Improve the treatment of numerical errors.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Boundary_condition	24
Bc_imp_discharge	15
Bc_imp_height	18
Bc_neumann	19
Bc_periodic	21
Bc_wall	23
Choice_condition	26
Choice_flux	28
Choice_friction	30
Choice_infiltration	32
Choice_init_hu	33
Choice_init_topo	34
Choice_limiter	35
Choice_output	36
Choice_rain	39
Choice_reconstruction	40
Choice_scheme	41
Flux	52
F_HLL	45
F_HLL2	46
F_Kinetic	48
F_Rusanov	49
F_VFRoe	51
Friction	59
Fr_Darcy_Weisbach	54
Fr_Laminar	56
Fr_Manning	57
No_Friction	82
Hydrostatic	71
Infiltration	73
GreenAmpt	62
No_Infiltration	83
Initialization_hu	75
Hu_generated	64
Hu_generated_Dressler_Dam_break	65

Hu_generated_Dry_Dam_break	66
Hu_generated_Thacker	67
Hu_generated_Wet_Dam_break	69
Hu_read	70
Initialization_topo	76
Topo_generated_bump	129
Topo_generated_flat	130
Topo_generated_Thacker	131
Topo_read	132
Limiter	78
Minmod	79
VanAlbada	133
VanLeer	135
Output	88
Gnuplot	61
Parameters	92
Parser	113
Rain	114
No_Rain	84
Rain_generated	116
Rain_read	117
Reconstruction	118
ENO	42
ENO_mod	43
MUSCL	80
Scheme	120
Order1	85
Order2	87

Chapter 3

Class Index

3.1 Class List

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

Bc_imp_discharge	Imposed discharge	15
Bc_imp_height	Imposed water height	18
Bc_neumann	Neumann condition	19
Bc_periodic	Periodic condition	21
Bc_wall	Wall condition	23
Boundary_condition	Boundary condition	24
Choice_condition	Choice of boundary condition	26
Choice_flux	Choice of numerical flux	28
Choice_friction	Choice of friction law	30
Choice_infiltration	Choice of infiltration law	32
Choice_init_hu	Choice of initialization for h and u	33
Choice_init_topo	Choice of initialization for the topography	34
Choice_limiter	Choice of slope limiter	35
Choice_output	Choice of output format	36
Choice_rain	Choice of initialization for the rain	39
Choice_reconstruction	Choice of reconstruction	40
Choice_scheme	Choice of numerical scheme	41
ENO	ENO reconstruction	42

ENO_mod	Modified ENO reconstruction	43
F_HLL	HLL flux	45
F_HLL2	HLL flux	46
F_Kinetic	Kinetic flux	48
F_Rusanov	Rusanov flux	49
F_VFRoe	VFRoe flux	51
Flux	Numerical flux	52
Fr_Darcy_Weisbach	Darcy-Weisbach law	54
Fr_Laminar	Laminar law	56
Fr_Manning	Manning law	57
Friction	Friction law	59
Gnuplot	Gnuplot output	61
GreenAmpt	Green-Ampt law	62
Hu_generated	No water configuration	64
Hu_generated_Dressler_Dam_break	Dressler dam break configuration	65
Hu_generated_Dry_Dam_break	Dry dam break configuration	66
Hu_generated_Thacker	Thacker configuration	67
Hu_generated_Wet_Dam_break	Wet dam break configuration	69
Hu_read	File configuration	70
Hydrostatic	Hydrostatic reconstruction	71
Infiltration	Definition of infiltration law	73
Initialization_hu	Initialization of h and u	75
Initialization_topo	Initialization of z	76
Limiter	Slope limiter	78
Minmod	Minmod slope limiter	79
MUSCL	MUSCL reconstruction	80

No_Friction	
No friction	82
No_Infiltration	
No infiltration	83
No_Rain	
No rain	84
Order1	
Order 1 scheme	85
Order2	
Order 2 scheme	87
Output	
Output format	88
Parameters	
Gets parameters	92
Parser	
Parser to read the entries	113
Rain	
Initialization of the rain	114
Rain_generated	
Constant rain configuration	116
Rain_read	
File configuration	117
Reconstruction	
Reconstruction of the variables	118
Scheme	
Numerical scheme	120
Topo_generated_bump	
Bump configuration	129
Topo_generated_flat	
Flat configuration	130
Topo_generated_Thacker	
Thacker configuration	131
Topo_read	
File configuration	132
VanAlbada	
Van Albada slope limiter	133
VanLeer	
Van Leer slope limiter	135

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Headers/libboundaryconditions/ bc_imp_discharge.hpp	
Imposed discharge	137
Headers/libboundaryconditions/ bc_imp_height.hpp	
Imposed water height	138
Headers/libboundaryconditions/ bc_neumann.hpp	
Neumann condition	138
Headers/libboundaryconditions/ bc_periodic.hpp	
Periodic condition	139
Headers/libboundaryconditions/ bc_wall.hpp	
Wall condition	140
Headers/libboundaryconditions/ boundary_condition.hpp	
Boundary condition	140
Headers/libboundaryconditions/ choice_condition.hpp	
Choice of boundary condition	141
Headers/libflux/ choice_flux.hpp	
Choice of numerical flux	142
Headers/libflux/ f_hll.hpp	
HLL flux	143
Headers/libflux/ f_hll2.hpp	
HLL flux	144
Headers/libflux/ f_kinetic.hpp	
Kinetic flux	144
Headers/libflux/ f_rusanov.hpp	
Rusanov flux	145
Headers/libflux/ f_vfroe.hpp	
VFRoe flux	146
Headers/libflux/ flux.hpp	
Numerical flux	146
Headers/libfrictions/ choice_friction.hpp	
Choice of friction law	147
Headers/libfrictions/ fr_darcy_weisbach.hpp	
Darcy-Weisbach law	148
Headers/libfrictions/ fr_laminar.hpp	
Laminar law	149
Headers/libfrictions/ fr_manning.hpp	
Manning law	149

Headers/libfrictions/ friction.hpp	
Friction law	150
Headers/libfrictions/ no_friction.hpp	
No friction	151
Headers/libinitializations/ choice_init_hu.hpp	
Choice of initialization for h and u	151
Headers/libinitializations/ choice_init_topo.hpp	
Choice of initialization for the topography	152
Headers/libinitializations/ hu_generated.hpp	
No water configuration	153
Headers/libinitializations/ hu_generated_dressler_dam_break.hpp	
Dressler dam break configuration	154
Headers/libinitializations/ hu_generated_dry_dam_break.hpp	
Dry dam break configuration	154
Headers/libinitializations/ hu_generated_thacker.hpp	
Thacker configuration	155
Headers/libinitializations/ hu_generated_wet_dam_break.hpp	
Wet dam break configuration	156
Headers/libinitializations/ hu_read.hpp	
File configuration	156
Headers/libinitializations/ initialization_hu.hpp	
Initialization of h and u	157
Headers/libinitializations/ initialization_topo.hpp	
Initialization of z	157
Headers/libinitializations/ topo_generated_bump.hpp	
Bump configuration	158
Headers/libinitializations/ topo_generated_flat.hpp	
Flat configuration	159
Headers/libinitializations/ topo_generated_thacker.hpp	
Thacker configuration	159
Headers/libinitializations/ topo_read.hpp	
File configuration	160
Headers/liblimitations/ choice_limiter.hpp	
Choice of slope limiter	161
Headers/liblimitations/ limiter.hpp	
Slope limiter	162
Headers/liblimitations/ minmod.hpp	
Minmod limiter	162
Headers/liblimitations/ vanalbada.hpp	
Van Albada limiter	163
Headers/liblimitations/ vanleer.hpp	
Van Leer limiter	164
Headers/libparameters/ misc.hpp	
Definitions	164
Headers/libparameters/ parameters.hpp	
Gets parameters	167
Headers/libparser/ parser.hpp	
Parser	167
Headers/librain_infiltration/ choice_infiltration.hpp	
Choice of infiltration law	168
Headers/librain_infiltration/ choice_rain.hpp	
Choice of initialization for the rain	169

Headers/librain_infiltration/ greenampt.hpp	
Green-Ampt law	169
Headers/librain_infiltration/ infiltration.hpp	
Infiltration	170
Headers/librain_infiltration/ no_infiltration.hpp	
No infiltration	171
Headers/librain_infiltration/ no_rain.hpp	
No rain	171
Headers/librain_infiltration/ rain.hpp	
Rain	172
Headers/librain_infiltration/ rain_generated.hpp	
Constant rain configuration	172
Headers/librain_infiltration/ rain_read.hpp	
File configuration	173
Headers/libreconstructions/ choice_reconstruction.hpp	
Choice of reconstruction	174
Headers/libreconstructions/ eno.hpp	
ENO reconstruction	174
Headers/libreconstructions/ eno_mod.hpp	
Modified ENO reconstruction	175
Headers/libreconstructions/ hydrostatic.hpp	
Hydrostatic reconstruction	176
Headers/libreconstructions/ muscl.hpp	
MUSCL reconstruction	176
Headers/libreconstructions/ reconstruction.hpp	
Reconstruction	177
Headers/libsave/ choice_output.hpp	
Choice of output format	178
Headers/libsave/ gnuplot.hpp	
Gnuplot output	178
Headers/libsave/ output.hpp	
Output format	179
Headers/libschemes/ choice_scheme.hpp	
Choice of numerical scheme	180
Headers/libschemes/ order1.hpp	
Order 1 scheme	180
Headers/libschemes/ order2.hpp	
Order 2 scheme	181
Headers/libschemes/ scheme.hpp	
Numerical scheme	182
Sources/ FullSWOF_1D.cpp	
Main function	183
Sources/libboundaryconditions/ bc_imp_discharge.cpp	
Imposed discharge	183
Sources/libboundaryconditions/ bc_imp_height.cpp	
Imposed water height	184
Sources/libboundaryconditions/ bc_neumann.cpp	
Neumann condition	185
Sources/libboundaryconditions/ bc_periodic.cpp	
Periodic condition	185
Sources/libboundaryconditions/ bc_wall.cpp	
Wall condition	186

Sources/libboundaryconditions/ boundary_condition.cpp	
Boundary condition	186
Sources/libboundaryconditions/ choice_condition.cpp	
Choice of boundary condition	187
Sources/libflux/ choice_flux.cpp	
Choice of numerical flux	187
Sources/libflux/ f_hll.cpp	
HLL flux	188
Sources/libflux/ f_hll2.cpp	
HLL flux	188
Sources/libflux/ f_kinetic.cpp	
Kinetic flux	189
Sources/libflux/ f_rusanov.cpp	
Rusanov flux	189
Sources/libflux/ f_vfroee.cpp	
VFRoe flux	190
Sources/libflux/ flux.cpp	
Numerical flux	190
Sources/libfrictions/ choice_friction.cpp	
Choice of friction law	191
Sources/libfrictions/ fr_darcy_weisbach.cpp	
Darcy-Weisbach law	191
Sources/libfrictions/ fr_laminar.cpp	
Laminar law	192
Sources/libfrictions/ fr_manning.cpp	
Manning law	192
Sources/libfrictions/ friction.cpp	
Friction law	193
Sources/libfrictions/ no_friction.cpp	
No friction	193
Sources/libinitializations/ choice_init_hu.cpp	
Choice of initialization for h and u	194
Sources/libinitializations/ choice_init_topo.cpp	
Choice of initialization for the topography	194
Sources/libinitializations/ hu_generated.cpp	
No water configuration	195
Sources/libinitializations/ hu_generated_dressler_dam_break.cpp	
Dressler dam break configuration	195
Sources/libinitializations/ hu_generated_dry_dam_break.cpp	
Dry dam break configuration	196
Sources/libinitializations/ hu_generated_thacker.cpp	
Thacker configuration	196
Sources/libinitializations/ hu_generated_wet_dam_break.cpp	
Wet dam break configuration	197
Sources/libinitializations/ hu_read.cpp	
File configuration	197
Sources/libinitializations/ initialization_hu.cpp	
Initialization of h and u	198
Sources/libinitializations/ initialization_topo.cpp	
Initialization of z	198
Sources/libinitializations/ topo_generated_bump.cpp	
Bump configuration	199

Sources/libinitializations/ topo_generated_flat.cpp	
Flat configuration	199
Sources/libinitializations/ topo_generated_thacker.cpp	
Thacker configuration	200
Sources/libinitializations/ topo_read.cpp	
File configuration	200
Sources/liblimitations/ choice_limiter.cpp	
Choice of slope limiter	201
Sources/liblimitations/ limiter.cpp	
Slope limiter	201
Sources/liblimitations/ minmod.cpp	
Minmod limiter	202
Sources/liblimitations/ vanalbada.cpp	
Van Albada limiter	202
Sources/liblimitations/ vanleer.cpp	
Van Leer limiter	203
Sources/libparameters/ parameters.cpp	
Gets parameters	203
Sources/libparser/ parser.cpp	
Parser	204
Sources/librain_infiltration/ choice_infiltration.cpp	
Choice of infiltration law	204
Sources/librain_infiltration/ choice_rain.cpp	
Choice of initialization for the rain	205
Sources/librain_infiltration/ greenampt.cpp	
Green-Ampt law	205
Sources/librain_infiltration/ infiltration.cpp	
Infiltration	206
Sources/librain_infiltration/ no_infiltration.cpp	
No infiltration	206
Sources/librain_infiltration/ no_rain.cpp	
No rain	207
Sources/librain_infiltration/ rain.cpp	
Rain	207
Sources/librain_infiltration/ rain_generated.cpp	
Constant rain configuration	208
Sources/librain_infiltration/ rain_read.cpp	
File configuration	208
Sources/libreconstructions/ choice_reconstruction.cpp	
Choice of reconstruction	209
Sources/libreconstructions/ eno.cpp	
ENO reconstruction	209
Sources/libreconstructions/ eno_mod.cpp	
Modified ENO reconstruction	210
Sources/libreconstructions/ hydrostatic.cpp	
Hydrostatic reconstruction	210
Sources/libreconstructions/ muscl.cpp	
MUSCL reconstruction	211
Sources/libreconstructions/ reconstruction.cpp	
Reconstruction	211
Sources/libsave/ choice_output.cpp	
Choice of output format	212

Sources/libsave/ gnuplot.cpp	
Gnuplot output	212
Sources/libsave/ output.cpp	
Output format	213
Sources/libschemas/ choice_scheme.cpp	
Choice of numerical scheme	213
Sources/libschemas/ order1.cpp	
Order 1 scheme	214
Sources/libschemas/ order2.cpp	
Order 2 scheme	214
Sources/libschemas/ scheme.cpp	
Numerical scheme	215

Chapter 5

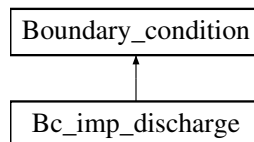
Class Documentation

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 &`, `VECT &`, `int`)
Constructor.
- `void calc` (`SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `int`)
Calculates the boundary condition.
- `SCALAR NewtonSolver` (`const SCALAR`, `const SCALAR`, `const SCALAR`, `const SCALAR`, `int`) `const`
Solves the equation with Newton iterative method.
- `SCALAR ValPoly` (`const SCALAR`, `const SCALAR`, `const SCALAR`, `const SCALAR`, `int`) `const`
Gives the value of the function that must vanish.
- `SCALAR ValDerPoly` (`const SCALAR`, `const SCALAR`, `const SCALAR`, `int`) `const`
Gives the value of the derivative of the function that must vanish.
- `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 72 of file `bc_imp_discharge.hpp`.

5.1.2 Constructor & Destructor Documentation

`Bc_imp_discharge::Bc_imp_discharge` (`Parameters &` *par*, `VECT &` *z*, `int` *normal*)

Constructor.

Parameters

in	<i>par</i>	parameter, contains all the values from the parameters file (unused).
in	<i>normal</i>	integer to specify whether it is the left (-1) or the right (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 209 of file `bc_imp_discharge.cpp`.

5.1.3 Member Function Documentation

void Bc_imp_discharge::calc (SCALAR *hin*, SCALAR *uin*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_oppbound*, SCALAR *uin_oppbound*, SCALAR *time*, int *normal*) [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>uin</i>	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>uin_oppbound</i>	value of the velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>time</i>	current time (unused).
in	<i>normal</i>	integer to specify whether it is the left (-1) or the right (1) boundary.

Warning

Warning in the method `Bc_imp_discharge::calc()` The water height at the inflow is zero ... continuing!

Modifies

`Boundary_condition::hbound` water height on the fictive cell.

`Boundary_condition::ubound` velocity on the fictive cell.

Implements `Boundary_condition`.

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

SCALAR Bc_imp_discharge::NewtonSolver (const SCALAR *HIN*, const SCALAR *UIN*, const SCALAR *QFIX*, const SCALAR *H_INIT*, int *normal*) 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>UIN</i>	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>normal</i>	integer to specify whether it is the left (-1) or the right (1) boundary.

Warning

Warning: Newton bc did not converge.

Returns

h: water height that satisfies Riemann invariants.

Definition at line 141 of file `bc_imp_discharge.cpp`.

SCALAR Bc_imp_discharge::ValDerPoly (const SCALAR *HIN*, const SCALAR *UIN*, const SCALAR *H*, int *normal*) const

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

Computes $3\sqrt{gH} - n(UIN + 2n\sqrt{gHIN})$ where n is the normal.

Parameters

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

Returns

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

Definition at line 193 of file `bc_imp_discharge.cpp`.

SCALAR Bc_imp_discharge::ValPoly (const SCALAR *HIN*, const SCALAR *UIN*, const SCALAR *QFIX*, const SCALAR *H*, int *normal*) const

Gives the value of the function that must vanish.

Computes $2H\sqrt{gH} - n(UIN + 2n\sqrt{gHIN})H - |QFIX|$ where n is the normal.

Parameters

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

Returns

The value of the polynomial function.

Definition at line 177 of file `bc_imp_discharge.cpp`.

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

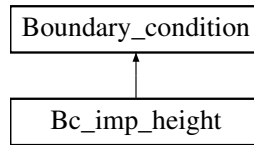
- Headers/libboundaryconditions/`bc_imp_discharge.hpp`
- Sources/libboundaryconditions/`bc_imp_discharge.cpp`

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 &, VECT &, int)`
Constructor.
- `void calc (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, 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*, VECT & *z*, int *normal*)

Constructor.

Parameters

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

Definition at line 62 of file bc_imp_height.cpp.

Bc_imp_height::~~Bc_imp_height () [virtual]

Destructor.

Definition at line 142 of file bc_imp_height.cpp.

5.2.3 Member Function Documentation

void Bc_imp_height::calc (SCALAR *hin*, SCALAR *uin*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_opbound*, SCALAR *uin_opbound*, SCALAR *time*, int *normal*) [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>uin</i>	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>uin_oppbound</i>	value of the velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>time</i>	current time (unused).
in	<i>normal</i>	integer to specify whether it is the left (-1) or the right (1) boundary.

Modifies

[Boundary_condition::hbound](#) water height on the fictive cell.

[Boundary_condition::ubound](#) velocity on the fictive cell.

Implements [Boundary_condition](#).

Definition at line 84 of file `bc_imp_height.cpp`.

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

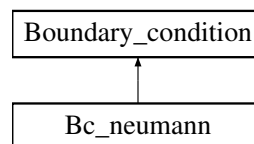
- `Headers/libboundaryconditions/bc_imp_height.hpp`
- `Sources/libboundaryconditions/bc_imp_height.cpp`

5.3 Bc_neumann Class Reference

Neumann condition.

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

Inheritance diagram for `Bc_neumann`:

**Public Member Functions**

- [Bc_neumann](#) ([Parameters](#) &, [VECT](#) &, int)
Constructor.
- void [calc](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), 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 71 of file `bc_neumann.hpp`.

5.3.2 Constructor & Destructor Documentation

Bc_neumann::Bc_neumann (Parameters & *par*, VECT & *z*, int *normal*)

Constructor.

Parameters

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

Definition at line 59 of file bc_neumann.cpp.

Bc_neumann::~Bc_neumann () [virtual]

Destructor.

Definition at line 109 of file bc_neumann.cpp.

5.3.3 Member Function Documentation

void Bc_neumann::calc (SCALAR *hin*, SCALAR *uin*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_oppbound*, SCALAR *uin_oppbound*, SCALAR *time*, int *normal*) [virtual]

Calculates the boundary condition.

Parameters

in	<i>hin</i>	water height of the first cell inside the domain.
in	<i>uin</i>	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>uin_oppbound</i>	value of the velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>time</i>	current time (unused).
in	<i>normal</i>	integer to specify whether it is the left (-1) or the right (1) boundary (unused).

Modifies

[Boundary_condition::hbound](#) water height on the fictive cell.

[Boundary_condition::ubound](#) velocity on the fictive cell.

Implements [Boundary_condition](#).

Definition at line 81 of file bc_neumann.cpp.

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

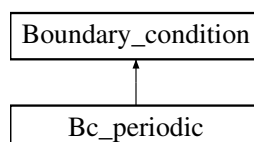
- Headers/libboundaryconditions/bc_neumann.hpp
- Sources/libboundaryconditions/bc_neumann.cpp

5.4 Bc_periodic Class Reference

Periodic condition.

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

Inheritance diagram for Bc_periodic:



Public Member Functions

- `Bc_periodic` (`Parameters` &, `VECT` &, `int`)
Constructor.
- `void calc` (`SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `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 71 of file `bc_periodic.hpp`.

5.4.2 Constructor & Destructor Documentation

`Bc_periodic::Bc_periodic` (`Parameters` & *par*, `VECT` & *z*, `int` *normal*)

Constructor.

Parameters

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

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

`Bc_periodic::~~Bc_periodic` () [`virtual`]

Destructor.

Definition at line 111 of file `bc_periodic.cpp`.

5.4.3 Member Function Documentation

`void Bc_periodic::calc` (`SCALAR` *hin*, `SCALAR` *uin*, `SCALAR` *hfix*, `SCALAR` *qfix*, `SCALAR` *hin_oppbound*, `SCALAR` *uin_oppbound*, `SCALAR` *time*, `int` *normal*) [`virtual`]

Calculates boundary condition.

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

Parameters

<code>in</code>	<i>hin</i>	water height of the first cell inside the domain (unused).
<code>in</code>	<i>uin</i>	velocity of the first cell inside the domain (unused).
<code>in</code>	<i>hfix</i>	fixed (imposed) value of the water height (unused).
<code>in</code>	<i>qfix</i>	fixed (imposed) value of the discharge (unused).
<code>in</code>	<i>hin_oppbound</i>	value of the water height of the first cell inside the domain at the opposite bound.
<code>in</code>	<i>uin_oppbound</i>	value of the velocity of the first cell inside the domain at the opposite bound.

<code>in</code>	<code>time</code>	current time (unused).
<code>in</code>	<code>normal</code>	integer to specify whether it is the left (-1) or the right (1) boundary (unused).

Modifies

[Boundary_condition::hbound](#) water height on the fictive cell.

[Boundary_condition::ubound](#) velocity on the fictive cell.

Implements [Boundary_condition](#).

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

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

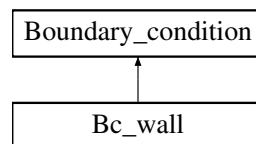
- [Headers/libboundaryconditions/bc_periodic.hpp](#)
- [Sources/libboundaryconditions/bc_periodic.cpp](#)

5.5 Bc_wall Class Reference

Wall condition.

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

Inheritance diagram for `Bc_wall`:



Public Member Functions

- [Bc_wall](#) ([Parameters](#) &, [VECT](#) &, int)
Constructor.
- void [calc](#) ([SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), 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, VECT & z, int normal)`

Constructor.

Parameters

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

Definition at line 59 of file bc_wall.cpp.

Bc_wall::~Bc_wall() [virtual]

Destructor.

Definition at line 110 of file bc_wall.cpp.

5.5.3 Member Function Documentation

void Bc_wall::calc (SCALAR *hin*, SCALAR *uin*, SCALAR *hfix*, SCALAR *qfix*, SCALAR *hin_oppbound*, SCALAR *uin_oppbound*, SCALAR *time*, int *normal*) [virtual]

Calculates the boundary condition.

The water height (on the fictive cell) is the same as inside the domain and the velocity is the opposite. This gives a null discharge at the boundary.

Parameters

in	<i>hin</i>	water height of the first cell inside the domain.
in	<i>uin</i>	velocity of the first cell inside the domain.
in	<i>hfix</i>	fixed (imposed) value of the water height (only for the supercritical case) (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>uin_oppbound</i>	value of the velocity of the first cell inside the domain at the opposite bound (unused).
in	<i>time</i>	current time (unused).
in	<i>normal</i>	integer to specify whether it is the left (-1) or the right (1) boundary (unused).

Modifies

[Boundary_condition::hbound](#) water height on the fictive cell.

[Boundary_condition::ubound](#) velocity on the fictive cell.

Implements [Boundary_condition](#).

Definition at line 80 of file bc_wall.cpp.

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

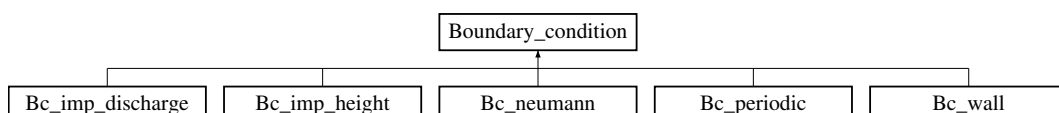
- Headers/libboundaryconditions/bc_wall.hpp
- Sources/libboundaryconditions/bc_wall.cpp

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 [calc](#) (SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, SCALAR, int)=0
Function to be specified in each boundary condition.
- [SCALAR get_hbound](#) () const
Gives the water height on the fictive cell.
- [SCALAR get_ubound](#) () const
Gives the velocity of the flow on the fictive cell.
- virtual [~Boundary_condition](#) ()
Destructor.

Protected Attributes

- [SCALAR hbound](#)
- [SCALAR ubound](#)
- [SCALAR ufix](#)
- const int [NXCELL](#)

5.6.1 Detailed Description

Boundary condition.

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

Todo Add time 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 72 of file `boundary_condition.hpp`.

5.6.2 Constructor & Destructor Documentation

Boundary_condition::Boundary_condition (Parameters & par)

Constructor.

Defines the number of cells.

Parameters

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

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

Boundary_condition::~~Boundary_condition () [virtual]

Destructor.

Definition at line 87 of file `boundary_condition.cpp`.

5.6.3 Member Function Documentation

virtual void Boundary_condition::calc (SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , SCALAR , int) [pure virtual]

Function to be specified in each boundary condition.

Implemented in [Bc_imp_height](#), [Bc_imp_discharge](#), [Bc_neumann](#), [Bc_periodic](#), 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 67 of file boundary_condition.cpp.

SCALAR Boundary_condition::get_ubound () const

Gives the velocity of the flow on the fictive cell.

Returns

[Boundary_condition::ubound](#) velocity on the fictive cell.

Definition at line 77 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 94 of file boundary_condition.hpp.

const int Boundary_condition::NXCELL [protected]

Number of cells (in space).

Definition at line 100 of file boundary_condition.hpp.

SCALAR Boundary_condition::ubound [protected]

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

Definition at line 96 of file boundary_condition.hpp.

SCALAR Boundary_condition::ufix [protected]

Imposed value of the velocity from [Parameters::L_imp_q](#) (or [Parameters::R_imp_q](#)) and [Parameters::L_imp_h](#) (or [Parameters::R_imp_h](#)).

Definition at line 98 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` &, `VECT` &, int)
Constructor.
- void `calc` (`SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, `SCALAR`, int)
Calculates the boundary condition.
- `SCALAR get_hbound` ()
Gives the water height on the fictive cell.
- `SCALAR get_ubound` ()
Gives the 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 90 of file `choice_condition.hpp`.

5.7.2 Constructor & Destructor Documentation

`Choice_condition::Choice_condition (int choice, Parameters & par, VECT & z, int normal)`

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>	vector that represents the topography.
in	<i>normal</i>	integer to specify whether it is the left (-1) or the right (1) boundary.

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

`Choice_condition::~~Choice_condition () [virtual]`

Destructor.

Definition at line 129 of file `choice_condition.cpp`.

5.7.3 Member Function Documentation

`void Choice_condition::calc (SCALAR hin, SCALAR uin, SCALAR hfix, SCALAR qfix, SCALAR hin_oppbound, SCALAR uin_oppbound, SCALAR time, int normal)`

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>uin</i>	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>uin_oppbound</i>	value of the velocity of the first cell inside the domain at the opposite bound.
in	<i>time</i>	current time (unused, for the moment).
in	<i>normal</i>	integer to specify whether it is the left (-1) or the right (1) boundary.

Definition at line 89 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 107 of file choice_condition.cpp.

SCALAR Choice_condition::get_ubound ()

Gives the velocity of the flow on the fictive cell.

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

Returns

[Boundary_condition::ubound](#) velocity on the fictive cell for the chosen boundary condition.

Definition at line 118 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 [calc](#) (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_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 92 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

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

Definition at line 59 of file choice_flux.cpp.

Choice_flux::~~Choice_flux () [virtual]

Destructor.

Definition at line 144 of file choice_flux.cpp.

5.8.3 Member Function Documentation

void Choice_flux::calc (SCALAR *h_L*, SCALAR *u_L*, SCALAR *h_R*, SCALAR *u_R*)

Calculates the numerical flux.

Calls the calculation of the numerical flux.

Parameters

in	<i>h_L</i>	water height at the left of the interface where the flux is calculated.
in	<i>u_L</i>	velocity at the left of the interface where the flux is calculated.
in	<i>h_R</i>	water height at the right of the interface where the flux is calculated.
in	<i>u_R</i>	velocity at the right of the interface where the flux is calculated.

Definition at line 86 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 133 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.

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](#) (int)
Constructor.
- void [calc](#) (SCALAR, SCALAR, SCALAR)
Calculates the friction term.
- [SCALAR get_qmod](#) ()
Gives the discharge modified by the friction term.
- void [set_c](#) (SCALAR)
Sets the variable [Friction::c](#).
- void [set_dt](#) (SCALAR)
Sets the variable [Friction::dt](#).
- 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 86 of file choice_friction.hpp.

5.9.2 Constructor & Destructor Documentation**Choice_friction::Choice_friction (int choice)**

Constructor.

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

Parameters

<i>in</i>	<i>choice</i>	integer that correspond to the chosen friction law.
-----------	---------------	---

Definition at line 59 of file choice_friction.cpp.

Choice_friction::~~Choice_friction () [virtual]

Destructor.

Definition at line 135 of file choice_friction.cpp.

5.9.3 Member Function Documentation**void Choice_friction::calc (SCALAR *uold*, SCALAR *hnew*, SCALAR *qnew*)**

Calculates the friction term.

Calls the calculation of the friction law.

Parameters

<i>in</i>	<i>uold</i>	velocity at the previous time (<i>n</i> if you are calculating the <i>n</i> + 1th time step).
<i>in</i>	<i>hnew</i>	water height after the Shallow-Water computation (without friction).
<i>in</i>	<i>qnew</i>	discharge after the Shallow-Water computation (without friction).

Note

The friction only affects the discharge.

Definition at line 87 of file choice_friction.cpp.

SCALAR Choice_friction::get_qmod ()

Gives the discharge modified by the friction term.

Calls the function to get the discharge modified by the friction term.

Returns

[Friction::qmod](#) discharge modified by the friction term.

Definition at line 101 of file choice_friction.cpp.

void Choice_friction::set_c (SCALAR *c*)

Sets the variable [Friction::c](#).

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

Parameters

<i>in</i>	<i>c</i>	value of the friction coefficient.
-----------	----------	------------------------------------

Todo For future evolutions: [Friction::set_c](#) will not be necessary when the friction coefficient will be read as in 2D.

Definition at line 112 of file choice_friction.cpp.

void Choice_friction::set_dt (SCALAR *dt*)

Sets the variable [Friction::dt](#).

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

Parameters

<code>in</code>	<code>dt</code>	value of the time step.
-----------------	-----------------	-------------------------

Definition at line 124 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 [calc](#) (const [VECT](#) &, const [VECT](#) &, const [SCALAR](#))
Performs the computation of the modified water height and the infiltrated volume.
- virtual [~Choice_infiltration](#) ()
Destructor.
- [VECT](#) [get_hmod](#) ()
Gives the value of the modified water height.
- [VECT](#) [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 80 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 59 of file `choice_infiltration.cpp`.

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

Destructor.

Definition at line 109 of file `choice_infiltration.cpp`.

5.10.3 Member Function Documentation**void Choice_infiltration::calc (const [VECT](#) & *h*, const [VECT](#) & *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.
<code>in</code>	<code>dt</code>	time step.

Definition at line 77 of file `choice_infiltration.cpp`.

VECT Choice_infiltration::get_hmod ()

Gives the value of the modified water height.

Returns

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

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

VECT Choice_infiltration::get_Vin ()

Gives the value of the infiltrated volume.

Returns

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

Definition at line 99 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_hu Class Reference

Choice of initialization for `h` and `u`.

```
#include <choice_init_hu.hpp>
```

Public Member Functions

- [Choice_init_hu](#) ([Parameters](#) &)
Constructor.
- void [initialization](#) ([VECT](#) &, [VECT](#) &)
Performs the initialization.
- virtual [~Choice_init_hu](#) ()
Destructor.

5.11.1 Detailed Description

Choice of initialization for `h` and `u`.

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

Definition at line 96 of file `choice_init_hu.hpp`.

5.11.2 Constructor & Destructor Documentation

Choice_init_hu::Choice_init_hu ([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 59 of file `choice_init_hu.cpp`.

Choice_init_hu::~~Choice_init_hu () [virtual]

Destructor.

Definition at line 103 of file `choice_init_hu.cpp`.

5.11.3 Member Function Documentation**void Choice_init_hu::initialization (VECT & *h*, VECT & *u*)**

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>	velocity.

Definition at line 91 of file `choice_init_hu.cpp`.

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

- Headers/libinitializations/[choice_init_hu.hpp](#)
- Sources/libinitializations/[choice_init_hu.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 \(VECT &\)](#)
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 87 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 59 of file choice_init_topo.cpp.

Choice_init_topo::~Choice_init_topo () [virtual]

Destructor.

Definition at line 96 of file choice_init_topo.cpp.

5.12.3 Member Function Documentation**void Choice_init_topo::initialization (VECT & z)**

Performs the initialization.

Calls the initialization of the topography.

Parameters

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

Definition at line 85 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 [calc](#) (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 83 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

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

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

Choice_limiter::~~Choice_limiter () [virtual]

Destructor.

Definition at line 103 of file `choice_limiter.cpp`.

5.13.3 Member Function Documentation**void Choice_limiter::calc (SCALAR a, SCALAR b)**

Calculates the slope limiter.

Calls the calculation of the slope limiter.

Parameters

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

Definition at line 80 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 92 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](#) ([VECT](#), [VECT](#), [VECT](#), int)
Save the current time.
- void [initial](#) ([VECT](#), [VECT](#), [VECT](#))
Saves the initial time.
- void [bound_flux](#) (int, [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))
Saves the flux and the cumulative flux on the boundaries.
- void [check_vol](#) (int, [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))
Saves the infiltrated and rain volumes.
- void [result](#) ([SCALAR](#), [clock_t](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))

Saves global values.

- void **save** (VECT, VECT, VECT)

Saves the final time.

- 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 74 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

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

Definition at line 58 of file choice_output.cpp.

Choice_output::~~Choice_output () [virtual]

Destructor.

Definition at line 159 of file choice_output.cpp.

5.14.3 Member Function Documentation

void Choice_output::bound_flux (int *time*, SCALAR *f_l*, SCALAR *f_r*, SCALAR *cum_f_l*, SCALAR *cum_f_r*)

Saves the flux and the cumulative flux on the boundaries.

Calls the saving of the flux and the cumulative flux on the boundaries.

Parameters

in	<i>time</i>	current time (number of iterations).
in	<i>f_l</i>	flux on the left boundary.
in	<i>f_r</i>	flux on the right boundary.
in	<i>cum_f_l</i>	cumulative flux on the left boundary.
in	<i>cum_f_r</i>	cumulative flux on the right boundary.

Definition at line 100 of file choice_output.cpp.

void Choice_output::check_vol (int *time*, SCALAR *Vol_rain_tot*, SCALAR *Vol_inf_tot*, SCALAR *Vol_of_tot*, 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 (number of iterations).
<i>in</i>	<i>Vol_rain_tot</i>	total rain volume.
<i>in</i>	<i>Vol_inf_tot</i>	total volume of infiltrated water.
<i>in</i>	<i>Vol_of_tot</i>	total volume of overland flow.
<i>in</i>	<i>Vol_bound_tot</i>	total volume of water at the boundary.

Definition at line 115 of file choice_output.cpp.

void Choice_output::initial (VECT *z*, VECT *h*, VECT *u*)

Saves the initial time.

Calls the saving of the initial time.

Parameters

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

Definition at line 87 of file choice_output.cpp.

void Choice_output::result (SCALAR *t*, clock_t *cpu*, SCALAR *froude*, SCALAR *Vol_rain_tot*, SCALAR *Vol_inf_tot*, SCALAR *Vol_of_tot*, SCALAR *Vol_bound_tot*)

Saves global values.

Calls the saving of the global values.

Parameters

<i>in</i>	<i>t</i>	elapsed time.
<i>in</i>	<i>cpu</i>	CPU time.
<i>in</i>	<i>froude</i>	mean Froude number (in space) at the final time.
<i>in</i>	<i>Vol_rain_tot</i>	total rain volume.
<i>in</i>	<i>Vol_inf_tot</i>	total volume of infiltrated water.
<i>in</i>	<i>Vol_of_tot</i>	total volume of overland flow.
<i>in</i>	<i>Vol_bound_tot</i>	total volume of water at the boundary.

Definition at line 130 of file choice_output.cpp.

void Choice_output::save (VECT *z*, VECT *h*, VECT *q*)

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>q</i>	discharge.

Definition at line 146 of file choice_output.cpp.

void Choice_output::write (VECT *h*, VECT *q*, VECT *z*, int *n*)

Saves the current time.

Calls the saving of the current time.

Parameters

<code>in</code>	<code>h</code>	water height.
<code>in</code>	<code>q</code>	discharge.
<code>in</code>	<code>z</code>	topography.
<code>in</code>	<code>n</code>	index of the current time.

Definition at line 73 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](#), [VECT](#) &)
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 83 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

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

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

`Choice_rain::~~Choice_rain () [virtual]`

Destructor.

Definition at line 92 of file `choice_rain.cpp`.

5.15.3 Member Function Documentation

`void Choice_rain::rain_func (SCALAR time, VECT & Tab_rain)`

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

Calls the initialization of the rain.

Parameters

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

Definition at line 81 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](#) &, [VECT](#))
Constructor.
- void [calc](#) ([VECT](#), [VECT](#), [VECT](#), [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &)
- 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 82 of file choice_reconstruction.hpp.

5.16.2 Constructor & Destructor Documentation

Choice_reconstruction::Choice_reconstruction ([Parameters](#) & *par*, [VECT](#) *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>	vector that represents the topography.

Definition at line 59 of file choice_reconstruction.cpp.

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

Destructor.

Definition at line 100 of file choice_reconstruction.cpp.

5.16.3 Member Function Documentation

void Choice_reconstruction::calc ([VECT](#) *h*, [VECT](#) *u*, [VECT](#) *z*, [VECT](#) & *hr*, [VECT](#) & *hl*, [VECT](#) & *delz*, [VECT](#) & *ur*, [VECT](#) & *ul*, [VECT](#) & *dzi*)

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	<i>z</i>	topography.
out	<i>hr</i>	reconstructed water height on the right of the cell.
out	<i>hl</i>	reconstructed water height on the left of the cell.
out	<i>delz</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells).
out	<i>ur</i>	reconstructed velocity on the right of the cell.
out	<i>ul</i>	reconstructed velocity on the left of the cell.
out	<i>dzi</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell.

Definition at line 81 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 [calc](#) ()
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 78 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

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

Definition at line 59 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::calc ()

Performs the scheme.

Calls the computation of the solution.

Definition at line 77 of file choice_scheme.cpp.

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

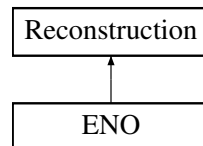
- Headers/libschemas/[choice_scheme.hpp](#)
- Sources/libschemas/[choice_scheme.cpp](#)

5.18 ENO Class Reference

ENO reconstruction

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

Inheritance diagram for ENO:



Public Member Functions

- [ENO \(Parameters &, VECT\)](#)
Constructor.
- void [calc \(VECT, VECT, VECT, VECT &, VECT &, VECT &, VECT &, VECT &, VECT &\)](#)
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 69 of file eno.hpp.

5.18.2 Constructor & Destructor Documentation

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

Constructor.

Initializations.

Parameters

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

in	z	topography.
----	---	-------------

Warning

Problem: allocation of som_z failed.

Note

If som_z cannot be allocated, the code will exit with failure termination code.

Definition at line 59 of file eno.cpp.

ENO::~~ENO ()

Destructor.

Definition at line 200 of file eno.cpp.

5.18.3 Member Function Documentation

void ENO::calc (VECT *h*, VECT *u*, VECT *z*, VECT & *hr*, VECT & *hl*, VECT & *delz*, VECT & *ur*, VECT & *ul*, VECT & *dzi*) [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	<i>z</i>	topography.
out	<i>hr</i>	reconstructed water height on the right of the cell.
out	<i>hl</i>	reconstructed water height on the left of the cell.
out	<i>delz</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells).
out	<i>ur</i>	reconstructed velocity on the right of the cell.
out	<i>ul</i>	reconstructed velocity on the left of the cell.
out	<i>dzi</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell.

Implements [Reconstruction](#).

Definition at line 82 of file eno.cpp.

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

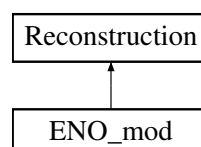
- Headers/libreconstructions/[eno.hpp](#)
- Sources/libreconstructions/[eno.cpp](#)

5.19 ENO_mod Class Reference

Modified ENO reconstruction.

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

Inheritance diagram for ENO_mod:



Public Member Functions

- `ENO_mod` (`Parameters &`, `VECT`)
Constructor.
- `void calc` (`VECT`, `VECT`, `VECT`, `VECT &`, `VECT &`, `VECT &`, `VECT &`, `VECT &`, `VECT &`)
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 69 of file `eno_mod.hpp`.

5.19.2 Constructor & Destructor Documentation

`ENO_mod::ENO_mod (Parameters & par, VECT z)`

Constructor.

Initializations.

Parameters

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

Warning

Problem: allocation of `som_z` failed.

Note

If `som_z` cannot be allocated, the code will exit with failure termination code.

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

`ENO_mod::~~ENO_mod ()`

Destructor.

Definition at line 215 of file `eno_mod.cpp`.

5.19.3 Member Function Documentation

`void ENO_mod::calc (VECT h, VECT u, VECT z, VECT & hr, VECT & hl, VECT & delz, VECT & ur, VECT & ul, VECT & dzi) [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	<i>z</i>	topography.
out	<i>hr</i>	reconstructed water height on the right of the cell.
out	<i>hl</i>	reconstructed water height on the left of the cell.
out	<i>delz</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells).
out	<i>ur</i>	reconstructed velocity on the right of the cell.
out	<i>ul</i>	reconstructed velocity on the left of the cell.
out	<i>dzi</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell.

Implements [Reconstruction](#).

Definition at line 81 of file eno_mod.cpp.

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

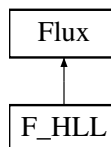
- Headers/libreconstructions/[eno_mod.hpp](#)
- Sources/libreconstructions/[eno_mod.cpp](#)

5.20 F_HLL Class Reference

HLL flux.

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

Inheritance diagram for F_HLL:



Public Member Functions

- [F_HLL \(\)](#)
Constructor.
- void [calc \(SCALAR, SCALAR, SCALAR, SCALAR\)](#)
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 69 of file f_hll.hpp.

5.20.2 Constructor & Destructor Documentation

F_HLL::F_HLL ()

Constructor.

Definition at line 58 of file f_hll.cpp.

F_HLL::~~F_HLL() [virtual]

Destructor.

Definition at line 124 of file f_hll.cpp.

5.20.3 Member Function Documentation

void F_HLL::calc(SCALAR h_L, SCALAR u_L, SCALAR h_R, SCALAR u_R) [virtual]

Calculates the numerical flux.

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, $U = {}^t(h, hu)$ and $F(U) = {}^t(hu, hu^2 + gh^2/2)$.

Parameters

in	h_L	water height at the left of the interface where the flux is calculated.
in	u_L	velocity 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 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::cfl value of the CFL.

Note

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

Todo Check if the use of long double should be generalized to other fluxes.

Implements [Flux](#).

Definition at line 61 of file f_hll.cpp.

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

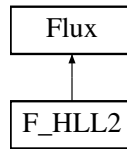
- Headers/libflux/f_hll.hpp
- Sources/libflux/f_hll.cpp

5.21 F_HLL2 Class Reference

HLL flux.

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

Inheritance diagram for F_HLL2:



Public Member Functions

- `F_HLL2()`
Constructor.
- `void calc(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 70 of file `f_hll2.hpp`.

5.21.2 Constructor & Destructor Documentation

`F_HLL2::F_HLL2()`

Constructor.

Definition at line 58 of file `f_hll2.cpp`.

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

Destructor.

Definition at line 106 of file `f_hll2.cpp`.

5.21.3 Member Function Documentation

`void F_HLL2::calc(SCALAR h_L, SCALAR u_L, SCALAR h_R, SCALAR u_R)` **[virtual]**

Calculates the numerical flux.

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

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

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, $U = {}^t(h, hu)$ and $F(U) = {}^t(hu, hu^2 + gh^2/2)$.

Parameters

in	h_L	water height at the left of the interface where the flux is calculated.
in	u_L	velocity 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 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::cfl](#) value of the CFL.

Implements [Flux](#).

Definition at line 61 of file `f_hll2.cpp`.

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

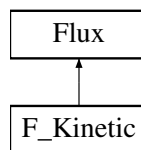
- [Headers/libflux/f_hll2.hpp](#)
- [Sources/libflux/f_hll2.cpp](#)

5.22 F_Kinetic Class Reference

Kinetic flux.

```
#include <f_kinetic.hpp>
```

Inheritance diagram for `F_Kinetic`:

**Public Member Functions**

- [F_Kinetic \(\)](#)
Constructor.
- void [calc \(SCALAR, SCALAR, SCALAR, SCALAR\)](#)
Calculates the numerical flux.
- virtual [~F_Kinetic \(\)](#)
Destructor.

Additional Inherited Members**5.22.1 Detailed Description**

Kinetic flux.

Class that computes the kinetic numerical flux.

Definition at line 70 of file `f_kinetic.hpp`.

5.22.2 Constructor & Destructor Documentation**F_Kinetic::F_Kinetic ()**

Constructor.

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

F_Kinetic::~~F_Kinetic() [virtual]

Destructor.

Definition at line 124 of file f_kinetic.cpp.

5.22.3 Member Function Documentation**void F_Kinetic::calc (SCALAR h_L, SCALAR u_L, SCALAR h_R, SCALAR u_R) [virtual]**

Calculates the numerical flux.

If the water heights on the two sides are small, there is no water. Else, the kinetic formulation is used (see [Audusse et al. \[2000\]](#)):

$$\mathcal{F}(U_L, U_R) = F_+(U_L) + F_-(U_R)$$

$$F_+(U_L) = \frac{h_L}{2\sqrt{3T_L}} \begin{pmatrix} \frac{M_+^2 - M_-^2}{2} \\ \frac{M_+^3 - M_-^3}{3} \end{pmatrix}, \quad F_-(U_R) = \frac{h_R}{2\sqrt{3T_R}} \begin{pmatrix} \frac{m_+^2 - m_-^2}{2} \\ \frac{m_+^3 - m_-^3}{3} \end{pmatrix},$$

where

$$\begin{aligned} M_+ &= \max(0, u_L + \sqrt{3T_L}), & m_+ &= \min(0, u_R + \sqrt{3T_R}), \\ M_- &= \max(0, u_L - \sqrt{3T_L}), & m_- &= \min(0, u_R - \sqrt{3T_R}), \end{aligned}$$

with $T_L = gh_L/2, T_R = gh_R/2, U = {}^t(h, hu)$ and $F(U) = {}^t(hu, hu^2 + gh^2/2)$.

Parameters

in	h_L	water height at the left of the interface where the flux is calculated.
in	u_L	velocity 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 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::cfl** value of the CFL.Implements **Flux**.

Definition at line 64 of file f_kinetic.cpp.

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

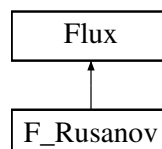
- Headers/libflux/f_kinetic.hpp
- Sources/libflux/f_kinetic.cpp

5.23 F_Rusanov Class Reference

Rusanov flux.

#include <f_rusanov.hpp>

Inheritance diagram for F_Rusanov:



Public Member Functions

- [F_Rusanov](#) ()
Constructor.
- void [calc](#) (SCALAR, SCALAR, SCALAR, SCALAR)
Calculates the numerical flux.
- virtual [~F_Rusanov](#) ()
Destructor.

Additional Inherited Members

5.23.1 Detailed Description

Rusanov flux.

Class that computes Rusanov numerical flux.

Definition at line 69 of file `f_rusanov.hpp`.

5.23.2 Constructor & Destructor Documentation

`F_Rusanov::F_Rusanov ()`

Constructor.

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

`F_Rusanov::~~F_Rusanov () [virtual]`

Destructor.

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

5.23.3 Member Function Documentation

`void F_Rusanov::calc (SCALAR h_L, SCALAR u_L, SCALAR h_R, SCALAR u_R) [virtual]`

Calculates the numerical flux.

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

Parameters

in	<code>h_L</code>	water height at the left of the interface where the flux is calculated.
in	<code>u_L</code>	velocity at the left of the interface where the flux is calculated.
in	<code>h_R</code>	water height at the right of the interface where the flux is calculated.
in	<code>u_R</code>	velocity 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::cfl` value of the CFL.

Implements `Flux`.

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

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

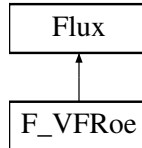
- `Headers/libflux/f_rusanov.hpp`
- `Sources/libflux/f_rusanov.cpp`

5.24 F_VFRoe Class Reference

VFRoe flux.

```
#include <f_vfroe.hpp>
```

Inheritance diagram for F_VFRoe:



Public Member Functions

- [F_VFRoe](#) ()
Constructor.
- void [calc](#) (SCALAR, SCALAR, SCALAR, SCALAR)
Calculates the numerical flux.
- virtual [~F_VFRoe](#) ()
Destructor.

Additional Inherited Members

5.24.1 Detailed Description

VFRoe flux.

Class that computes the VFRoe-ncv numerical flux.

Definition at line 70 of file f_vfroe.hpp.

5.24.2 Constructor & Destructor Documentation

F_VFRoe::F_VFRoe ()

Constructor.

Definition at line 58 of file f_vfroe.cpp.

F_VFRoe::~~F_VFRoe () [virtual]

Destructor.

Definition at line 151 of file f_vfroe.cpp.

5.24.3 Member Function Documentation

void F_VFRoe::calc (SCALAR h_L, SCALAR u_L, SCALAR h_R, SCALAR u_R) [virtual]

Calculates the numerical flux.

The VFRoe-ncv formulation is used (see [Gallouët et al. \[2003\]](#)):

$$\mathcal{F}(U_L, U_R) = \begin{cases} F(U_L) & \text{if } 0 < \lambda_1(\tilde{U}), \\ F(U_*) & \text{if } \lambda_1(\tilde{U}) < 0 < \lambda_2(\tilde{U}), \\ F(U_R) & \text{if } \lambda_2(\tilde{U}) < 0, \end{cases}$$

with $U = \begin{pmatrix} h \\ hu \end{pmatrix}$, $F(U) = \begin{pmatrix} hu \\ hu^2 + gh^2/2 \end{pmatrix}$ and $\lambda_1(U) = u - \sqrt{gh}$, $\lambda_2(U) = u + \sqrt{gh}$ are the eigenvalues of the Shallow Water system. The mean state \tilde{W} is defined by

$$\tilde{W} = \begin{pmatrix} 2\sqrt{g\tilde{h}} \\ \tilde{u} \end{pmatrix} = \begin{pmatrix} 2\tilde{c} \\ \tilde{u} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{gh_L} + \sqrt{gh_R}}{2} \\ \frac{u_L + u_R}{2} \end{pmatrix},$$

and the mean Roe states h_* and u_* are given by $\begin{cases} \sqrt{gh_*} = c_* = \tilde{c} - (u_R - u_L)/4, \\ u_* = \tilde{u} - (\sqrt{gh_R} - \sqrt{gh_L}). \end{cases}$

If the eigenvalues satisfy $\lambda_1(U_L) < 0 < \lambda_1(U_R)$ or $\lambda_2(U_L) < 0 < \lambda_2(U_R)$, we can get non-entropic solutions. Then the entropy is corrected thanks to the Rusanov flux.

Parameters

in	h_L	water height at the left of the interface where the flux is calculated.
in	u_L	velocity 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 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::cfl value of the CFL.

Implements [Flux](#).

Definition at line 74 of file `f_vfroe.cpp`.

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

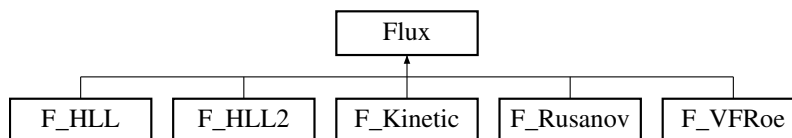
- Headers/libflux/f_vfroe.hpp
- Sources/libflux/f_vfroe.cpp

5.25 Flux Class Reference

Numerical flux.

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

Inheritance diagram for Flux:



Public Member Functions

- [Flux \(\)](#)
Constructor.
- virtual void [calc \(SCALAR, SCALAR, SCALAR, SCALAR\)=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_cfl](#) () const

Gives the CFL value.

- virtual [~Flux](#) ()

Destructor.

Protected Attributes

- [SCALAR f1](#)
- [SCALAR f2](#)
- [SCALAR cfl](#)
- [SCALAR tx](#)

5.25.1 Detailed Description

Numerical flux.

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

Definition at line 71 of file flux.hpp.

5.25.2 Constructor & Destructor Documentation

Flux::Flux ()

Constructor.

Definition at line 58 of file flux.cpp.

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

Destructor.

Definition at line 106 of file flux.cpp.

5.25.3 Member Function Documentation

virtual void Flux::calc (**SCALAR** , **SCALAR** , **SCALAR** , **SCALAR**) [**pure virtual**]

Function to be specified in each numerical flux.

Implemented in [F_HLL2](#), [F_Kinetic](#), [F_VFRoe](#), [F_HLL](#), and [F_Rusanov](#).

SCALAR Flux::get_cfl () const

Gives the CFL value.

Returns

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

Definition at line 96 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.

void Flux::set_tx (SCALAR tx)

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

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

Parameters

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

Definition at line 65 of file flux.cpp.

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

CFL value.

Definition at line 103 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::tx [protected]

Value of dt/dx.

Definition at line 105 of file flux.hpp.

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

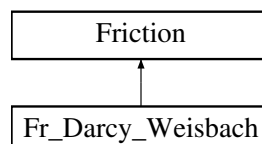
- [Headers/libflux/flux.hpp](#)
- [Sources/libflux/flux.cpp](#)

5.26 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](#) ()
Constructor.
- void [calc](#) (SCALAR, SCALAR, SCALAR)
Calculates the Darcy-Weisbach friction term.
- virtual [~Fr_Darcy_Weisbach](#) ()
Destructor.

Additional Inherited Members

5.26.1 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 70 of file `fr_darcy_weisbach.hpp`.

5.26.2 Constructor & Destructor Documentation

`Fr_Darcy_Weisbach::Fr_Darcy_Weisbach ()`

Constructor.

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

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

Destructor.

Definition at line 81 of file `fr_darcy_weisbach.cpp`.

5.26.3 Member Function Documentation

`void Fr_Darcy_Weisbach::calc (SCALAR uold, SCALAR hnew, SCALAR qnew) [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:

$$q_i^{n+1} = \frac{q_i^*}{1 + dt \frac{f|u_i^n|}{8h_i^{n+1}}}$$

where f is the friction coefficient.

Parameters

in	<i>uold</i>	velocity at the previous time (n if you are calculating the $n + 1$ th time step), denoted by 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>qnew</i>	discharge after the Shallow-Water computation (without friction), denoted by q_i^* in the above formula.

Modifies

`friction::qmod` discharge modified by the friction term.

Note

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

Todo For future evolutions: the friction coefficient should be added as in 2D as a parameter of `Fr_Darcy_↔Weisbach::calc`.

Implements `Friction`.

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

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

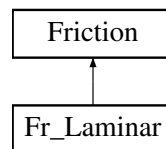
- Headers/libfrictions/`fr_darcy_weisbach.hpp`
- Sources/libfrictions/`fr_darcy_weisbach.cpp`

5.27 Fr_Laminar Class Reference

Laminar law.

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

Inheritance diagram for `Fr_Laminar`:



Public Member Functions

- `Fr_Laminar ()`
Constructor.
- void `calc (SCALAR, SCALAR, SCALAR)`
Calculates the laminar friction term.
- virtual `~Fr_Laminar ()`
Destructor.

Additional Inherited Members

5.27.1 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 70 of file `fr_laminar.hpp`.

5.27.2 Constructor & Destructor Documentation

`Fr_Laminar::Fr_Laminar ()`

Constructor.

Definition at line 57 of file `fr_laminar.cpp`.

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

Destructor.

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

5.27.3 Member Function Documentation

void Fr_Laminar::calc (SCALAR uold, SCALAR hnew, SCALAR qnew) [virtual]

Calculates the laminar friction term.

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

$$q_i^{n+1} = \frac{q_i^*}{1 + vdt \frac{1}{(h_i^{n+1})^2}}$$

where v is the friction coefficient.

Parameters

in	<i>uold</i>	velocity at the previous time (unused).
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>qnew</i>	discharge after the Shallow-Water computation (without friction), denoted by q_i^* in the above formula.

Modifies

friction::qmod discharge modified by the friction term.

Note

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

Todo For future evolutions: the friction coefficient should be added as in 2D as a parameter of `Fr_Laminar::calc`.

Implements [Friction](#).

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

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

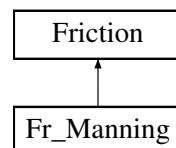
- Headers/libfrictions/[fr_laminar.hpp](#)
- Sources/libfrictions/[fr_laminar.cpp](#)

5.28 Fr_Manning Class Reference

Manning law.

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

Inheritance diagram for `Fr_Manning`:



Public Member Functions

- [Fr_Manning \(\)](#)
Constructor.
- void [calc \(SCALAR, SCALAR, SCALAR\)](#)
Calculates the Manning friction term.
- virtual [~Fr_Manning \(\)](#)
Destructor.

Additional Inherited Members

5.28.1 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 71 of file `fr_manning.hpp`.

5.28.2 Constructor & Destructor Documentation

`Fr_Manning::Fr_Manning ()`

Constructor.

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

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

Destructor.

Definition at line 81 of file `fr_manning.cpp`.

5.28.3 Member Function Documentation

`void Fr_Manning::calc (SCALAR uold, SCALAR hnew, SCALAR qnew) [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:

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

where c is the friction coefficient.

Parameters

in	<i>uold</i>	velocity at the previous time (n if you are calculating the $n + 1$ th time step), denoted by 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>qnew</i>	discharge after the Shallow-Water computation (without friction), denoted by q_i^* in the above formula.

Modifies

`friction::qmod` discharge modified by the friction term.

Note

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

Todo For future evolutions: the friction coefficient should be added as in 2D as a parameter of `Fr_Manning::calc`.

Implements [Friction](#).

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

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

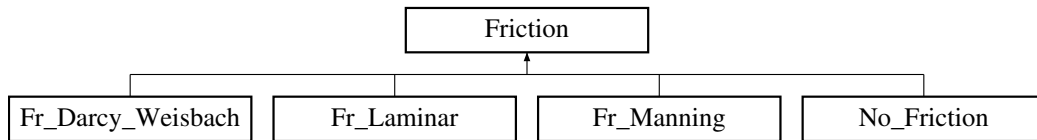
- `Headers/libfrictions/fr_manning.hpp`
- `Sources/libfrictions/fr_manning.cpp`

5.29 Friction Class Reference

Friction law

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

Inheritance diagram for Friction:



Public Member Functions

- [Friction](#) ()
Constructor.
- virtual void [calc](#) (SCALAR, SCALAR, SCALAR)=0
Function to be specified in each friction law.
- [SCALAR get_qmod](#) () const
Gives the discharge modified by the friction term.
- void [set_c](#) (SCALAR)
Sets the friction coefficient [Friction::c](#).
- void [set_dt](#) (SCALAR)
Sets the time step [Friction::dt](#).
- virtual [~Friction](#) ()
Destructor.

Protected Attributes

- [SCALAR qmod](#)
- [SCALAR c](#)
- [SCALAR dt](#)

5.29.1 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 71 of file friction.hpp.

5.29.2 Constructor & Destructor Documentation

[Friction::Friction](#) ()

Constructor.

Definition at line 59 of file friction.cpp.

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

Destructor.

Definition at line 96 of file friction.cpp.

5.29.3 Member Function Documentation

virtual void Friction::calc (SCALAR , SCALAR , SCALAR) [pure virtual]

Function to be specified in each friction law.

Implemented in [Fr_Manning](#), [Fr_Darcy_Weisbach](#), [Fr_Laminar](#), and [No_Friction](#).

SCALAR Friction::get_qmod () const

Gives the discharge modified by the friction term.

Returns

[Friction::qmod](#) discharge modified by the friction term.

Definition at line 62 of file friction.cpp.

void Friction::set_c (SCALAR c)

Sets the friction coefficient [Friction::c](#).

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

Parameters

in	<i>c</i>	value of the friction coefficient.
----	----------	------------------------------------

Todo For future evolutions: [Friction::set_c](#) will not be necessary when the friction coefficient will be read as in 2D.

Definition at line 72 of file friction.cpp.

void Friction::set_dt (SCALAR dt)

Sets the time step [Friction::dt](#).

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

Parameters

in	<i>dt</i>	value of the time step.
----	-----------	-------------------------

Definition at line 84 of file friction.cpp.

5.29.4 Member Data Documentation

SCALAR Friction::c [protected]

[Friction](#) coefficient.

Definition at line 98 of file friction.hpp.

SCALAR Friction::dt [protected]

Time step.

Definition at line 100 of file friction.hpp.

SCALAR Friction::qmod [protected]

Discharge modified by the friction term.

Definition at line 96 of file friction.hpp.

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

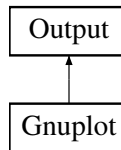
- Headers/libfrictions/[friction.hpp](#)
- Sources/libfrictions/[friction.cpp](#)

5.30 Gnuplot Class Reference

Gnuplot output

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

Inheritance diagram for Gnuplot:



Public Member Functions

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

Additional Inherited Members

5.30.1 Detailed Description

Gnuplot output

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

Definition at line 70 of file gnuplot.hpp.

5.30.2 Constructor & Destructor Documentation

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

Constructor.

Writes the header of the file 'huz_evolution.dat' if [Parameters::nbtimes](#) is (strictly) positive.

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

Definition at line 58 of file gnuplot.cpp.

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

Destructor.

Definition at line 122 of file gnuplot.cpp.

5.30.3 Member Function Documentation

void Gnuplot::write (VECT *h*, VECT *q*, VECT *z*, int *n*) [virtual]

Saves one time step.

Writes the values of [Scheme::h](#), [Scheme::u](#) (=q/h), [Scheme::z](#), [Scheme::q](#), [Scheme::h](#)+ [Scheme::z](#) (free surface), and the Froude number $\frac{q}{h\sqrt{gh}}$ 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

in	<i>h</i>	the water height.
in	<i>q</i>	the discharge.
in	<i>z</i>	the topography.
in	<i>n</i>	the index of the current time step (the time is n Parameters::dt).

Implements [Output](#).

Definition at line 91 of file gnuplot.cpp.

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

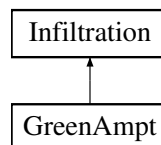
- [Headers/libsave/gnuplot.hpp](#)
- [Sources/libsave/gnuplot.cpp](#)

5.31 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 [calc](#) (const [VECT](#) &, const [VECT](#) &, const [SCALAR](#))
Calculates the infiltrated volume.
- virtual [~GreenAmpt \(\)](#)
Destructor.

Additional Inherited Members

5.31.1 Detailed Description

Green-Ampt law.

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

Definition at line 69 of file greenampt.hpp.

5.31.2 Constructor & Destructor Documentation

GreenAmpt::GreenAmpt (Parameters & *par*)

Constructor.

Initializes the values for Green-Ampt infiltration.

Parameters

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

Warning

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

Definition at line 57 of file greenampt.cpp.

GreenAmpt::~~GreenAmpt () [virtual]

Destructor.

Definition at line 288 of file greenampt.cpp.

5.31.3 Member Function Documentation

void GreenAmpt::calc (const VECT & *h*, const VECT & *Vin_tot*, const SCALAR *dt*) [virtual]

Calculates the infiltrated volume.

Parameters

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

Modifies

[Infiltration::hmod](#) modified water height.

[Infiltration::Vin](#) total infiltrated volume containing the current time step.

Implements [Infiltration](#).

Definition at line 248 of file greenampt.cpp.

SCALAR GreenAmpt::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

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

Parameters

in	<i>h</i>	water height.
in	<i>Vin_tot</i>	total infiltrated volume.
in	<i>dt</i>	time step (unused).
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

lc: infiltration capacity.

Definition at line 207 of file greenampt.cpp.

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

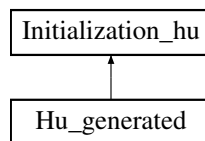
- Headers/librain_infiltration/[greenampt.hpp](#)
- Sources/librain_infiltration/[greenampt.cpp](#)

5.32 Hu_generated Class Reference

No water configuration.

```
#include <hu_generated.hpp>
```

Inheritance diagram for Hu_generated:

**Public Member Functions**

- [Hu_generated](#) ([Parameters](#) &)
Constructor.
- void [initialization](#) ([VECT](#) &, [VECT](#) &)
Performs the initialization.
- virtual [~Hu_generated](#) ()
Destructor.

Additional Inherited Members**5.32.1 Detailed Description**

No water configuration.

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

Definition at line 71 of file hu_generated.hpp.

5.32.2 Constructor & Destructor Documentation

Hu_generated::Hu_generated ([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 hu_generated.cpp.

Hu_generated::~~Hu_generated () [virtual]

Destructor.

Definition at line 81 of file hu_generated.cpp.

5.32.3 Member Function Documentation**void Hu_generated::initialization (VECT & *h*, VECT & *u*) [virtual]**

Performs the initialization.

Initializes the water height and the velocity at 0.

Parameters

<i>in, out</i>	<i>h</i>	water height.
<i>in, out</i>	<i>u</i>	velocity.

Implements [Initialization_hu](#).

Definition at line 66 of file hu_generated.cpp.

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

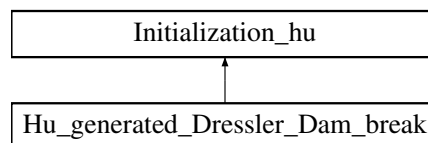
- [Headers/libinitializations/hu_generated.hpp](#)
- [Sources/libinitializations/hu_generated.cpp](#)

5.33 Hu_generated_Dressler_Dam_break Class Reference

Dressler dam break configuration.

```
#include <hu_generated_dressler_dam_break.hpp>
```

Inheritance diagram for Hu_generated_Dressler_Dam_break:

**Public Member Functions**

- [Hu_generated_Dressler_Dam_break](#) (Parameters &)
Constructor.
- void [initialization](#) (VECT &, VECT &)
Performs the initialization.
- virtual [~Hu_generated_Dressler_Dam_break](#) ()
Destructor.

Additional Inherited Members**5.33.1 Detailed Description**

Dressler dam break configuration.

Class that initializes the water height and the velocity for a dam break as studied by Dressler (with friction).

Definition at line 71 of file hu_generated_dressler_dam_break.hpp.

5.33.2 Constructor & Destructor Documentation

Hu_generated_Dressler_Dam_break::Hu_generated_Dressler_Dam_break (Parameters & par)

Constructor.

Defines the position of the dam (half of the domain), the water height before the dam (6 meters), the water height after the dam (0 meters) and the velocity (0 m/s), see [Dressler \[1952\]](#).

Parameters

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

Definition at line 58 of file hu_generated_dressler_dam_break.cpp.

Hu_generated_Dressler_Dam_break::~~Hu_generated_Dressler_Dam_break () [virtual]

Destructor.

Definition at line 92 of file hu_generated_dressler_dam_break.cpp.

5.33.3 Member Function Documentation

void Hu_generated_Dressler_Dam_break::initialization (VECT & h, VECT & u) [virtual]

Performs the initialization.

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

Parameters

in, out	h	water height.
in, out	u	velocity.

Implements [Initialization_hu](#).

Definition at line 73 of file hu_generated_dressler_dam_break.cpp.

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

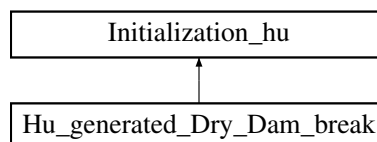
- [Headers/libinitializations/hu_generated_dressler_dam_break.hpp](#)
- [Sources/libinitializations/hu_generated_dressler_dam_break.cpp](#)

5.34 Hu_generated_Dry_Dam_break Class Reference

Dry dam break configuration.

```
#include <hu_generated_dry_dam_break.hpp>
```

Inheritance diagram for Hu_generated_Dry_Dam_break:



Public Member Functions

- [Hu_generated_Dry_Dam_break \(Parameters &\)](#)

Constructor.

- void [initialization \(VECT &, VECT &\)](#)

Performs the initialization.

- virtual [~Hu_generated_Dry_Dam_break \(\)](#)

Destructor.

Additional Inherited Members

5.34.1 Detailed Description

Dry dam break configuration.

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

Definition at line 71 of file hu_generated_dry_dam_break.hpp.

5.34.2 Constructor & Destructor Documentation

Hu_generated_Dry_Dam_break::Hu_generated_Dry_Dam_break (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 meters) and the velocity (0 m/s), see [Goutal and Maurel \[1997\]](#), [Audusse et al. \[2000\]](#).

Parameters

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

Definition at line 58 of file hu_generated_dry_dam_break.cpp.

Hu_generated_Dry_Dam_break::~~Hu_generated_Dry_Dam_break () [virtual]

Destructor.

Definition at line 92 of file hu_generated_dry_dam_break.cpp.

5.34.3 Member Function Documentation

void Hu_generated_Dry_Dam_break::initialization (VECT & *h*, VECT & *u*) [virtual]

Performs the initialization.

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

Parameters

<i>in, out</i>	<i>h</i>	water height.
<i>in, out</i>	<i>u</i>	velocity.

Implements [Initialization_hu](#).

Definition at line 73 of file hu_generated_dry_dam_break.cpp.

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

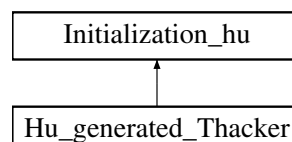
- [Headers/libinitializations/hu_generated_dry_dam_break.hpp](#)
- [Sources/libinitializations/hu_generated_dry_dam_break.cpp](#)

5.35 Hu_generated_Thacker Class Reference

Thacker configuration.

```
#include <hu_generated_thacker.hpp>
```

Inheritance diagram for Hu_generated_Thacker:



Public Member Functions

- [Hu_generated_Thacker](#) ([Parameters](#) &)
Constructor.
- void [initialization](#) ([VECT](#) &, [VECT](#) &)
Performs the initialization.
- virtual [~Hu_generated_Thacker](#) ()
Destructor.

Additional Inherited Members

5.35.1 Detailed Description

Thacker configuration.

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

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

5.35.2 Constructor & Destructor Documentation

`Hu_generated_Thacker::Hu_generated_Thacker (Parameters & par)`

Constructor.

Defines the position of the wet domain (between `x1` and `x2`) and the value of the topography at the center of the domain (`-h0`).

Parameters

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

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

`Hu_generated_Thacker::~Hu_generated_Thacker () [virtual]`

Destructor.

Definition at line 101 of file `hu_generated_thacker.cpp`.

5.35.3 Member Function Documentation

`void Hu_generated_Thacker::initialization (VECT & h, VECT & u) [virtual]`

Performs the initialization.

Initializes the water height in order to have a plane surface between `x1` and `x2`, and the velocity is null, see [Thacker \[1981\]](#).

Parameters

<code>in, out</code>	<code>h</code>	water height.
<code>in, out</code>	<code>u</code>	velocity.

Implements [Initialization_hu](#).

Definition at line 75 of file `hu_generated_thacker.cpp`.

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

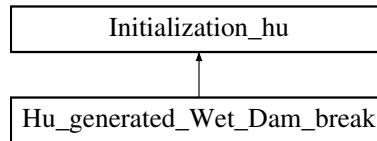
- [Headers/libinitializations/hu_generated_thacker.hpp](#)
- [Sources/libinitializations/hu_generated_thacker.cpp](#)

5.36 Hu_generated_Wet_Dam_break Class Reference

Wet dam break configuration.

```
#include <hu_generated_wet_dam_break.hpp>
```

Inheritance diagram for Hu_generated_Wet_Dam_break:



Public Member Functions

- [Hu_generated_Wet_Dam_break](#) ([Parameters](#) &)
Constructor.
- void [initialization](#) ([VECT](#) &, [VECT](#) &)
Performs the initialization.
- virtual [~Hu_generated_Wet_Dam_break](#) ()
Destructor.

Additional Inherited Members

5.36.1 Detailed Description

Wet dam break configuration.

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

Definition at line 70 of file `hu_generated_wet_dam_break.hpp`.

5.36.2 Constructor & Destructor Documentation

Hu_generated_Wet_Dam_break::Hu_generated_Wet_Dam_break ([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 (1 millimeter) and the velocity (0 m/s), see [Goutal and Maurel \[1997\]](#).

Parameters

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

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

Hu_generated_Wet_Dam_break::~~Hu_generated_Wet_Dam_break () [virtual]

Destructor.

Definition at line 92 of file `hu_generated_wet_dam_break.cpp`.

5.36.3 Member Function Documentation

void Hu_generated_Wet_Dam_break::initialization ([VECT](#) & *h*, [VECT](#) & *u*) [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>	velocity.

Implements [Initialization_hu](#).

Definition at line 73 of file `hu_generated_wet_dam_break.cpp`.

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

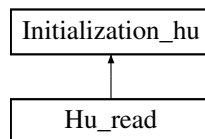
- [Headers/libinitializations/hu_generated_wet_dam_break.hpp](#)
- [Sources/libinitializations/hu_generated_wet_dam_break.cpp](#)

5.37 Hu_read Class Reference

File configuration.

```
#include <hu_read.hpp>
```

Inheritance diagram for `Hu_read`:

**Public Member Functions**

- [Hu_read \(Parameters &\)](#)
Constructor.
- void [initialization \(VECT &, VECT &\)](#)
Performs the initialization.
- virtual [~Hu_read \(\)](#)
Destructor.

Additional Inherited Members**5.37.1 Detailed Description**

File configuration.

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

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

5.37.2 Constructor & Destructor Documentation**Hu_read::Hu_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 59 of file `hu_read.cpp`.

Hu_read::~~Hu_read () [virtual]

Destructor.

Definition at line 177 of file `hu_read.cpp`.

5.37.3 Member Function Documentation

void Hu_read::initialization (VECT & *h*, VECT & *u*) [virtual]

Performs the initialization.

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

Parameters

in, out	<i>h</i>	water height.
in, out	<i>u</i>	velocity.

Warning

(hu_namefile): ERROR: cannot open the hu file.
 (hu_namefile): ERROR: the number of data in this file is too big
 (hu_namefile): ERROR: line ***.
 (hu_namefile): WARNING: line ***.
 (hu_namefile): ERROR: the number of data in this file is too small
 (hu_namefile): ERROR: the value for the point x *** 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_hu](#).

Definition at line 70 of file hu_read.cpp.

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

- Headers/libinitializations/[hu_read.hpp](#)
- Sources/libinitializations/[hu_read.cpp](#)

5.38 Hydrostatic Class Reference

Hydrostatic reconstruction

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

Public Member Functions

- [Hydrostatic](#) ()
Constructor.
- void [calc](#) (SCALAR, SCALAR, SCALAR)
Calculates the hydrostatic reconstruction.
- [SCALAR get_hhydro_l](#) () const
Gives the reconstructed water height on the left.
- [SCALAR get_hhydro_r](#) () const
Gives the reconstructed water height on the right.
- virtual [~Hydrostatic](#) ()
Destructor.

Protected Attributes

- [SCALAR hl_rec](#)
- [SCALAR hr_rec](#)

5.38.1 Detailed Description

Hydrostatic reconstruction

Class that computes the hydrostatic reconstruction.

Definition at line 68 of file hydrostatic.hpp.

5.38.2 Constructor & Destructor Documentation

Hydrostatic::Hydrostatic ()

Constructor.

Definition at line 57 of file hydrostatic.cpp.

Hydrostatic::~Hydrostatic () [virtual]

Destructor.

Definition at line 97 of file hydrostatic.cpp.

5.38.3 Member Function Documentation

void Hydrostatic::calc (SCALAR *hl_0*, SCALAR *hr_0*, SCALAR *dz*)

Calculates the hydrostatic reconstruction.

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

Parameters

in	<i>hl_0</i>	water height on the cell located at the left of the boundary.
in	<i>hr_0</i>	water height on the cell located at the right of the boundary.
in	<i>dz</i>	Difference between the values of the topography of the two adjacent cells.

Modifies

[Hydrostatic::hl_rec](#), set to $(hl_0 - \max(0, dz))_+$.

[Hydrostatic::hr_rec](#), set to $(hr_0 - \max(0, -dz))_+$.

Definition at line 60 of file hydrostatic.cpp.

SCALAR Hydrostatic::get_hhydro_l () const

Gives the reconstructed water height on the left.

Returns

[Hydrostatic::hl_rec](#) Hydrostatic reconstruction on the left.

Definition at line 77 of file hydrostatic.cpp.

SCALAR Hydrostatic::get_hhydro_r () const

Gives the reconstructed water height on the right.

Returns

[Hydrostatic::hr_rec](#) Hydrostatic reconstruction on the right.

Definition at line 87 of file hydrostatic.cpp.

5.38.4 Member Data Documentation

SCALAR Hydrostatic::hl_rec [protected]

Hydrostatic reconstruction on the left

Definition at line 88 of file hydrostatic.hpp.

SCALAR Hydrostatic::hr_rec [protected]

Hydrostatic reconstruction on the right

Definition at line 90 of file hydrostatic.hpp.

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

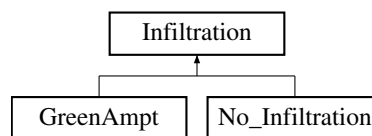
- Headers/libreconstructions/[hydrostatic.hpp](#)
- Sources/libreconstructions/[hydrostatic.cpp](#)

5.39 Infiltration Class Reference

Definition of infiltration law.

```
#include <infiltration.hpp>
```

Inheritance diagram for Infiltration:



Public Member Functions

- [Infiltration](#) ([Parameters](#) &)
Constructor.
- virtual void [calc](#) (const [VECT](#) &, const [VECT](#) &, const [SCALAR](#))=0
Function to be specified in each case.
- [VECT](#) [get_hmod](#) () const
Gives the modified valued of the water height.
- [VECT](#) [get_Vin](#) () const
Gives the infiltrated volume.
- virtual [~Infiltration](#) ()
Destructor.

Protected Attributes

- const int [NXCELL](#)
- const [SCALAR](#) [DX](#)
- [VECT](#) [hmod](#)
- [VECT](#) [Vin](#)

5.39.1 Detailed Description

Definition of infiltration law.

Class that contains all the common declarations for the infiltration law.

Definition at line 69 of file infiltration.hpp.

5.39.2 Constructor & Destructor Documentation

Infiltration::Infiltration (Parameters & *par*)

Constructor.

Defines the number of cells, the space step, and initializes [Infiltration::hmod](#) and [Infiltration::Vin](#).

Parameters

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

Definition at line 57 of file infiltration.cpp.

Infiltration::~~Infiltration () [virtual]

Destructor.

Definition at line 95 of file infiltration.cpp.

5.39.3 Member Function Documentation

virtual void Infiltration::calc (const VECT & , const VECT & , const SCALAR) [pure virtual]

Function to be specified in each case.

Implemented in [GreenAmpt](#), and [No_Infiltration](#).

VECT Infiltration::get_hmod () const

Gives the modified valued of the water height.

Returns

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

Definition at line 75 of file infiltration.cpp.

VECT Infiltration::get_Vin () const

Gives the infiltrated volume.

Returns

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

Definition at line 85 of file infiltration.cpp.

5.39.4 Member Data Documentation

const SCALAR Infiltration::DX [protected]

Space step.

Definition at line 91 of file infiltration.hpp.

VECT Infiltration::hmod [protected]

Modified valued of the water height

Definition at line 93 of file infiltration.hpp.

const int Infiltration::NXCELL [protected]

Number of cells in space.

Definition at line 89 of file infiltration.hpp.

VECT Infiltration::Vin [protected]

Infiltrated volume

Definition at line 95 of file infiltration.hpp.

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

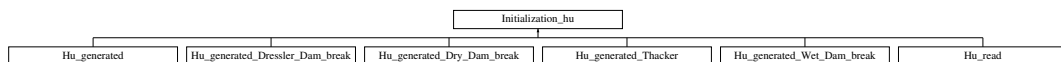
- Headers/librain_infiltration/infiltration.hpp
- Sources/librain_infiltration/infiltration.cpp

5.40 Initialization_hu Class Reference

Initialization of h and u.

```
#include <initialization_hu.hpp>
```

Inheritance diagram for Initialization_hu:

**Public Member Functions**

- [Initialization_hu \(Parameters &\)](#)
Constructor.
- virtual void [initialization \(VECT &, VECT &\)=0](#)
Function to be specified in each initialization.
- virtual [~Initialization_hu \(\)](#)
Destructor.

Protected Attributes

- const int [NXCELL](#)
- const int [M](#)
- const [SCALAR DX](#)

5.40.1 Detailed Description

Initialization of h and u.

Class that contains all the common declarations for the initialization of the water height and of the velocity.

Definition at line 69 of file initialization_hu.hpp.

5.40.2 Constructor & Destructor Documentation**Initialization_hu::Initialization_hu (Parameters & par)**

Constructor.

Defines the number of cells, of time steps and the space step.

Parameters

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

Definition at line 58 of file initialization_hu.cpp.

Initialization_hu::~~Initialization_hu () [virtual]

Destructor.

Definition at line 69 of file initialization_hu.cpp.

5.40.3 Member Function Documentation

virtual void Initialization_hu::initialization (VECT &, VECT &) [pure virtual]

Function to be specified in each initialization.

Implemented in [Hu_generated_Dressler_Dam_break](#), [Hu_generated_Dry_Dam_break](#), [Hu_generated_Thacker](#), [Hu_read](#), [Hu_generated](#), and [Hu_generated_Wet_Dam_break](#).

5.40.4 Member Data Documentation

const SCALAR Initialization_hu::DX [protected]

Space step.

Definition at line 88 of file initialization_hu.hpp.

const int Initialization_hu::M [protected]

Number of time steps.

Definition at line 86 of file initialization_hu.hpp.

const int Initialization_hu::NXCELL [protected]

Number of cells in space.

Definition at line 84 of file initialization_hu.hpp.

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

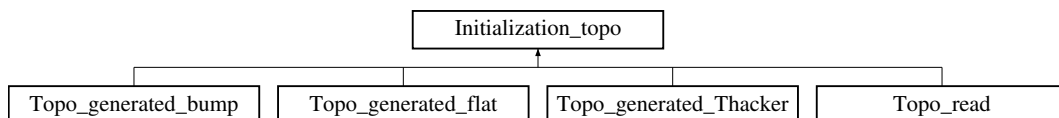
- Headers/libinitializations/[initialization_hu.hpp](#)
- Sources/libinitializations/[initialization_hu.cpp](#)

5.41 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](#) (VECT &)=0
Function to be specified in each initialization.
- virtual [~Initialization_topo](#) ()
Destructor.

Protected Attributes

- [VECT z](#)
- const int [NXCELL](#)
- const int [M](#)
- const [SCALAR DX](#)

5.41.1 Detailed Description

Initialization of z.

Class that contains all the common declarations for the initialization of the topography.

Definition at line 69 of file initialization_topo.hpp.

5.41.2 Constructor & Destructor Documentation

Initialization_topo::Initialization_topo (Parameters & *par*)

Constructor.

Defines the number of cells, of time steps and the space step. Creates z the vector for the topography.

Parameters

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

Warning

Problem: allocation of z failed.

Note

If the vector cannot be allocated, the code will exit with failure termination code.

Definition at line 58 of file initialization_topo.cpp.

Initialization_topo::~~Initialization_topo () [virtual]

Destructor.

Definition at line 77 of file initialization_topo.cpp.

5.41.3 Member Function Documentation

virtual void Initialization_topo::initialization (VECT &) [pure virtual]

Function to be specified in each initialization.

Implemented in [Topo_read](#), [Topo_generated_flat](#), [Topo_generated_Thacker](#), and [Topo_generated_bump](#).

5.41.4 Member Data Documentation

const SCALAR Initialization_topo::DX [protected]

Space step.

Definition at line 91 of file initialization_topo.hpp.

const int Initialization_topo::M [protected]

Number of time steps.

Definition at line 89 of file initialization_topo.hpp.

const int Initialization_topo::NXCELL [protected]

Number of cells in space.

Definition at line 87 of file initialization_topo.hpp.

VECT Initialization_topo::z [protected]

Vector for the topography.

Definition at line 85 of file initialization_topo.hpp.

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

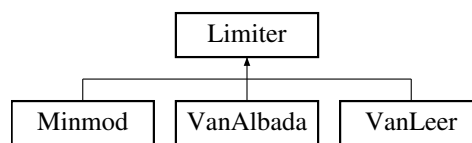
- Headers/libinitializations/[initialization_topo.hpp](#)
- Sources/libinitializations/[initialization_topo.cpp](#)

5.42 Limiter Class Reference

Slope limiter.

```
#include <limiter.hpp>
```

Inheritance diagram for Limiter:

**Public Member Functions**

- [Limiter](#) ()
Constructor.
- virtual void [calc](#) (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](#)

5.42.1 Detailed Description

Slope limiter.

Class that contains all the common declarations for the slope limiters.

Definition at line 69 of file limiter.hpp.

5.42.2 Constructor & Destructor Documentation**Limiter::Limiter ()**

Constructor.

Definition at line 58 of file limiter.cpp.

Limiter::~~Limiter () [virtual]

Destructor.

Definition at line 72 of file limiter.cpp.

5.42.3 Member Function Documentation

virtual void Limiter::calc (SCALAR , SCALAR) [pure virtual]

Function to be specified in each slope limiter.

Implemented in [VanLeer](#), [Minmod](#), and [VanAlbada](#).

SCALAR Limiter::get_rec () const

Gives the reconstructed value.

Returns

[Limiter::rec](#) reconstructed value.

Definition at line 62 of file limiter.cpp.

5.42.4 Member Data Documentation

SCALAR Limiter::rec [protected]

Reconstructed value

Definition at line 88 of file limiter.hpp.

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

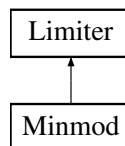
- [Headers/liblimitations/limiter.hpp](#)
- [Sources/liblimitations/limiter.cpp](#)

5.43 Minmod Class Reference

Minmod slope limiter

```
#include <minmod.hpp>
```

Inheritance diagram for Minmod:



Public Member Functions

- [Minmod \(\)](#)
Constructor.
- void [calc \(SCALAR, SCALAR\)](#)
Calculates the value of the slope limiter.
- virtual [~Minmod \(\)](#)
Destructor.

Additional Inherited Members

5.43.1 Detailed Description

Minmod slope limiter

Class that calculates the minmod slope limiter.

Definition at line 69 of file minmod.hpp.

5.43.2 Constructor & Destructor Documentation

Minmod::Minmod ()

Constructor.

Definition at line 58 of file minmod.cpp.

Minmod::~Minmod () [virtual]

Destructor.

Definition at line 87 of file minmod.cpp.

5.43.3 Member Function Documentation

void Minmod::calc (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 61 of file minmod.cpp.

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

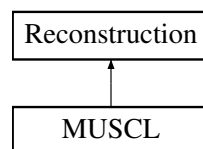
- [Headers/liblimitations/minmod.hpp](#)
- [Sources/liblimitations/minmod.cpp](#)

5.44 MUSCL Class Reference

MUSCL reconstruction

```
#include <muscl.hpp>
```

Inheritance diagram for MUSCL:



Public Member Functions

- [MUSCL](#) ([Parameters](#) &, [VECT](#))
Constructor.
- void [calc](#) ([VECT](#), [VECT](#), [VECT](#), [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &)
- [~MUSCL](#) ()
Destructor.

Additional Inherited Members

5.44.1 Detailed Description

MUSCL reconstruction

Class that computes MUSCL reconstruction in space.

Definition at line 70 of file muscl.hpp.

5.44.2 Constructor & Destructor Documentation

MUSCL::MUSCL (Parameters & *par*, VECT *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 58 of file muscl.cpp.

MUSCL::~~MUSCL ()

Destructor.

Definition at line 139 of file muscl.cpp.

5.44.3 Member Function Documentation

void MUSCL::calc (VECT *h*, VECT *u*, VECT *z*, VECT & *hr*, VECT & *hl*, VECT & *delz*, VECT & *ur*, VECT & *ul*, VECT & *dzi*) [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	<i>z</i>	topography.
out	<i>hr</i>	reconstructed water height on the right of the cell.
out	<i>hl</i>	reconstructed water height on the left of the cell.
out	<i>delz</i>	difference between two reconstructed topographies on the same boundary (from two adjacent cells).
out	<i>ur</i>	reconstructed velocity on the right of the cell.
out	<i>ul</i>	reconstructed velocity on the left of the cell.
out	<i>dzi</i>	difference between the reconstructed topographies on the left and on the right boundary of a cell.

Note

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

Todo Check if the use of long double should be generalized to other reconstructions.

Implements [Reconstruction](#).

Definition at line 68 of file muscl.cpp.

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

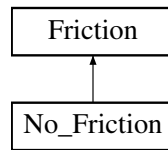
- Headers/libreconstructions/[muscl.hpp](#)
- Sources/libreconstructions/[muscl.cpp](#)

5.45 No_Friction Class Reference

No friction.

```
#include <no_friction.hpp>
```

Inheritance diagram for No_Friction:



Public Member Functions

- [No_Friction \(\)](#)
Constructor.
- void [calc \(SCALAR, SCALAR, SCALAR\)](#)
Does no calculation.
- virtual [~No_Friction \(\)](#)
Destructor.

Additional Inherited Members

5.45.1 Detailed Description

No friction.

Does no computation.

Definition at line 69 of file no_friction.hpp.

5.45.2 Constructor & Destructor Documentation

No_Friction::No_Friction ()

Constructor.

Definition at line 57 of file no_friction.cpp.

No_Friction::~~No_Friction () [virtual]

Destructor.

Definition at line 79 of file no_friction.cpp.

5.45.3 Member Function Documentation

void No_Friction::calc (SCALAR *uold*, SCALAR *hnew*, SCALAR *qnew*) [virtual]

Does no calculation.

No computation (no friction).

Parameters

<i>in</i>	<i>uold</i>	velocity at the previous time (unused).
<i>in</i>	<i>hnew</i>	water height after the Shallow-Water computation (without friction) (unused).

<code>in</code>	<code>qnew</code>	discharge after the Shallow-Water computation (without friction).
-----------------	-------------------	---

Modifies

[Friction::qmod](#) discharge modified by the friction term.

Todo For future evolutions: the friction coefficient should be added as in 2D as a parameter of [Fr_Manning::calc](#).

Implements [Friction](#).

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

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

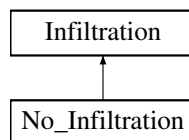
- [Headers/libfrictions/no_friction.hpp](#)
- [Sources/libfrictions/no_friction.cpp](#)

5.46 No_Infiltration Class Reference

No infiltration.

```
#include <no_infiltration.hpp>
```

Inheritance diagram for No_Infiltration:



Public Member Functions

- [No_Infiltration](#) ([Parameters](#) &)
Constructor.
- void [calc](#) (const [VECT](#) &, const [VECT](#) &, const [SCALAR](#))
Does no infiltration.
- virtual [~No_Infiltration](#) ()
Destructor.

Additional Inherited Members

5.46.1 Detailed Description

No infiltration.

The water height and infiltrated volume remain unchanged.

Definition at line 68 of file `no_infiltration.hpp`.

5.46.2 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 57 of file `no_infiltration.cpp`.

No_Infiltration::~~No_Infiltration () [virtual]

Destructor.

Definition at line 84 of file `no_infiltration.cpp`.

5.46.3 Member Function Documentation

void No_Infiltration::calc (const VECT & *h*, const VECT & *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 64 of file `no_infiltration.cpp`.

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

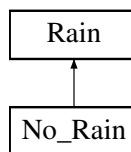
- [Headers/librain_infiltration/no_infiltration.hpp](#)
- [Sources/librain_infiltration/no_infiltration.cpp](#)

5.47 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 \(SCALAR, VECT &\)](#)
Sets the rain intensity to zero.
- virtual [~No_Rain \(\)](#)
Destructor.

Additional Inherited Members

5.47.1 Detailed Description

No rain.

Sets the rain intensity to zero.

Definition at line 69 of file no_rain.hpp.

5.47.2 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 57 of file no_rain.cpp.

No_Rain::~~No_Rain () [virtual]

Destructor.

Definition at line 82 of file no_rain.cpp.

5.47.3 Member Function Documentation

void No_Rain::rain_func (SCALAR *time*, VECT & *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 66 of file no_rain.cpp.

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

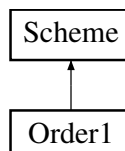
- [Headers/librain_infiltration/no_rain.hpp](#)
- [Sources/librain_infiltration/no_rain.cpp](#)

5.48 Order1 Class Reference

Order 1 scheme.

```
#include <order1.hpp>
```

Inheritance diagram for Order1:



Public Member Functions

- [Order1](#) ([Parameters](#) &)
Constructor.
- void [calc](#) ()
Performs the numerical scheme.
- virtual [~Order1](#) ()
Destructor.

Additional Inherited Members

5.48.1 Detailed Description

Order 1 scheme.

Class that computes the solution with a numerical scheme at order 1.

Definition at line 69 of file order1.hpp.

5.48.2 Constructor & Destructor Documentation

Order1::Order1 ([Parameters](#) & *par*)

Constructor.

Initialization of [Scheme::delta_z](#).

Parameters

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

Definition at line 58 of file order1.cpp.

Order1::~~Order1 () [[virtual](#)]

Destructor.

Definition at line 166 of file order1.cpp.

5.48.3 Member Function Documentation

void Order1::calc () [[virtual](#)]

Performs the numerical scheme.

Performs the first order numerical scheme.

Note

In DEBUG mode, the programme will save two other files with boundary fluxes and volumes of water.

Implements [Scheme](#).

Definition at line 71 of file order1.cpp.

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

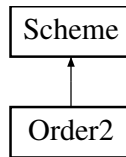
- [Headers/lib schemes/order1.hpp](#)
- [Sources/lib schemes/order1.cpp](#)

5.49 Order2 Class Reference

Order 2 scheme.

```
#include <order2.hpp>
```

Inheritance diagram for Order2:



Public Member Functions

- [Order2 \(Parameters &\)](#)
Constructor.
- void [calc \(\)](#)
Performs the numerical scheme.
- virtual [~Order2 \(\)](#)
Destructor.

Additional Inherited Members

5.49.1 Detailed Description

Order 2 scheme.

Class that computes the solution with a numerical scheme at order 2.

Definition at line 69 of file order2.hpp.

5.49.2 Constructor & Destructor Documentation

Order2::Order2 (Parameters & *par*)

Constructor.

Initializations, definition of the reconstruction and creation 3 vectors for this reconstruction.

Parameters

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

Warning

Problem: allocation of *sa failed.

Note

If a vector cannot be allocated, the code will exit with failure termination code.

Definition at line 59 of file order2.cpp.

Order2::~~Order2 () [virtual]

Destructor.

Definition at line 223 of file order2.cpp.

5.49.3 Member Function Documentation

void Order2::calc() [virtual]

Performs the numerical scheme.

Performs the second order numerical scheme.

Note

In DEBUG mode, the program will save two other files with boundary fluxes and volumes of water.

Implements [Scheme](#).

Definition at line 114 of file order2.cpp.

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

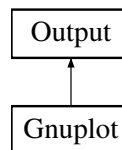
- Headers/libschemas/[order2.hpp](#)
- Sources/libschemas/[order2.cpp](#)

5.50 Output Class Reference

Output format

```
#include <output.hpp>
```

Inheritance diagram for Output:



Public Member Functions

- [Output](#) ([Parameters](#) &)
Constructor.
- virtual void [write](#) ([VECT](#), [VECT](#), [VECT](#), int)=0
Function to be specified in each output format.
- void [initial](#) ([VECT](#), [VECT](#), [VECT](#))
Saves the initial time.
- void [result](#) ([SCALAR](#), clock_t, [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))
Saves global values.
- void [bound_flux](#) (int, [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))
Saves the flux and the cumulative flux on the boundaries.
- void [check_vol](#) (int, [SCALAR](#), [SCALAR](#), [SCALAR](#), [SCALAR](#))
Saves the infiltrated and rain volumes.
- void [save](#) ([VECT](#), [VECT](#), [VECT](#))
Saves the final time.
- virtual [~Output](#) ()
Destructor.

Protected Attributes

- const int [NXCELL](#)
- const int [M](#)
- const [SCALAR DX](#)
- const [SCALAR DT](#)
- const int [NBTIMES](#)
- int [nbt](#)
- string [outputDirectory](#)
- string [namefile_init](#)
- string [namefile_end](#)
- string [namefile_res](#)
- string [namefile_flux](#)
- string [namefile_check_volume](#)

5.50.1 Detailed Description

Output format

Class that contains all the common declarations for the output formats.

Definition at line 68 of file output.hpp.

5.50.2 Constructor & Destructor Documentation

Output::Output (Parameters & *par*)

Constructor.

Defines the names of the outputs.

If run in DEBUG mode, writes the header of the files 'boundary_flux.dat' and 'check_vol.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 boundary_flux.dat or check_vol.dat cannot be opened, the code will exit with failure termination code.

Definition at line 57 of file output.cpp.

Output::~~Output () [virtual]

Destructor.

Definition at line 248 of file output.cpp.

5.50.3 Member Function Documentation

void Output::bound_flux (int *time*, SCALAR *f_l*, SCALAR *f_r*, SCALAR *cum_f_l*, SCALAR *cum_f_r*)

Saves the flux and the cumulative flux on the boundaries.

Parameters

in	<i>time</i>	current time (number of iterations).
in	<i>f_l</i>	flux on the left boundary (m ² /s).
in	<i>f_r</i>	flux on the right boundary (m ² /s).
in	<i>cum_f_l</i>	cumulative flux on the left boundary (m ²).
in	<i>cum_f_r</i>	cumulative flux on the right boundary (m ²).

Definition at line 140 of file output.cpp.

void Output::check_vol (int *time*, SCALAR *Vol_rain_tot*, SCALAR *Vol_inf_tot*, SCALAR *Vol_of_tot*, SCALAR *Vol_bound_tot*)

Saves the infiltrated and rain volumes.

Parameters

in	<i>time</i>	current time (number of iterations).
in	<i>Vol_rain_tot</i>	total rain volume.
in	<i>Vol_inf_tot</i>	total volume of infiltrated water.
in	<i>Vol_of_tot</i>	total volume of overland flow.
in	<i>Vol_bound_tot</i>	total outflow volume at the boundary.

Definition at line 190 of file output.cpp.

void Output::initial (VECT *z*, VECT *h*, VECT *u*)

Saves the initial time.

Parameters

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

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 112 of file output.cpp.

void Output::result (SCALAR *t*, clock_t *cpu*, SCALAR *froude*, SCALAR *Vol_rain_tot*, SCALAR *Vol_inf_tot*, SCALAR *Vol_of_tot*, SCALAR *Vol_bound_tot*)

Saves global values.

Parameters

in	<i>t</i>	elapsed time.
in	<i>cpu</i>	CPU time.
in	<i>froude</i>	mean Froude number (in space) at the final time.
in	<i>Vol_rain_tot</i>	total rain volume.

in	<i>Vol_inf_tot</i>	total volume of infiltrated water.
in	<i>Vol_of_tot</i>	total volume of overland flow.
in	<i>Vol_bound_tot</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 208 of file output.cpp.

void Output::save (VECT z, VECT h, VECT q)

Saves the final time.

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

Parameters

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

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 156 of file output.cpp.

virtual void Output::write (VECT, VECT, VECT, int) [pure virtual]

Function to be specified in each output format.

Implemented in [Gnuplot](#).

5.50.4 Member Data Documentation**const SCALAR Output::DT [protected]**

Time step.

Definition at line 105 of file output.hpp.

const SCALAR Output::DX [protected]

Space step.

Definition at line 103 of file output.hpp.

const int Output::M [protected]

Number of time steps.

Definition at line 101 of file output.hpp.

string Output::namefile_check_volume [protected]

Name of the file where the infiltrated and rain volumes are saved.

Definition at line 121 of file output.hpp.

string Output::namefile_end [protected]

Name of the file where the final time is saved.

Definition at line 115 of file output.hpp.

string Output::namefile_flux [protected]

Name of the file where the boundary fluxes are saved.

Definition at line 119 of file output.hpp.

string Output::namefile_init [protected]

Name of the file where the initialization is saved.

Definition at line 113 of file output.hpp.

string Output::namefile_res [protected]

Name of the file where the global results are saved.

Definition at line 117 of file output.hpp.

int Output::nbt [protected]

Number of times steps between two savings.

Definition at line 109 of file output.hpp.

const int Output::NBTIMES [protected]

Number of times saved.

Definition at line 107 of file output.hpp.

const int Output::NXCELL [protected]

Number of cells in space.

Definition at line 99 of file output.hpp.

string Output::outputDirectory [protected]

Name of the output directory.

Definition at line 111 of file output.hpp.

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

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

5.51 Parameters Class Reference

Gets parameters.

```
#include <parameters.hpp>
```

Public Member Functions

- [Parameters](#) ()
Constructor.
- virtual [~Parameters](#) ()
Destructor.
- void [setparameters](#) (const char *)
Sets the parameters.
- int [get_Nxcell](#) () const
Gives the number of cells in space.
- int [get_m](#) () const
Gives the number of time steps to attain the final time.
- int [get_nbtimes](#) () const
Gives the number of times saved.
- SCALAR [get_dx](#) () const
Gives the space step.
- SCALAR [get_dt](#) () const
Gives the time step.
- SCALAR [get_tx](#) () const
Gives the value of dt/dx.
- int [get_Lbound](#) () const
Gives the value corresponding to the left boundary condition.
- SCALAR [get_L_imp_q](#) () const
Gives the value of the imposed discharge in left bc.
- SCALAR [get_L_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_R_imp_q](#) () const
Gives the value of the imposed discharge in right bc.
- SCALAR [get_R_imp_h](#) () const
Gives the value of the imposed water height in right bc.
- int [get_flux](#) () const
Gives the value corresponding to the flux.
- int [get_order](#) () const
Gives the order of the scheme.
- SCALAR [get_cfl](#) () const
Gives the cfl 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.
- SCALAR [get_friccoef](#) () const
Gives the value of the friction coefficient.
- int [get_lim](#) () const
Gives the value corresponding to the limiter.
- int [get_topo](#) () const

- Gives the value corresponding to the topography.*
- int `get_hu ()` const
 - Gives the value corresponding to the initialization of h and u.*
 - int `get_rain ()` const
 - Gives the value corresponding to the initialization of the rain.*
 - int `get_inf ()` const
 - Gives the value corresponding to the initialization of the infiltration.*
 - int `get_Kc_init ()` const
 - Gives the value corresponding to the initialization of Kc.*
 - SCALAR `get_Kc_coef ()` const
 - Gives the value of the Kc coefficient.*
 - int `get_Ks_init ()` const
 - Gives the value corresponding to the initialization of Ks.*
 - SCALAR `get_Ks_coef ()` const
 - Gives the value of the Ks coefficient.*
 - int `get_dtheta_init ()` const
 - Gives the value corresponding to the initialization of dtheta.*
 - SCALAR `get_dtheta_coef ()` const
 - Gives the value of the dtheta coefficient.*
 - int `get_Psi_init ()` const
 - Gives the value corresponding to the initialization of Psi.*
 - SCALAR `get_Psi_coef ()` const
 - Gives the value of the Psi coefficient.*
 - int `get_zcrust_init ()` const
 - Gives the value corresponding to the initialization of zcrust.*
 - SCALAR `get_zcrust_coef ()` const
 - Gives the value of the zcrust coefficient.*
 - int `get_imax_init ()` const
 - Gives the value corresponding to the initialization of imax.*
 - SCALAR `get_imax_coef ()` const
 - Gives the value of the imax coefficient.*
 - SCALAR `get_amortENO ()` const
 - Gives the value of the amortENO parameter.*
 - SCALAR `get_modifENO ()` const
 - Gives the value of the modifENO parameter.*
- string `get_topographyNameFile ()` const
 - Gives the topography path + Input directory.*
- string `get_huNameFile (void)` const
 - Gives the h and u path for the initialization + Input directory.*
- string `get_rainNameFile (void)` const
 - Gives the rain path for the initialization + Input directory.*
- string `get_KcNameFile (void)` const
 - Gives the Kc path for the initialization + Input directory.*
- string `get_KsNameFile (void)` const
 - Gives the Ks path for the initialization + Input directory.*
- string `get_dthetaNameFile (void)` const
 - Gives the dtheta path for the initialization + Input directory.*

- string `get_PsiNameFile` (void) const
Gives the Psi path for the initialization + Input directory.
- string `get_zcrustNameFile` (void) const
Gives the zcrust path for the initialization + Input directory.
- string `get_imaxNameFile` (void) const
Gives the imax path for the initialization + Input directory.
- string `get_topographyNameFileS` () const
Gives the topography namefile (inside the Input directory)
- string `get_huNameFileS` (void) const
Gives the h and u namefile for the initialization (inside the Input directory)
- string `get_rainNameFileS` (void) const
Gives the rain namefile for the initialization (inside the Input directory)
- string `get_KcNameFileS` () const
Gives the Kc namefile (inside the Input directory)
- string `get_KsNameFileS` () const
Gives the Ks namefile (inside the Input directory)
- string `get_dthetaNameFileS` () const
Gives the dtheta namefile (inside the Input directory)
- string `get_PsiNameFileS` () const
Gives the Psi namefile (inside the Input directory)
- string `get_zcrustNameFileS` () const
Gives the zcrust namefile (inside the Input directory)
- string `get_imaxNameFileS` () const
Gives the imax namefile (inside the Input directory)
- string `get_outputDirectory` (void) const
Gives the output directory with the suffix.
- string `get_suffix` (void) const
Gives the suffix for the 'Outputs' directory.
- void `fill_array` (VECT &, const SCALAR) const
Fills the VECT array with a SCALAR.
- void `fill_array` (VECT &, string) const
Fills the VECT array with the values contained in a file.

Protected Attributes

- int `Nxcell`
- int `m`
- int `nbtimes`
- SCALAR `dx`
- SCALAR `dt`
- SCALAR `tx`
- SCALAR `L`
- SCALAR `T`
- SCALAR `cfl`
- int `Lbound`
- int `Rbound`
- SCALAR `L_imp_q`
- SCALAR `L_imp_h`

- SCALAR R_imp_q
- SCALAR R_imp_h
- int flux
- int order
- int rec
- int fric
- int lim
- int topo
- int hu_init
- int rain
- int inf
- int Kc_init
- int Ks_init
- int dtheta_init
- int Psi_init
- int zcrust_init
- int imax_init
- SCALAR amortENO
- SCALAR modifENO
- 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 hu_namefile
- string hu_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_o

5.51.1 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 74 of file parameters.hpp.

5.51.2 Constructor & Destructor Documentation

Parameters::Parameters ()

Constructor.

Definition at line 59 of file parameters.cpp.

Parameters::~~Parameters () [virtual]

Destructor.

Definition at line 1070 of file parameters.cpp.

5.51.3 Member Function Documentation

void Parameters::fill_array (VECT & *myarray*, const SCALAR *myvalue*) const

Fills the VECT 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 1644 of file parameters.cpp.

void Parameters::fill_array (VECT & *myarray*, string *namefile*) const

Fills the VECT 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 1656 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 1263 of file parameters.cpp.

SCALAR Parameters::get_cfl () const

Gives the cfl of the scheme.

Returns

The cfl of the scheme [Parameters::cfl](#).

Definition at line 1213 of file parameters.cpp.

SCALAR Parameters::get_dt () const

Gives the time step.

Returns

The time step [Parameters::dt](#).

Definition at line 1113 of file parameters.cpp.

SCALAR Parameters::get_dtheta_coef () const

Gives the value of the dtheta coefficient.

Returns

The value of dtheta [Parameters::dtheta_coef](#).

Definition at line 1473 of file parameters.cpp.

int Parameters::get_dtheta_init () const

Gives the value corresponding to the initialization of dtheta.

Returns

The value corresponding to the initialization of dtheta [Parameters::dtheta_init](#).

Definition at line 1463 of file parameters.cpp.

string Parameters::get_dthetaNameFile (void) const

Gives the dtheta path for the initialization + Input directory.

Returns

The dtheta path for the initialization + Input directory [Parameters::dtheta_namefile](#).

Definition at line 1483 of file parameters.cpp.

string Parameters::get_dthetaNameFileS () const

Gives the dtheta namefile (inside the Input directory)

Returns

The dtheta namefile for the initialization (inside the Input directory) [Parameters::dtheta_NF](#).

Definition at line 1493 of file parameters.cpp.

SCALAR Parameters::get_dx () const

Gives the space step.

Returns

The space step [Parameters::dx](#).

Definition at line 1093 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 1193 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 1233 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 1243 of file parameters.cpp.

int Parameters::get_hu () const

Gives the value corresponding to the initialization of h and u.

Returns

The value corresponding to the initialization of h and u [Parameters::hu_init](#).

Definition at line 1333 of file parameters.cpp.

string Parameters::get_huNameFile (void) const

Gives the h and u path for the initialization + Input directory.

Returns

The h and u path for the initialization + Input directory [Parameters::hu_namefile](#).

Definition at line 1313 of file parameters.cpp.

string Parameters::get_huNameFileS (void) const

Gives the h and u namefile for the initialization (inside the Input directory)

Returns

The h and u namefile for the initialization (inside the Input directory) [Parameters::hu_NF](#).

Definition at line 1323 of file parameters.cpp.

SCALAR Parameters::get_imax_coef () const

Gives the value of the imax coefficient.

Returns

The value of imax [Parameters::imax_coef](#).

Definition at line 1593 of file parameters.cpp.

int Parameters::get_imax_init () const

Gives the value corresponding to the initialization of imax.

Returns

The value corresponding to the initialization of imax [Parameters::imax_init](#).

Definition at line 1583 of file parameters.cpp.

string Parameters::get_imaxNameFile (void) const

Gives the imax path for the initialization + Input directory.

Returns

The imax path for the initialization + Input directory [Parameters::imax_namefile](#).

Definition at line 1603 of file parameters.cpp.

string Parameters::get_imaxNameFileS () const

Gives the imax namefile (inside the Input directory)

Returns

The imax namefile for the initialization (inside the Input directory) [Parameters::imax_NF](#).

Definition at line 1613 of file parameters.cpp.

int Parameters::get_inf () const

Gives the value corresponding to the initialization of the infiltration.

Returns

The value corresponding to the infiltration [Parameters::inf](#).

Definition at line 1373 of file parameters.cpp.

SCALAR Parameters::get_Kc_coef () const

Gives the value of the Kc coefficient.

Returns

The value of Kc [Parameters::Kc_coef](#).

Definition at line 1393 of file parameters.cpp.

int Parameters::get_Kc_init () const

Gives the value corresponding to the initialization of Kc.

Returns

The value corresponding to the initialization of Kc [Parameters::Kc_init](#).

Definition at line 1383 of file parameters.cpp.

string Parameters::get_KcNameFile (void) const

Gives the Kc path for the initialization + Input directory.

Returns

The Kc path for the initialization + Input directory [Parameters::Kc_namefile](#).

Definition at line 1403 of file parameters.cpp.

string Parameters::get_KcNameFileS () const

Gives the Kc namefile (inside the Input directory)

Returns

The Kc namefile for the initialization (inside the Input directory) [Parameters::Kc_NF](#).

Definition at line 1413 of file parameters.cpp.

SCALAR Parameters::get_Ks_coef () const

Gives the value of the Ks coefficient.

Returns

The value of Ks [Parameters::Ks_coef](#).

Definition at line 1433 of file parameters.cpp.

int Parameters::get_Ks_init () const

Gives the value corresponding to the initialization of Ks.

Returns

The value corresponding to the initialization of Ks [Parameters::Ks_init](#).

Definition at line 1423 of file parameters.cpp.

string Parameters::get_KsNameFile (void) const

Gives the Ks path for the initialization + Input directory.

Returns

The Ks path for the initialization + Input directory [Parameters::Ks_namefile](#).

Definition at line 1443 of file parameters.cpp.

string Parameters::get_KsNameFileS () const

Gives the Ks namefile (inside the Input directory)

Returns

The Ks namefile for the initialization (inside the Input directory) [Parameters::Ks_NF](#).

Definition at line 1453 of file parameters.cpp.

SCALAR Parameters::get_L_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::L_imp_h](#).

Definition at line 1153 of file parameters.cpp.

SCALAR Parameters::get_L_imp_q () const

Gives the value of the imposed discharge in left bc.

Returns

The value of the imposed discharge in the left boundary condition [Parameters::L_imp_q](#).

Definition at line 1143 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 1133 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 1253 of file parameters.cpp.

int Parameters::get_m () const

Gives the number of time steps to attain the final time.

Returns

The number of time steps to attain the final time [Parameters::m](#).

Definition at line 1083 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 1273 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 1103 of file parameters.cpp.

int Parameters::get_Nxcell () const

Gives the number of cells in space.

Returns

The number of cells in space [Parameters::Nxcell](#).

Definition at line 1073 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 1203 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 1623 of file parameters.cpp.

SCALAR Parameters::get_Psi_coef () const

Gives the value of the Psi coefficient.

Returns

The value of Psi [Parameters::Psi_coef](#).

Definition at line 1513 of file parameters.cpp.

int Parameters::get_Psi_init () const

Gives the value corresponding to the initialization of Psi.

Returns

The value corresponding to the initialization of Psi [Parameters::Psi_init](#).

Definition at line 1503 of file parameters.cpp.

string Parameters::get_PsiNameFile (void) const

Gives the Psi path for the initialization + Input directory.

Returns

The Psi path for the initialization + Input directory [Parameters::Psi_namefile](#).

Definition at line 1523 of file parameters.cpp.

string Parameters::get_PsiNameFileS () const

Gives the Psi namefile (inside the Input directory)

Returns

The Psi namefile for the initialization (inside the Input directory) [Parameters::Psi_NF](#).

Definition at line 1533 of file parameters.cpp.

SCALAR Parameters::get_R_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::R_imp_h](#).

Definition at line 1183 of file parameters.cpp.

SCALAR Parameters::get_R_imp_q () const

Gives the value of the imposed discharge in right bc.

Returns

The value of the imposed discharge in the right boundary condition [Parameters::R_imp_q](#).

Definition at line 1173 of file parameters.cpp.

int Parameters::get_rain () const

Gives the value corresponding to the initialization of the rain.

Returns

The value corresponding to the initialization of the rain [Parameters::rain](#).

Definition at line 1363 of file parameters.cpp.

string Parameters::get_rainNameFile (void) const

Gives the rain path for the initialization + Input directory.

Returns

The rain path for the initialization + Input directory [Parameters::rain_namefile](#).

Definition at line 1343 of file parameters.cpp.

string Parameters::get_rainNameFileS (void) const

Gives the rain namefile for the initialization (inside the Input directory)

Returns

The rain namefile for the initialization (inside the Input directory) [Parameters::rain_NF](#).

Definition at line 1353 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 1163 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 1223 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_o](#).

Definition at line 1633 of file parameters.cpp.

int Parameters::get_topo () const

Gives the value corresponding to the topography.

Returns

The value corresponding to the topography [Parameters::topo](#).

Definition at line 1303 of file parameters.cpp.

string Parameters::get_topographyNameFile (void) const

Gives the topography path + Input directory.

Returns

The topography path + Input directory [Parameters::topography_namefile](#).

Definition at line 1283 of file parameters.cpp.

string Parameters::get_topographyNameFileS (void) const

Gives the topography namefile (inside the Input directory)

Returns

The topography namefile (inside the Input directory) [Parameters::topo_NF](#).

Definition at line 1293 of file parameters.cpp.

SCALAR Parameters::get_tx () const

Gives the value of dt/dx.

Returns

The value of dt/dx [Parameters::tx](#).

Definition at line 1123 of file parameters.cpp.

SCALAR Parameters::get_zcrust_coef () const

Gives the value of the zcrust coefficient.

Returns

The value of zcrust [Parameters::zcrust_coef](#).

Definition at line 1553 of file parameters.cpp.

int Parameters::get_zcrust_init () const

Gives the value corresponding to the initialization of zcrust.

Returns

The value corresponding to the initialization of zcrust [Parameters::zcrust_init](#).

Definition at line 1543 of file parameters.cpp.

string Parameters::get_zcrustNameFile (void) const

Gives the zcrust path for the initialization + Input directory.

Returns

The zcrust path for the initialization + Input directory [Parameters::zcrust_namefile](#).

Definition at line 1563 of file parameters.cpp.

string Parameters::get_zcrustNameFileS () const

Gives the zcrust namefile (inside the Input directory)

Returns

The zcrust namefile for the initialization (inside the Input directory) [Parameters::zcrust_NF](#).

Definition at line 1573 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_1D 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 parameters 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 63 of file parameters.cpp.

5.51.4 Member Data Documentation**SCALAR Parameters::amortENO [protected]**

Parameter for eno.

Definition at line 326 of file parameters.hpp.

SCALAR Parameters::cfl [protected]

CFL value.

Definition at line 282 of file parameters.hpp.

SCALAR Parameters::dt [protected]

Time step.

Definition at line 274 of file parameters.hpp.

SCALAR Parameters::dtheta_coef [protected]

Value of dtheta.

Definition at line 336 of file parameters.hpp.

int Parameters::dtheta_init [protected]

Type of initialization of dtheta.

Definition at line 318 of file parameters.hpp.

string Parameters::dtheta_namefile [protected]

Name of the file for dtheta: Inputs/file.

Definition at line 364 of file parameters.hpp.

string Parameters::dtheta_NF [protected]

Name of the file for dtheta without 'Inputs'.

Definition at line 366 of file parameters.hpp.

SCALAR Parameters::dx [protected]

Space step.

Definition at line 272 of file parameters.hpp.

int Parameters::flux [protected]

Numerical flux.

Definition at line 296 of file parameters.hpp.

int Parameters::fric [protected]

Friction.

Definition at line 302 of file parameters.hpp.

SCALAR Parameters::friccoef [protected]

Friction coefficient.

Definition at line 330 of file parameters.hpp.

int Parameters::hu_init [protected]

Type of initial conditions for h and u.

Definition at line 308 of file parameters.hpp.

string Parameters::hu_namefile [protected]

Name of the file for the initialization of h and u: Inputs/file.
Definition at line 348 of file parameters.hpp.

string Parameters::hu_NF [protected]

Name of the file for the initialization of h and u without 'Inputs'.
Definition at line 350 of file parameters.hpp.

SCALAR Parameters::imax_coef [protected]

Value of imax.
Definition at line 342 of file parameters.hpp.

int Parameters::imax_init [protected]

Type of initialization of imax.
Definition at line 324 of file parameters.hpp.

string Parameters::imax_namefile [protected]

Name of the file for imax: Inputs/file.
Definition at line 376 of file parameters.hpp.

string Parameters::imax_NF [protected]

Name of the file for imax without 'Inputs'.
Definition at line 378 of file parameters.hpp.

int Parameters::inf [protected]

Type of infiltration.
Definition at line 312 of file parameters.hpp.

SCALAR Parameters::Kc_coef [protected]

Value of Kc.
Definition at line 332 of file parameters.hpp.

int Parameters::Kc_init [protected]

Type of initialization of Kc.
Definition at line 314 of file parameters.hpp.

string Parameters::Kc_namefile [protected]

Name of the file for Kc: Inputs/file.
Definition at line 356 of file parameters.hpp.

string Parameters::Kc_NF [protected]

Name of the file for Kc without 'Inputs'.
Definition at line 358 of file parameters.hpp.

SCALAR Parameters::Ks_coef [protected]

Value of Ks.

Definition at line 334 of file parameters.hpp.

int Parameters::Ks_init [protected]

Type of initialization of Ks.

Definition at line 316 of file parameters.hpp.

string Parameters::Ks_namefile [protected]

Name of the file for Ks: Inputs/file.

Definition at line 360 of file parameters.hpp.

string Parameters::Ks_NF [protected]

Name of the file for Ks without 'Inputs'.

Definition at line 362 of file parameters.hpp.

SCALAR Parameters::L [protected]

Length of the domain.

Definition at line 278 of file parameters.hpp.

SCALAR Parameters::L_imp_h [protected]

Imposed water height on the left boundary.

Definition at line 290 of file parameters.hpp.

SCALAR Parameters::L_imp_q [protected]

Imposed discharge on the left boundary.

Definition at line 288 of file parameters.hpp.

int Parameters::Lbound [protected]

Left boundary condition.

Definition at line 284 of file parameters.hpp.

int Parameters::lim [protected]

Slope limiter.

Definition at line 304 of file parameters.hpp.

int Parameters::m [protected]

Number of time steps.

Definition at line 268 of file parameters.hpp.

SCALAR Parameters::modifENO [protected]

Parameter for eno_modif.

Definition at line 328 of file parameters.hpp.

int Parameters::nbtimes [protected]

Number of times saved.

Definition at line 270 of file parameters.hpp.

int Parameters::Nxcell [protected]

Number of cells in space.

Definition at line 266 of file parameters.hpp.

int Parameters::order [protected]

Order of the numerical scheme.

Definition at line 298 of file parameters.hpp.

string Parameters::output_directory [protected]

Name of the output directory Outputs+suffix.

Definition at line 380 of file parameters.hpp.

SCALAR Parameters::Psi_coef [protected]

Value of Psi.

Definition at line 338 of file parameters.hpp.

int Parameters::Psi_init [protected]

Type of initialization of Psi.

Definition at line 320 of file parameters.hpp.

string Parameters::Psi_namefile [protected]

Name of the file for Psi: Inputs/file.

Definition at line 368 of file parameters.hpp.

string Parameters::Psi_NF [protected]

Name of the file for Psi without 'Inputs'.

Definition at line 370 of file parameters.hpp.

SCALAR Parameters::R_imp_h [protected]

Imposed water height on the right boundary.

Definition at line 294 of file parameters.hpp.

SCALAR Parameters::R_imp_q [protected]

Imposed discharge on the right boundary.

Definition at line 292 of file parameters.hpp.

int Parameters::rain [protected]

Type of rain.

Definition at line 310 of file parameters.hpp.

string Parameters::rain_namefile [protected]

Name of the file for the rain: Inputs/file.

Definition at line 352 of file parameters.hpp.

string Parameters::rain_NF [protected]

Name of the file for the rain without 'Inputs'.

Definition at line 354 of file parameters.hpp.

int Parameters::Rbound [protected]

Right boundary condition.

Definition at line 286 of file parameters.hpp.

int Parameters::rec [protected]

Reconstruction.

Definition at line 300 of file parameters.hpp.

string Parameters::suffix_o [protected]

Suffix for the output directory.

Definition at line 382 of file parameters.hpp.

SCALAR Parameters::T [protected]

Final time.

Definition at line 280 of file parameters.hpp.

int Parameters::topo [protected]

Type of topography.

Definition at line 306 of file parameters.hpp.

string Parameters::topo_NF [protected]

Name of the file for the topography without 'Inputs'.

Definition at line 346 of file parameters.hpp.

string Parameters::topography_namefile [protected]

Name of the file for the topography: Inputs/file.

Definition at line 344 of file parameters.hpp.

SCALAR Parameters::tx [protected]

Ratio dt/dx.

Definition at line 276 of file parameters.hpp.

SCALAR Parameters::zcrust_coef [protected]

Value of zcrust.

Definition at line 340 of file parameters.hpp.

int Parameters::zcrust_init [protected]

Type of initialization of zcrust.

Definition at line 322 of file parameters.hpp.

string Parameters::zcrust_namefile [protected]

Name of the file for zcrust: Inputs/file.

Definition at line 372 of file parameters.hpp.

string Parameters::zcrust_NF [protected]

Name of the file for zcrust without 'Inputs'.

Definition at line 374 of file parameters.hpp.

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

- Headers/libparameters/[parameters.hpp](#)
- Sources/libparameters/[parameters.cpp](#)

5.52 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.

5.52.1 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 79 of file parser.hpp.

5.52.2 Constructor & Destructor Documentation

Parser::Parser (const char * *FILENAME*)

Constructor.

Constructor: reads the input parameter and copy the data in a tabular.

Parameters

<i>in</i>	<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.

5.52.3 Member Function Documentation**string Parser::GetValue (const char * TAG)**

Returns the value of the variable.

Return the value corresponding to the tag.

Parameters

in	TAG	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:

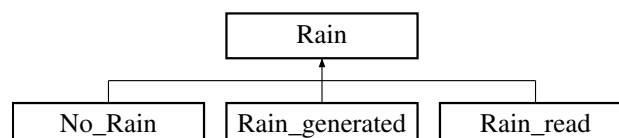
- Headers/libparser/[parser.hpp](#)
- Sources/libparser/[parser.cpp](#)

5.53 Rain Class Reference

Initialization of the rain.

```
#include <rain.hpp>
```

Inheritance diagram for Rain:



Public Member Functions

- [Rain \(Parameters &\)](#)
Constructor.
- virtual void [rain_func \(SCALAR, VECT &\)=0](#)
Function to be specified in each case.
- virtual [~Rain \(\)](#)
Destructor.

Protected Attributes

- const int [NXCELL](#)
- const [SCALAR DX](#)

5.53.1 Detailed Description

Initialization of the rain.

Class that contains all the common declarations for the initialization of the rain.

Definition at line 69 of file rain.hpp.

5.53.2 Constructor & Destructor Documentation

Rain::Rain (Parameters & *par*)

Constructor.

Defines the number of cells and the space step.

Parameters

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

Definition at line 57 of file rain.cpp.

Rain::~~Rain () [virtual]

Destructor.

Definition at line 68 of file rain.cpp.

5.53.3 Member Function Documentation

virtual void Rain::rain_func (SCALAR , VECT &) [pure virtual]

Function to be specified in each case.

Implemented in [Rain_read](#), [Rain_generated](#), and [No_Rain](#).

5.53.4 Member Data Documentation

const SCALAR Rain::DX [protected]

Space step (unused).

Definition at line 86 of file rain.hpp.

const int Rain::NXCELL [protected]

Number of cells in space.

Definition at line 84 of file rain.hpp.

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

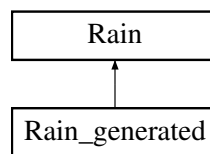
- Headers/librain_infiltration/rain.hpp
- Sources/librain_infiltration/rain.cpp

5.54 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 \(SCALAR, VECT &\)](#)
Performs the constant initialization.
- virtual [~Rain_generated \(\)](#)
Destructor.

Additional Inherited Members**5.54.1 Detailed Description**

Constant rain configuration.

Class that initializes a constant rain, with value 0.00001 m/s = 36 mm/h.

Definition at line 71 of file rain_generated.hpp.

5.54.2 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.

5.54.3 Member Function Documentation

void Rain_generated::rain_func (SCALAR *time*, VECT & *Tab_rain*) [virtual]

Performs the constant initialization.

Initializes the rain to 0.00001 m/s = 36 mm/h.

Parameters

in	<i>time</i>	the current time (unused)
in, out	<i>Tab_rain</i>	rain intensity at the current time on each cell.

Implements [Rain](#).

Definition at line 68 of file rain_generated.cpp.

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

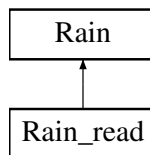
- [Headers/librain_infiltration/rain_generated.hpp](#)
- [Sources/librain_infiltration/rain_generated.cpp](#)

5.55 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, VECT &)
Performs the initialization.
- virtual [~Rain_read](#) ()
Destructor.

Additional Inherited Members

5.55.1 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.

5.55.2 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 153 of file rain_read.cpp.

5.55.3 Member Function Documentation**void Rain_read::rain_func (SCALAR *time*, VECT & *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 133 of file rain_read.cpp.

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

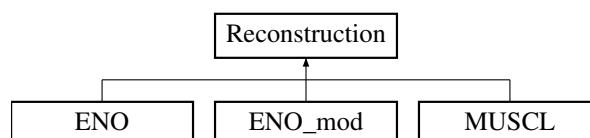
- Headers/librain_infiltration/[rain_read.hpp](#)
- Sources/librain_infiltration/[rain_read.cpp](#)

5.56 Reconstruction Class Reference

Reconstruction of the variables

```
#include <reconstruction.hpp>
```

Inheritance diagram for Reconstruction:



Public Member Functions

- [Reconstruction](#) ([Parameters](#) &, [VECT](#))
Constructor.
- virtual void [calc](#) ([VECT](#), [VECT](#), [VECT](#), [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &, [VECT](#) &)=0
Function to be specified in each reconstruction.
- virtual [~Reconstruction](#) ()
Destructor.

Protected Attributes

- const int [NXCELL](#)
- [VECT](#) [zr](#)
- [VECT](#) [zl](#)
- [VECT](#) [delta0_z](#)
- [Choice_limiter](#) * [limiter](#)

5.56.1 Detailed Description

Reconstruction of the variables

Class that contains all the common declarations for the second order reconstruction in space.

Definition at line 73 of file reconstruction.hpp.

5.56.2 Constructor & Destructor Documentation

Reconstruction::Reconstruction ([Parameters](#) & *par*, [VECT](#) *z*)

Constructor.

Defines the number of cells, the slope limiter, and initializes [Reconstruction::zl](#), [Reconstruction::zr](#), [Reconstruction::delta0_z](#).

Parameters

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

Warning

Problem: allocation of *** failed.

Note

If a vector cannot be allocated, the code will exit with failure termination code.

Definition at line 58 of file reconstruction.cpp.

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

Destructor.

Definition at line 97 of file reconstruction.cpp.

5.56.3 Member Function Documentation

virtual void [Reconstruction::calc](#) ([VECT](#) , [VECT](#) , [VECT](#) , [VECT](#) & , [VECT](#) & , [VECT](#) & , [VECT](#) & , [VECT](#) & , [VECT](#) &) [[pure virtual](#)]

Function to be specified in each reconstruction.

Implemented in [ENO](#), [MUSCL](#), and [ENO_mod](#).

5.56.4 Member Data Documentation

VECT Reconstruction::delta0_z [protected]

Difference between the values of the topography on two adjacent cells (on the right)

Definition at line 95 of file reconstruction.hpp.

Choice_limiter* Reconstruction::limiter [protected]

Slope limiter

Definition at line 97 of file reconstruction.hpp.

const int Reconstruction::NXCELL [protected]

Number of cells in space

Definition at line 89 of file reconstruction.hpp.

VECT Reconstruction::zl [protected]

Reconstructed topography on the left boundary

Definition at line 93 of file reconstruction.hpp.

VECT Reconstruction::zr [protected]

Reconstructed topography on the right boundary

Definition at line 91 of file reconstruction.hpp.

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

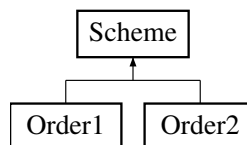
- Headers/libreconstructions/[reconstruction.hpp](#)
- Sources/libreconstructions/[reconstruction.cpp](#)

5.57 Scheme Class Reference

Numerical scheme.

```
#include <scheme.hpp>
```

Inheritance diagram for Scheme:



Public Member Functions

- [Scheme](#) ([Parameters](#) &)
Constructor.
- virtual void [calc](#) ()=0
Function to be specified in each numerical scheme.
- void [allocation](#) ()
- void [deallocation](#) ()
- void [maincalc](#) ([VECT](#), [VECT](#), [SCALAR](#), [VECT](#) &, [VECT](#) &, [VECT](#) &, [SCALAR](#) &, [SCALAR](#) &, [VECT](#) &)
Main calculation of the scheme.
- [SCALAR](#) [froude_number](#) ([VECT](#), [VECT](#))

Computation of the mean value of the Froude number.

- void `boundary` (`VECT &`, `VECT &`, const `SCALAR`, const `SCALAR`, const `SCALAR`, const `SCALAR`, `SCALAR`, const int)

Calls the boundary conditions and affects the boundary values.

- virtual `~Scheme` ()

Destructor.

Protected Attributes

- const int `NXCELL`
- const int `M`
- const `SCALAR DX`
- const `SCALAR DT`
- const `SCALAR TX`
- const int `NBTIMES`
- const `SCALAR CFL`
- const `SCALAR L_IMP_Q`
- const `SCALAR L_IMP_H`
- const `SCALAR R_IMP_Q`
- const `SCALAR R_IMP_H`
- const int `ORDER`
- int `nbt`
- `VECT h`
- `VECT u`
- `VECT q`
- `VECT z`
- `VECT rain`
- `VECT Vin_tot`
- `VECT delta_z`
- `VECT dzi`
- `VECT f1`
- `VECT f2`
- `VECT hs`
- `VECT us`
- `VECT qs`
- `VECT hl`
- `VECT hr`
- `VECT ul`
- `VECT ur`
- `VECT hleft`
- `VECT hright`
- `SCALAR flux_l`
- `SCALAR flux_r`
- `SCALAR cum_flux_l`
- `SCALAR cum_flux_r`
- `SCALAR Vol_rain_tot`
- `SCALAR Total_volume_outflow`
- `SCALAR height_of_tot`
- `SCALAR height_inf_tot`

- SCALAR Vol_inf_tot
- SCALAR Vol_of_tot
- SCALAR Fr
- int time_initial
- time_t start
- time_t end
- SCALAR elapsed_time
- clock_t cpu_time
- Choice_condition * Lbound
- Choice_condition * Rbound
- Choice_output * out

5.57.1 Detailed Description

Numerical scheme.

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

Definition at line 105 of file scheme.hpp.

5.57.2 Constructor & Destructor Documentation

Scheme::Scheme (Parameters & *par*)

Constructor.

Initializations and allocations.

Parameters

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

Definition at line 58 of file scheme.cpp.

Scheme::~Scheme () [virtual]

Destructor.

Definition at line 249 of file scheme.cpp.

5.57.3 Member Function Documentation

void Scheme::allocation ()

Allocation of spatialized variables

Warning

Problem: allocation of *** failed.

Note

If a vector cannot be allocated, the code will exit with failure termination code.

Definition at line 257 of file scheme.cpp.

void Scheme::boundary (VECT & *h_tmp*, VECT & *u_tmp*, const SCALAR *L_IMP_Q*, const SCALAR *L_IMP_H*, const SCALAR *R_IMP_Q*, const SCALAR *R_IMP_H*, SCALAR *time_tmp*, const int *NODEX*)

Calls the boundary conditions and affects the boundary values.

Parameters

in, out	<i>h_tmp</i>	water height.
in, out	<i>u_tmp</i>	velocity.
in	<i>L_IMP_Q</i>	left imposed discharge.
in	<i>L_IMP_H</i>	left imposed water height.
in	<i>R_IMP_Q</i>	right imposed discharge.
in	<i>R_IMP_H</i>	right imposed water height.
in	<i>time_tmp</i>	current time.
in	<i>NODEX</i>	number of space cells.

Definition at line 194 of file scheme.cpp.

virtual void Scheme::calc () [pure virtual]

Function to be specified in each numerical scheme.

Implemented in [Order1](#), and [Order2](#).

void Scheme::deallocation ()

Deallocation of variables

Definition at line 381 of file scheme.cpp.

SCALAR Scheme::froude_number (VECT *h*, VECT *u*)

Computation of the mean value of the Froude number.

Mean value in space of the Froude number at the final time.

Parameters

in	<i>h</i>	water height.
in	<i>u</i>	velocity.

Returns

The mean Froude number $\frac{u}{\sqrt{gh}}$.

Definition at line 218 of file scheme.cpp.

void Scheme::maincalc (VECT *he*, VECT *ve*, SCALAR *time*, VECT & *hes*, VECT & *ves*, VECT & *qes*, SCALAR & *flux_l*, SCALAR & *flux_r*, VECT & *Vin*)

Main calculation of the scheme.

This calculation is called once at the order 1, and twice at the second order.

Parameters

in	<i>he</i>	water height.
in	<i>ve</i>	velocity.
in	<i>time</i>	time to be attained at the end of the loop.
out	<i>hes</i>	water height after one step of the scheme.
out	<i>ves</i>	velocity after one step of the scheme.
out	<i>qes</i>	discharge after one step of the scheme.

out	<i>flux_l</i>	Flux on the left.
out	<i>flux_r</i>	Flux on the right.
in, out	<i>Vin</i>	infiltrated volume.

Warning

the CFL condition is not satisfied: CFL > ***

Todo Improve the treatment of numerical errors.

Definition at line 115 of file scheme.cpp.

5.57.4 Member Data Documentation**const SCALAR Scheme::CFL [protected]**

CFL value.

Definition at line 148 of file scheme.hpp.

clock_t Scheme::cpu_time [protected]

CPU time.

Definition at line 230 of file scheme.hpp.

SCALAR Scheme::cum_flux_l [protected]

Cumulative flux on the left (m^2).

Definition at line 204 of file scheme.hpp.

SCALAR Scheme::cum_flux_r [protected]

Cumulative flux on the right (m^2).

Definition at line 206 of file scheme.hpp.

VECT Scheme::delta_z [protected]

Variations of the topography.

Definition at line 174 of file scheme.hpp.

const SCALAR Scheme::DT [protected]

Time step.

Definition at line 142 of file scheme.hpp.

const SCALAR Scheme::DX [protected]

Space step.

Definition at line 140 of file scheme.hpp.

VECT Scheme::dzi [protected]

Difference between the reconstructed topographies on the left and on the right boundary of a cell.

Definition at line 176 of file scheme.hpp.

SCALAR Scheme::elapsed_time [protected]

Duration of the computation.

Definition at line 228 of file scheme.hpp.

time_t Scheme::end [protected]

End of timer.

Definition at line 226 of file scheme.hpp.

VECT Scheme::f1 [protected]

First component of the numerical flux.

Definition at line 178 of file scheme.hpp.

VECT Scheme::f2 [protected]

Second component of the numerical flux.

Definition at line 180 of file scheme.hpp.

SCALAR Scheme::flux_l [protected]

Flux on the left (m^2/s).

Definition at line 200 of file scheme.hpp.

SCALAR Scheme::flux_r [protected]

Flux on the right (m^2/s).

Definition at line 202 of file scheme.hpp.

SCALAR Scheme::Fr [protected]

Mean Froude number.

Definition at line 220 of file scheme.hpp.

VECT Scheme::h [protected]

Water height.

Definition at line 162 of file scheme.hpp.

SCALAR Scheme::height_inf_tot [protected]

Cumulative height of infiltrated volume on the whole domain

Definition at line 214 of file scheme.hpp.

SCALAR Scheme::height_of_tot [protected]

Cumulative water height on the whole domain

Definition at line 212 of file scheme.hpp.

VECT Scheme::hl [protected]

Water height on the cell located at the left of the boundary.

Definition at line 188 of file scheme.hpp.

VECT Scheme::hleft [protected]

Hydrostatic reconstruction on the left.

Definition at line 196 of file scheme.hpp.

VECT Scheme::hr [protected]

Water height on the cell located at the right of the boundary.

Definition at line 190 of file scheme.hpp.

VECT Scheme::hright [protected]

Hydrostatic reconstruction on the right.

Definition at line 198 of file scheme.hpp.

VECT Scheme::hs [protected]

Water height after one step of the scheme.

Definition at line 182 of file scheme.hpp.

const SCALAR Scheme::L_IMP_H [protected]

Imposed water height on the left boundary.

Definition at line 152 of file scheme.hpp.

const SCALAR Scheme::L_IMP_Q [protected]

Imposed discharge on the left boundary.

Definition at line 150 of file scheme.hpp.

Choice_condition* Scheme::Lbound [protected]

The choice of the left boundary condition.

Definition at line 232 of file scheme.hpp.

const int Scheme::M [protected]

Number of time steps.

Definition at line 138 of file scheme.hpp.

int Scheme::nbt [protected]

Number of time steps between two savings.

Definition at line 160 of file scheme.hpp.

const int Scheme::NBTIMES [protected]

Number of times saved.

Definition at line 146 of file scheme.hpp.

const int Scheme::NXCELL [protected]

Number of cells in space.

Definition at line 136 of file scheme.hpp.

const int Scheme::ORDER [protected]

Order of the scheme.

Definition at line 158 of file scheme.hpp.

Choice_output* Scheme::out [protected]

The choice of output.

Definition at line 236 of file scheme.hpp.

VECT Scheme::q [protected]

Discharge.

Definition at line 166 of file scheme.hpp.

VECT Scheme::qs [protected]

Discharge after one step of the scheme.

Definition at line 186 of file scheme.hpp.

const SCALAR Scheme::R_IMP_H [protected]

Imposed water height on the right boundary.

Definition at line 156 of file scheme.hpp.

const SCALAR Scheme::R_IMP_Q [protected]

Imposed discharge on the right boundary.

Definition at line 154 of file scheme.hpp.

VECT Scheme::rain [protected]

[Rain](#).

Definition at line 170 of file scheme.hpp.

Choice_condition* Scheme::Rbound [protected]

The choice of the right boundary condition.

Definition at line 234 of file scheme.hpp.

time_t Scheme::start [protected]

Beginning of timer.

Definition at line 224 of file scheme.hpp.

int Scheme::time_initial [protected]

Initial time: the initial time is $\text{time_initial} * dt$.

Definition at line 222 of file scheme.hpp.

SCALAR Scheme::Total_volume_outflow [protected]

Cumulative outflow volume at the boundary

Definition at line 210 of file scheme.hpp.

const SCALAR Scheme::TX [protected]

Ratio dt/dx.

Definition at line 144 of file scheme.hpp.

VECT Scheme::u [protected]

Velocity.

Definition at line 164 of file scheme.hpp.

VECT Scheme::ul [protected]

Velocity on the cell located at the left of the boundary.

Definition at line 192 of file scheme.hpp.

VECT Scheme::ur [protected]

Velocity on the cell located at the right of the boundary.

Definition at line 194 of file scheme.hpp.

VECT Scheme::us [protected]

Velocity after one step of the scheme.

Definition at line 184 of file scheme.hpp.

VECT Scheme::Vin_tot [protected]

Volume of infiltrated water at each point during one time step.

Definition at line 172 of file scheme.hpp.

SCALAR Scheme::Vol_inf_tot [protected]

Cumulative Volume of water infiltrated.

Definition at line 216 of file scheme.hpp.

SCALAR Scheme::Vol_of_tot [protected]

Cumulative stream volume.

Definition at line 218 of file scheme.hpp.

SCALAR Scheme::Vol_rain_tot [protected]

Cumulative Volume of rain on the whole domain.

Definition at line 208 of file scheme.hpp.

VECT Scheme::z [protected]

Topography.

Definition at line 168 of file scheme.hpp.

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

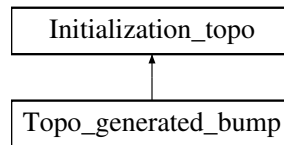
- Headers/lib schemes/[scheme.hpp](#)
- Sources/lib schemes/[scheme.cpp](#)

5.58 Topo_generated_bump Class Reference

Bump configuration.

```
#include <topo_generated_bump.hpp>
```

Inheritance diagram for Topo_generated_bump:



Public Member Functions

- [Topo_generated_bump](#) ([Parameters](#) &)
Constructor.
- void [initialization](#) ([VECT](#) &)
Performs the initialization.
- virtual [~Topo_generated_bump](#) ()
Destructor.

Additional Inherited Members

5.58.1 Detailed Description

Bump configuration.

Class that initializes a bump topography.

Definition at line 69 of file topo_generated_bump.hpp.

5.58.2 Constructor & Destructor Documentation

Topo_generated_bump::Topo_generated_bump (Parameters & par)

Constructor.

Parameters

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

Definition at line 58 of file topo_generated_bump.cpp.

Topo_generated_bump::~~Topo_generated_bump () [virtual]

Destructor.

Definition at line 78 of file topo_generated_bump.cpp.

5.58.3 Member Function Documentation

void Topo_generated_bump::initialization (VECT & z) [virtual]

Performs the initialization.

Initializes the topography to $0.2 - 0.005(x - 10)^2$ if $x \in [8, 12]$, and 0 outside the bump, see [Goutal and Maurel \[1997\]](#).

Parameters

<code>in, out</code>	<code>z</code>	topography.
----------------------	----------------	-------------

Implements [Initialization_topo](#).

Definition at line 65 of file `topo_generated_bump.cpp`.

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

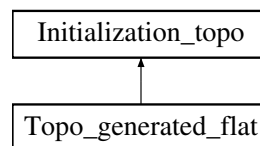
- [Headers/libinitializations/topo_generated_bump.hpp](#)
- [Sources/libinitializations/topo_generated_bump.cpp](#)

5.59 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](#) ([VECT](#) &)
Performs the initialization.
- virtual [~Topo_generated_flat](#) ()
Destructor.

Additional Inherited Members**5.59.1 Detailed Description**

Flat configuration.

Class that initializes a flat topography, with value 0.

Definition at line 70 of file `topo_generated_flat.hpp`.

5.59.2 Constructor & Destructor Documentation

Topo_generated_flat::Topo_generated_flat ([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 `topo_generated_flat.cpp`.

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

Destructor.

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

5.59.3 Member Function Documentation

void Topo_generated_flat::initialization (VECT & z) [virtual]

Performs the initialization.

Initializes the topography to 0.

Parameters

in, out	z	topography.
---------	---	-------------

Implements [Initialization_topo](#).

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

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

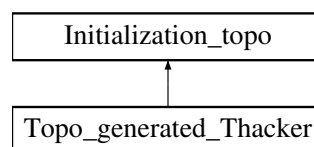
- [Headers/libinitializations/topo_generated_flat.hpp](#)
- [Sources/libinitializations/topo_generated_flat.cpp](#)

5.60 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](#) ([VECT](#) &)
Performs the initialization.
- virtual [~Topo_generated_Thacker](#) ()
Destructor.

Additional Inherited Members

5.60.1 Detailed Description

Thacker configuration.

Class that initializes a parabolic topography for Thacker's benchmark.

Definition at line 70 of file `topo_generated_thacker.hpp`.

5.60.2 Constructor & Destructor Documentation

Topo_generated_Thacker::Topo_generated_Thacker (Parameters & par)

Constructor.

Defines the parameter `h0` and `a` of the parabola.

Parameters

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

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

Topo_generated_Thacker::~~Topo_generated_Thacker () [virtual]

Destructor.

Definition at line 87 of file `topo_generated_thacker.cpp`.

5.60.3 Member Function Documentation**void Topo_generated_Thacker::initialization (VECT & z) [virtual]**

Performs the initialization.

Initializes the topography to $h_0 \left(\frac{(x-L/2)^2}{a^2} - 1 \right)$, see [Thacker \[1981\]](#).

Parameters

<code>in, out</code>	<code>z</code>	topography.
----------------------	----------------	-------------

Implements [Initialization_topo](#).

Definition at line 70 of file `topo_generated_thacker.cpp`.

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

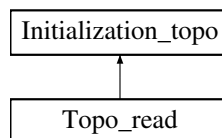
- [Headers/libinitializations/topo_generated_thacker.hpp](#)
- [Sources/libinitializations/topo_generated_thacker.cpp](#)

5.61 Topo_read Class Reference

File configuration.

```
#include <topo_read.hpp>
```

Inheritance diagram for `Topo_read`:

**Public Member Functions**

- [Topo_read \(Parameters &\)](#)
Constructor.
- void [initialization \(VECT &\)](#)
Performs the initialization.
- virtual [~Topo_read \(\)](#)
Destructor.

Additional Inherited Members**5.61.1 Detailed Description**

File configuration.

Class that initializes the topography to the values read in a file.

Definition at line 70 of file `topo_read.hpp`.

5.61.2 Constructor & Destructor Documentation

Topo_read::Topo_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 59 of file topo_read.cpp.

Topo_read::~~Topo_read () [virtual]

Destructor.

Definition at line 168 of file topo_read.cpp.

5.61.3 Member Function Documentation

void Topo_read::initialization (VECT & *z*) [virtual]

Performs the initialization.

Initializes the topography to the values read in the corresponding file.

Parameters

<i>in, out</i>	<i>z</i>	topography.
----------------	----------	-------------

Warning

(topography_namefile): ERROR: cannot open the topography file.

(topography_namefile): ERROR: the number of data in this file is too big

(topography_namefile): ERROR: line ***.

(topography_namefile): WARNING: line ***.

(topography_namefile): ERROR: the number of data in this file is too small

(topography_namefile): ERROR: the value for the point x *** 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 70 of file topo_read.cpp.

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

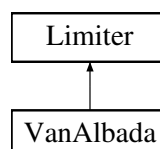
- [Headers/libinitializations/topo_read.hpp](#)
- [Sources/libinitializations/topo_read.cpp](#)

5.62 VanAlbada Class Reference

Van Albada slope limiter.

```
#include <vanalbada.hpp>
```

Inheritance diagram for VanAlbada:



Public Member Functions

- [VanAlbada](#) ()
Constructor.
- void [calc](#) (SCALAR, SCALAR)
Calculates the value of the slope limiter.
- virtual [~VanAlbada](#) ()
Destructor.

Additional Inherited Members

5.62.1 Detailed Description

Van Albada slope limiter.

Class that calculates Van Albada slope limiter.

Definition at line 69 of file vanalbada.hpp.

5.62.2 Constructor & Destructor Documentation

VanAlbada::VanAlbada ()

Constructor.

Definition at line 58 of file vanalbada.cpp.

VanAlbada::~VanAlbada () [virtual]

Destructor.

Definition at line 84 of file vanalbada.cpp.

5.62.3 Member Function Documentation

void VanAlbada::calc (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](#) recontructed value.

Implements [Limiter](#).

Definition at line 61 of file vanalbada.cpp.

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

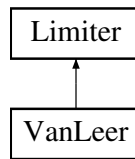
- Headers/liblimitations/[vanalbada.hpp](#)
- Sources/liblimitations/[vanalbada.cpp](#)

5.63 VanLeer Class Reference

Van Leer slope limiter.

```
#include <vanleer.hpp>
```

Inheritance diagram for VanLeer:



Public Member Functions

- [VanLeer](#) ()
Constructor.
- void [calc](#) (SCALAR, SCALAR)
Calculates the value of the slope limiter.
- virtual [~VanLeer](#) ()
Destructor.

Additional Inherited Members

5.63.1 Detailed Description

Van Leer slope limiter.

Class that calculates Van Leer slope limiter.

Definition at line 69 of file vanleer.hpp.

5.63.2 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.

5.63.3 Member Function Documentation

void VanLeer::calc (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

<code>in</code>	<code>a</code>	slope on the left of the cell.
<code>in</code>	<code>b</code>	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](#)

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

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2015)

Version

1.02.01

Date

2015-03-12

Boundary condition: imposed discharge (and water height if necessary).

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (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 (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (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 (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Boundary condition: Neumann condition (the normal derivative is null).

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (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 (2010)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Boundary condition: periodic condition.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (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 (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (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 (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the boundary conditions.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.7 Headers/libboundaryconditions/choice_condition.hpp File Reference

Choice of boundary condition.

```
#include "boundary_condition.hpp"  
#include "bc_neumann.hpp"  
#include "bc_imp_height.hpp"  
#include "bc_imp_discharge.hpp"  
#include "bc_wall.hpp"  
#include "bc_periodic.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 (2008)
 Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.7.2 Macro Definition Documentation

#define CHOICE_CONDITION_HPP

Definition at line 82 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"
#include "f_kinetic.hpp"
#include "f_vfroe.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 (2010)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.8.2 Macro Definition Documentation

#define CHOICE_FLUX_HPP

Definition at line 83 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 (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: Harten, Lax, van Leer formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (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 (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: Harten, Lax, van Leer reduced formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.11 Headers/libflux/f_kinetic.hpp File Reference

Kinetic flux.

```
#include "flux.hpp"
```

Classes

- class [F_Kinetic](#)
Kinetic flux.

6.11.1 Detailed Description

Kinetic flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: kinetic formulation.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.12 Headers/libflux/f_rusanov.hpp File Reference

Rusanov flux.

```
#include "flux.hpp"
```

Classes

- class [F_Rusanov](#)
Rusanov flux.

6.12.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: Rusanov formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.13 Headers/libflux/f_vfroe.hpp File Reference

VFRoe flux.

```
#include "flux.hpp"
```

Classes

- class [F_VFRoe](#)

VFRoe flux.

6.13.1 Detailed Description

VFRoe flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: VFRoe-ncv formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.14 Headers/libflux/flux.hpp File Reference

Numerical flux.

```
#include "parameters.hpp"
```

Classes

- class [Flux](#)
Numerical flux.

6.14.1 Detailed Description

Numerical flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the numerical fluxes.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.15 Headers/libfrictions/choice_friction.hpp File Reference

Choice of friction law.

```
#include "friction.hpp"  
#include "no_friction.hpp"  
#include "fr_manning.hpp"  
#include "fr_darcy_weisbach.hpp"  
#include "fr_laminar.hpp"
```

Classes

- class [Choice_friction](#)
Choice of friction law.

Macros

- #define [CHOICE_FRICTION_HPP](#)

6.15.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.15.2 Macro Definition Documentation

#define CHOICE_FRICTION_HPP

Definition at line 78 of file choice_friction.hpp.

6.16 Headers/libfrictions/fr_darcy_weisbach.hpp File Reference

Darcy-Weisbach law.

```
#include "friction.hpp"
```

Classes

- class [Fr_Darcy_Weisbach](#)
Darcy-Weisbach law.

6.16.1 Detailed Description

Darcy-Weisbach law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Friction law: Darcy-Weisbach.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.17 Headers/libfrictions/fr_laminar.hpp File Reference

laminar law

```
#include "friction.hpp"
```

Classes

- class [Fr_Laminar](#)

Laminar law.

6.17.1 Detailed Description

laminar law

Author

Carine Lucas carine.lucas@univ-orleans.fr (2014-2015)

Version

1.02.01

Date

2015-03-12

Friction law: laminar.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.18 Headers/libfrictions/fr_manning.hpp File Reference

Manning law.

```
#include "friction.hpp"
```

Classes

- class [Fr_Manning](#)

Manning law.

6.18.1 Detailed Description

Manning law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Friction law: Manning.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.19 Headers/libfrictions/friction.hpp File Reference

Friction law

```
#include "parameters.hpp"
```

Classes

- class [Friction](#)
Friction law

6.19.1 Detailed Description

Friction law

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the friction laws.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.20 Headers/libfrictions/no_friction.hpp File Reference

No friction.

```
#include "friction.hpp"
```

Classes

- class [No_Friction](#)

No friction.

6.20.1 Detailed Description

No friction.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Friction law: does no computation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.21 Headers/libinitializations/choice_init_hu.hpp File Reference

Choice of initialization for h and u.

```
#include "initialization_hu.hpp"
#include "hu_read.hpp"
#include "hu_generated.hpp"
#include "hu_generated_wet_dam_break.hpp"
#include "hu_generated_dry_dam_break.hpp"
#include "hu_generated_dressler_dam_break.hpp"
#include "hu_generated_thacker.hpp"
```

Classes

- class [Choice_init_hu](#)

Choice of initialization for h and u.

Macros

- #define [CHOICE_INIT_HU_HPP](#)

6.21.1 Detailed Description

Choice of initialization for h and u.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
 Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.21.2 Macro Definition Documentation

#define CHOICE_INIT_HU_HPP

Definition at line 87 of file choice_init_hu.hpp.

6.22 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"
#include "topo_generated_bump.hpp"
```

Classes

- class [Choice_init_topo](#)
Choice of initialization for the topography.

Macros

- #define [CHOICE_INIT_TOPO_HPP](#)

6.22.1 Detailed Description

Choice of initialization for the topography.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.22.2 Macro Definition Documentation

#define CHOICE_INIT_TOPO_HPP

Definition at line 78 of file choice_init_topo.hpp.

6.23 Headers/libinitializations/hu_generated.hpp File Reference

No water configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class [Hu_generated](#)

No water configuration.

6.23.1 Detailed Description

No water configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.24 Headers/libinitializations/hu_generated_dressler_dam_break.hpp File Reference

Dressler dam break configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class [Hu_generated_Dressler_Dam_break](#)
Dressler dam break configuration.

6.24.1 Detailed Description

Dressler dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the water height and of the velocity: case of a dam break as studied by Dressler (with friction).

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.25 Headers/libinitializations/hu_generated_dry_dam_break.hpp File Reference

Dry dam break configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class [Hu_generated_Dry_Dam_break](#)
Dry dam break configuration.

6.25.1 Detailed Description

Dry dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the water height and of the velocity: case of a dam break on a dry domain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.26 Headers/libinitializations/hu_generated_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class [Hu_generated_Thacker](#)
Thacker configuration.

6.26.1 Detailed Description

Thacker configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.27 Headers/libinitializations/hu_generated_wet_dam_break.hpp File Reference

Wet dam break configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class [Hu_generated_Wet_Dam_break](#)
Wet dam break configuration.

6.27.1 Detailed Description

Wet dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the water height and of the velocity: case of a dam break on a wet domain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.28 Headers/libinitializations/hu_read.hpp File Reference

File configuration.

```
#include "initialization_hu.hpp"
```

Classes

- class [Hu_read](#)
File configuration.

6.28.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-02-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.29 Headers/libinitializations/initialization_hu.hpp File Reference

Initialization of h and u

```
#include "parameters.hpp"
```

Classes

- class [Initialization_hu](#)
Initialization of h and u.

6.29.1 Detailed Description

Initialization of h and u

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.30 Headers/libinitializations/initialization_topo.hpp File Reference

Initialization of z

```
#include "parameters.hpp"
```

Classes

- class [Initialization_topo](#)

Initialization of z.

6.30.1 Detailed Description

Initialization of z

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the initialization of the topography.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.31 Headers/libinitializations/topo_generated_bump.hpp File Reference

Bump configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class [Topo_generated_bump](#)

Bump configuration.

6.31.1 Detailed Description

Bump configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2011-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the topography: the topography is a bump.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.32 Headers/libinitializations/topo_generated_flat.hpp File Reference

Flat configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class [Topo_generated_flat](#)

Flat configuration.

6.32.1 Detailed Description

Flat configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.33 Headers/libinitializations/topo_generated_thacker.hpp File Reference

Thacker configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class [Topo_generated_Thacker](#)
Thacker configuration.

6.33.1 Detailed Description

Thacker configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the topography: parabolic topography for Thacker's Benchmark.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.34 Headers/libinitializations/topo_read.hpp File Reference

File configuration.

```
#include "initialization_topo.hpp"
```

Classes

- class [Topo_read](#)
File configuration.

6.34.1 Detailed Description

File configuration.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.35 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](#)

6.35.1 Detailed Description

Choice of slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.35.2 Macro Definition Documentation

#define CHOICE_LIMITER_HPP

Definition at line 75 of file choice_limiter.hpp.

6.36 Headers/liblimitations/limiter.hpp File Reference

Slope limiter.

```
#include "parameters.hpp"
```

Classes

- class [Limiter](#)
Slope limiter.

6.36.1 Detailed Description

Slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the slope limiters.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.37 Headers/liblimitations/minmod.hpp File Reference

Minmod limiter

```
#include "limiter.hpp"
```

Classes

- class [Minmod](#)
Minmod slope limiter

6.37.1 Detailed Description

Minmod limiter

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Slope limiter: minmod.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.38 Headers/liblimitations/vanalbada.hpp File Reference

Van Albada limiter.

```
#include "limiter.hpp"
```

Classes

- class [VanAlbada](#)
Van Albada slope limiter.

6.38.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Slope limiter: Van Albada.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.39 Headers/liblimitations/vanleer.hpp File Reference

Van Leer limiter.

```
#include "limiter.hpp"
```

Classes

- class [VanLeer](#)

Van Leer slope limiter.

6.39.1 Detailed Description

Van Leer limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Slope limiter: Van Leer.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.40 Headers/libparameters/misc.hpp File Reference

Definitions.

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <fstream>
#include <cmath>
#include <cstring>
#include <cstdio>
#include <iomanip>
#include <vector>
#include <unistd.h>
#include <ctime>
#include <sstream>
#include <float>
```

Macros

- #define **MAX**(a, b) (a>=b?a:b)
- #define **MIN**(a, b) (a<=b?a:b)
- #define **GRAV** 9.81
- #define **GRAV_DEM** 4.905
- #define **NB_CHAR** 256
- #define **ZERO** 0.
- #define **HE_CA** 1.e-10
- #define **VE_CA** 1.e-10
- #define **IE_CA** 1.e-8
- #define **EPSILON** 1.e-13
- #define **RATIO_CLOSE_CELL** 1.e-3
- #define **VERSION** "FullSWOF_1D version 1.02.01, 2015-04-03"
- #define **MAX_SCAL** DBL_MAX

Typedefs

- typedef double **SCALAR**
- typedef vector< **SCALAR** > **VECT**

6.40.1 Detailed Description

Definitions.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-04-03

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.40.2 Macro Definition Documentation

#define EPSILON 1.e-13

Definition at line 81 of file misc.hpp.

#define GRAV 9.81

Definition at line 73 of file misc.hpp.

#define GRAV_DEM 4.905

Definition at line 74 of file misc.hpp.

#define HE_CA 1.e-10

Definition at line 78 of file misc.hpp.

#define IE_CA 1.e-8

Definition at line 80 of file misc.hpp.

#define MAX(a, b) (a>=b?a:b)

Definition at line 70 of file misc.hpp.

#define MAX_SCAL DBL_MAX

Definition at line 92 of file misc.hpp.

#define MIN(a, b) (a<=b?a:b)

Definition at line 71 of file misc.hpp.

#define NB_CHAR 256

Definition at line 76 of file misc.hpp.

#define RATIO_CLOSE_CELL 1.e-3

Definition at line 83 of file misc.hpp.

#define VE_CA 1.e-10

Definition at line 79 of file misc.hpp.

#define VERSION "FullSWOF_1D version 1.02.01, 2015-04-03"

Definition at line 85 of file misc.hpp.

#define ZERO 0.

Definition at line 77 of file misc.hpp.

6.40.3 Typedef Documentation

typedef double SCALAR

Definition at line 89 of file misc.hpp.

typedef vector<SCALAR> VECT

Definition at line 94 of file misc.hpp.

6.41 Headers/libparameters/parameters.hpp File Reference

Gets parameters.

```
#include "misc.hpp"  
#include "parser.hpp"
```

Classes

- class [Parameters](#)
Gets parameters.

6.41.1 Detailed Description

Gets parameters.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2011-2015)

Version

1.02.01

Date

2015-03-12

Reads the parameters, checks their values.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.42 Headers/libparser/parser.hpp File Reference

Parser

```
#include "misc.hpp"
```

Classes

- class [Parser](#)
Parser to read the entries

6.42.1 Detailed Description

Parser

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Reads the input file.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.43 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](#)

6.43.1 Detailed Description

Choice of infiltration law.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.43.2 Macro Definition Documentation

#define CHOICE_INFILTRATION_HPP

Definition at line 70 of file choice_infiltration.hpp.

6.44 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](#)

6.44.1 Detailed Description

Choice of initialization for the rain.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.44.2 Macro Definition Documentation

#define CHOICE_RAIN_HPP

Definition at line 73 of file choice_rain.hpp.

6.45 Headers/librain_infiltration/greenampt.hpp File Reference

Green-Ampt law.

```
#include "infiltration.hpp"
```

Classes

- class [GreenAmpt](#)
Green-Ampt law.

6.45.1 Detailed Description

Green-Ampt law.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Infiltration law: bi-layer Green-Ampt.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.46 Headers/librain_infiltration/infiltration.hpp File Reference

Infiltration

```
#include "parameters.hpp"
```

Classes

- class [Infiltration](#)
Definition of infiltration law.

6.46.1 Detailed Description

Infiltration

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Common part for the infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.47 Headers/librain_infiltration/no_infiltration.hpp File Reference

No infiltration.

```
#include "infiltration.hpp"
```

Classes

- class [No_Infiltration](#)
No infiltration.

6.47.1 Detailed Description

No infiltration.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Infiltration: there is no infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.48 Headers/librain_infiltration/no_rain.hpp File Reference

No rain.

```
#include "rain.hpp"
```

Classes

- class [No_Rain](#)
No rain.

6.48.1 Detailed Description

No rain.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Rain: there is no rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.49 Headers/librain_infiltration/rain.hpp File Reference

Rain

```
#include "parameters.hpp"
```

Classes

- class [Rain](#)

Initialization of the rain.

6.49.1 Detailed Description

Rain

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Common part for the initialization of the rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.50 Headers/librain_infiltration/rain_generated.hpp File Reference

Constant rain configuration.

```
#include "rain.hpp"
```

Classes

- class [Rain_generated](#)

Constant rain configuration.

6.50.1 Detailed Description

Constant rain configuration.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.51 Headers/librain_infiltration/rain_read.hpp File Reference

File configuration.

```
#include "rain.hpp"
```

Classes

- class [Rain_read](#)

File configuration.

6.51.1 Detailed Description

File configuration.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.52 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_HPP](#)

6.52.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.52.2 Macro Definition Documentation

#define CHOICE_RECONSTRUCTION_HPP

Definition at line 74 of file choice_reconstruction.hpp.

6.53 Headers/libreconstructions/eno.hpp File Reference

ENO reconstruction

```
#include "reconstruction.hpp"
```


Classes

- class [ENO](#)
ENO reconstruction

6.53.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Linear reconstruction: ENO.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.54 Headers/libreconstructions/eno_mod.hpp File Reference

Modified ENO reconstruction.

```
#include "reconstruction.hpp"
```

Classes

- class [ENO_mod](#)
Modified ENO reconstruction.

6.54.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Linear reconstruction: modified ENO.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.55 Headers/libreconstructions/hydrostatic.hpp File Reference

Hydrostatic reconstruction

```
#include "parameters.hpp"
```

Classes

- class [Hydrostatic](#)

Hydrostatic reconstruction

6.55.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.56 Headers/libreconstructions/muscl.hpp File Reference

MUSCL reconstruction

```
#include "reconstruction.hpp"
```

Classes

- class [MUSCL](#)

MUSCL reconstruction

6.56.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Linear reconstruction: MUSCL.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.57 Headers/libreconstructions/reconstruction.hpp File Reference

Reconstruction

```
#include "parameters.hpp"  
#include "choice_limiter.hpp"
```

Classes

- class [Reconstruction](#)
Reconstruction of the variables

6.57.1 Detailed Description

Reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the reconstructions.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.58 Headers/libsave/choice_output.hpp File Reference

Choice of output format.

```
#include "output.hpp"  
#include "gnuplot.hpp"
```

Classes

- class [Choice_output](#)
Choice of output format.

Macros

- #define [CHOICE_OUTPUT_HPP](#)

6.58.1 Detailed Description

Choice of output format.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.58.2 Macro Definition Documentation

#define CHOICE_OUTPUT_HPP

Definition at line 65 of file choice_output.hpp.

6.59 Headers/libsave/gnuplot.hpp File Reference

Gnuplot output

```
#include "output.hpp"
```

Classes

- class [Gnuplot](#)
Gnuplot output

6.59.1 Detailed Description

Gnuplot output

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Output format: optimized for Gnuplot.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.60 Headers/libsave/output.hpp File Reference

Output format

```
#include "parameters.hpp"
```

Classes

- class [Output](#)

Output format

6.60.1 Detailed Description

Output format

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the output formats.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.61 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](#)

6.61.1 Detailed Description

Choice of numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.61.2 Macro Definition Documentation

#define CHOICE_SCHEME_HPP

Definition at line 70 of file choice_scheme.hpp.

6.62 Headers/libschemas/order1.hpp File Reference

Order 1 scheme.

```
#include "scheme.hpp"
```

Classes

- class [Order1](#)
Order 1 scheme.

6.62.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical scheme: at order 1.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.63 Headers/libschemas/order2.hpp File Reference

Order 2 scheme.

```
#include "scheme.hpp"
```

Classes

- class [Order2](#)
Order 2 scheme.

6.63.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical scheme: at order 2.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.64 Headers/libschemas/scheme.hpp File Reference

Numerical scheme.

```
#include "hydrostatic.hpp"
#include "choice_condition.hpp"
#include "choice_flux.hpp"
#include "choice_friction.hpp"
#include "choice_init_topo.hpp"
#include "choice_init_hu.hpp"
#include "choice_rain.hpp"
#include "choice_infiltration.hpp"
#include "choice_output.hpp"
#include "choice_reconstruction.hpp"
```

Classes

- class [Scheme](#)

Numerical scheme.

6.64.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the numerical schemes.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.65 Sources/FullSWOF_1D.cpp File Reference

Main function.

```
#include "choice_scheme.hpp"
```

Functions

- int [main](#) ()

6.65.1 Detailed Description

Main function.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Runs the programm.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.65.2 Function Documentation

int main ()

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 57 of file FullSWOF_1D.cpp.

6.66 Sources/libboundaryconditions/bc_imp_discharge.cpp File Reference

Imposed discharge.

```
#include "bc_imp_discharge.hpp"
```

6.66.1 Detailed Description

Imposed discharge.

Author

Ulrich Razafison ulrich.razafison@math.cnrs.fr (2008)
 Carine Lucas carine.lucas@univ-orleans.fr (2011-2015)

Version

1.02.01

Date

2015-03-12

Boundary condition: imposed discharge (and water height if necessary).

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.67 Sources/libboundaryconditions/bc_imp_height.cpp File Reference

Imposed water height.

```
#include "bc_imp_height.hpp"
```

6.67.1 Detailed Description

Imposed water height.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
 Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.68 Sources/libboundaryconditions/bc_neumann.cpp File Reference

Neumann condition.

```
#include "bc_neumann.hpp"
```

6.68.1 Detailed Description

Neumann condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Boundary condition: Neumann condition (the normal derivative is null).

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.69 Sources/libboundaryconditions/bc_periodic.cpp File Reference

Periodic condition.

```
#include "bc_periodic.hpp"
```

6.69.1 Detailed Description

Periodic condition.

Author

Pierre-Antoine Ksinant pierreantoine.ksinantgarcia@gmail.com (2010)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Boundary condition: periodic condition.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.70 Sources/libboundaryconditions/bc_wall.cpp File Reference

Wall condition.

```
#include "bc_wall.hpp"
```

6.70.1 Detailed Description

Wall condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.71 Sources/libboundaryconditions/boundary_condition.cpp File Reference

Boundary condition.

```
#include "boundary_condition.hpp"
```

6.71.1 Detailed Description

Boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the boundary conditions.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.72 Sources/libboundaryconditions/choice_condition.cpp File Reference

Choice of boundary condition.

```
#include "choice_condition.hpp"
```

6.72.1 Detailed Description

Choice of boundary condition.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.73 Sources/libflux/choice_flux.cpp File Reference

Choice of numerical flux.

```
#include "choice_flux.hpp"
```

6.73.1 Detailed Description

Choice of numerical flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.74 Sources/libflux/f_hll.cpp File Reference

HLL flux.

```
#include "f_hll.hpp"
```

6.74.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: Harten, Lax, van Leer formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.75 Sources/libflux/f_hll2.cpp File Reference

HLL flux.

```
#include "f_hll2.hpp"
```

6.75.1 Detailed Description

HLL flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: Harten, Lax, van Leer reduced formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.76 Sources/libflux/f_kinetic.cpp File Reference

Kinetic flux.

```
#include "f_kinetic.hpp"
```

6.76.1 Detailed Description

Kinetic flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: kinetic formulation.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.77 Sources/libflux/f_rusanov.cpp File Reference

Rusanov flux.

```
#include "f_rusanov.hpp"
```

6.77.1 Detailed Description

Rusanov flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)
Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: Rusanov formulation.

Copyright

License Cecill-V2
http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.78 Sources/libflux/f_vfroe.cpp File Reference

VFRoe flux.

```
#include "f_vfroe.hpp"
```

6.78.1 Detailed Description

VFRoe flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical flux: VFRoe-ncv formulation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.79 Sources/libflux/flux.cpp File Reference

Numerical flux.

```
#include "flux.hpp"
```

6.79.1 Detailed Description

Numerical flux.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the numerical fluxes.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.80 Sources/libfrictions/choice_friction.cpp File Reference

Choice of friction law.

```
#include "choice_friction.hpp"
```

6.80.1 Detailed Description

Choice of friction law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.81 Sources/libfrictions/fr_darcy_weisbach.cpp File Reference

Darcy-Weisbach law.

```
#include "fr_darcy_weisbach.hpp"
```

6.81.1 Detailed Description

Darcy-Weisbach law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Friction law: Darcy-Weisbach.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.82 Sources/libfrictions/fr_laminar.cpp File Reference

Laminar law.

```
#include "fr_laminar.hpp"
```

6.82.1 Detailed Description

Laminar law.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2014-2015)

Version

1.02.01

Date

2015-03-12

Friction law: laminar.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.83 Sources/libfrictions/fr_manning.cpp File Reference

Manning law.

```
#include "fr_manning.hpp"
```

6.83.1 Detailed Description

Manning law.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Friction law: Manning.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.84 Sources/libfrictions/friction.cpp File Reference

Friction law

```
#include "friction.hpp"
```

6.84.1 Detailed Description

Friction law

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the friction laws.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.85 Sources/libfrictions/no_friction.cpp File Reference

No friction.

```
#include "no_friction.hpp"
```

6.85.1 Detailed Description

No friction.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Friction law: does no computation.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.86 Sources/libinitializations/choice_init_hu.cpp File Reference

Choice of initialization for h and u.

```
#include "choice_init_hu.hpp"
```

6.86.1 Detailed Description

Choice of initialization for h and u.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.87 Sources/libinitializations/choice_init_topo.cpp File Reference

Choice of initialization for the topography.

```
#include "choice_init_topo.hpp"
```

6.87.1 Detailed Description

Choice of initialization for the topography.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.88 Sources/libinitializations/hu_generated.cpp File Reference

No water configuration.

```
#include "hu_generated.hpp"
```

6.88.1 Detailed Description

No water configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.89 Sources/libinitializations/hu_generated_dressler_dam_break.cpp File Reference

Dressler dam break configuration.

```
#include "hu_generated_dressler_dam_break.hpp"
```

6.89.1 Detailed Description

Dressler dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the water height and of the velocity: case of a dam break as studied by Dressler (with friction).

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.90 Sources/libinitializations/hu_generated_dry_dam_break.cpp File Reference

Dry dam break configuration.

```
#include "hu_generated_dry_dam_break.hpp"
```

6.90.1 Detailed Description

Dry dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the water height and of the velocity: case of a dam break on a dry domain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.91 Sources/libinitializations/hu_generated_thacker.cpp File Reference

Thacker configuration.

```
#include "hu_generated_thacker.hpp"
```

6.91.1 Detailed Description

Thacker configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.92 Sources/libinitializations/hu_generated_wet_dam_break.cpp File Reference

Wet dam break configuration.

```
#include "hu_generated_wet_dam_break.hpp"
```

6.92.1 Detailed Description

Wet dam break configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the water height and of the velocity: case of a dam break on a wet domain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.93 Sources/libinitializations/hu_read.cpp File Reference

File configuration.

```
#include "hu_read.hpp"
```

6.93.1 Detailed Description

File configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.94 Sources/libinitializations/initialization_hu.cpp File Reference

Initialization of h and u

```
#include "initialization_hu.hpp"
```

6.94.1 Detailed Description

Initialization of h and u

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.95 Sources/libinitializations/initialization_topo.cpp File Reference

Initialization of z

```
#include "initialization_topo.hpp"
```

6.95.1 Detailed Description

Initialization of z

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the initialization of the topography.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.96 Sources/libinitializations/topo_generated_bump.cpp File Reference

Bump configuration.

```
#include "topo_generated_bump.hpp"
```

6.96.1 Detailed Description

Bump configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the topography: the topography is a bump.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.97 Sources/libinitializations/topo_generated_flat.cpp File Reference

Flat configuration.

```
#include "topo_generated_flat.hpp"
```

6.97.1 Detailed Description

Flat configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.98 Sources/libinitializations/topo_generated_thacker.cpp File Reference

Thacker configuration.

```
#include "topo_generated_thacker.hpp"
```

6.98.1 Detailed Description

Thacker configuration.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Initialization of the topography: parabolic topography for Thacker's Benchmark.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.99 Sources/libinitializations/topo_read.cpp File Reference

File configuration.

```
#include "topo_read.hpp"
```

6.99.1 Detailed Description

File configuration.

Author

Christian Laguerre christian.laguerre@math.cnrs.fr (2009)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.100 Sources/liblimitations/choice_limiter.cpp File Reference

Choice of slope limiter.

```
#include "choice_limiter.hpp"
```

6.100.1 Detailed Description

Choice of slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2010)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.101 Sources/liblimitations/limiter.cpp File Reference

Slope limiter.

```
#include "limiter.hpp"
```

6.101.1 Detailed Description

Slope limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the slope limiters.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.102 Sources/liblimitations/minmod.cpp File Reference

Minmod limiter

```
#include "minmod.hpp"
```

6.102.1 Detailed Description

Minmod limiter

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Slope limiter: minmod.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.103 Sources/liblimitations/vanalbada.cpp File Reference

Van Albada limiter.

```
#include "vanalbada.hpp"
```

6.103.1 Detailed Description

Van Albada limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Slope limiter: Van Albada.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.104 Sources/liblimitations/vanleer.cpp File Reference

Van Leer limiter.

```
#include "vanleer.hpp"
```

6.104.1 Detailed Description

Van Leer limiter.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Slope limiter: Van Leer.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.105 Sources/libparameters/parameters.cpp File Reference

Gets parameters.

```
#include "parameters.hpp"
```

6.105.1 Detailed Description

Gets parameters.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2011-2015)

Frederic Darboux frederic.darboux@orleans.inra.fr (2014)

Version

1.02.01

Date

2015-03-12

Reads the parameters, checks their values.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.106 Sources/libparser/parser.cpp File Reference

Parser

```
#include "parser.hpp"
```

6.106.1 Detailed Description

Parser

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Reads the input file.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.107 Sources/librain_infiltration/choice_infiltration.cpp File Reference

Choice of infiltration law.

```
#include "choice_infiltration.hpp"
```

6.107.1 Detailed Description

Choice of infiltration law.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.108 Sources/librain_infiltration/choice_rain.cpp File Reference

Choice of initialization for the rain.

```
#include "choice_rain.hpp"
```

6.108.1 Detailed Description

Choice of initialization for the rain.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.109 Sources/librain_infiltration/greenampt.cpp File Reference

Green-Ampt law.

```
#include "greenampt.hpp"
```

6.109.1 Detailed Description

Green-Ampt law.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Infiltration law: bi-layer Green-Ampt.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.110 Sources/librain_infiltration/infiltration.cpp File Reference

Infiltration

```
#include "infiltration.hpp"
```

6.110.1 Detailed Description

Infiltration

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Common part for the infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.111 Sources/librain_infiltration/no_infiltration.cpp File Reference

No infiltration.

```
#include "no_infiltration.hpp"
```

6.111.1 Detailed Description

No infiltration.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Infiltration: there is no infiltration.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.112 Sources/librain_infiltration/no_rain.cpp File Reference

No rain.

```
#include "no_rain.hpp"
```

6.112.1 Detailed Description

No rain.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Rain: there is no rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.113 Sources/librain_infiltration/rain.cpp File Reference

Rain

```
#include "rain.hpp"
```

6.113.1 Detailed Description

Rain

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

Common part for the initialization of the rain.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.114 Sources/librain_infiltration/rain_generated.cpp File Reference

Constant rain configuration.

```
#include "rain_generated.hpp"
```

6.114.1 Detailed Description

Constant rain configuration.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.115 Sources/librain_infiltration/rain_read.cpp File Reference

File configuration.

```
#include "rain_read.hpp"
```

6.115.1 Detailed Description

File configuration.

author Carine Lucas carine.lucas@univ-orleans.fr (2013-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.116 Sources/libreconstructions/choice_reconstruction.cpp File Reference

Choice of reconstruction.

```
#include "choice_reconstruction.hpp"
```

6.116.1 Detailed Description

Choice of reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.117 Sources/libreconstructions/eno.cpp File Reference

ENO reconstruction

```
#include "eno.hpp"
```

6.117.1 Detailed Description

ENO reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Linear reconstruction: ENO.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.118 Sources/libreconstructions/eno_mod.cpp File Reference

Modified ENO reconstruction.

```
#include "eno_mod.hpp"
```

6.118.1 Detailed Description

Modified ENO reconstruction.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Linear reconstruction: modified ENO.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.119 Sources/libreconstructions/hydrostatic.cpp File Reference

Hydrostatic reconstruction

```
#include "hydrostatic.hpp"
```

6.119.1 Detailed Description

Hydrostatic reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.120 Sources/libreconstructions/muscl.cpp File Reference

MUSCL reconstruction

```
#include "muscl.hpp"
```

6.120.1 Detailed Description

MUSCL reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Linear reconstruction: MUSCL.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.121 Sources/libreconstructions/reconstruction.cpp File Reference

Reconstruction

```
#include "reconstruction.hpp"
```

6.121.1 Detailed Description

Reconstruction

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the reconstructions.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.122 Sources/libsave/choice_output.cpp File Reference

Choice of output format.

```
#include "choice_output.hpp"
```

6.122.1 Detailed Description

Choice of output format.

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.123 Sources/libsave/gnuplot.cpp File Reference

Gnuplot output

```
#include "gnuplot.hpp"
```

6.123.1 Detailed Description

Gnuplot output

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Output format: optimized for Gnuplot (for huz_evolution.dat).

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.124 Sources/libsave/output.cpp File Reference

Output format

```
#include "output.hpp"
```

6.124.1 Detailed Description

Output format

Author

Carine Lucas carine.lucas@univ-orleans.fr (2010-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the output formats.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.125 Sources/libschemas/choice_scheme.cpp File Reference

Choice of numerical scheme.

```
#include "choice_scheme.hpp"
```

6.125.1 Detailed Description

Choice of numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

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

(c) CNRS - Universite d'Orleans - INRA (France)

6.126 Sources/libschemas/order1.cpp File Reference

Order 1 scheme.

```
#include "order1.hpp"
```

6.126.1 Detailed Description

Order 1 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical scheme: at order 1.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.127 Sources/libschemas/order2.cpp File Reference

Order 2 scheme.

```
#include "order2.hpp"
```

6.127.1 Detailed Description

Order 2 scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Numerical scheme: at order 2.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (France)

6.128 Sources/libschemas/scheme.cpp File Reference

Numerical scheme.

```
#include "scheme.hpp"
```

6.128.1 Detailed Description

Numerical scheme.

Author

Olivier Delestre olivierdelestre41@yahoo.fr (2008)

Carine Lucas carine.lucas@univ-orleans.fr (2012-2015)

Version

1.02.01

Date

2015-03-12

Common part for all the numerical schemes.

Copyright

License Cecill-V2

http://www.cecill.info/licences/Licence_CeCILL_V2-en.html

(c) CNRS - Universite d'Orleans - INRA (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. [49](#), [67](#)
- 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](#). [72](#)
- 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](#). [43](#), [44](#), [46](#), [47](#), [50](#)
- 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:[DOI: 10.1016/S1574-6909\(06\)02004-1](#). URL <http://www.sciencedirect.com/science/article/B8JG3-4PS6TNS-6/2/f4c3dbcf476626c1cb6f353e9bc66cc9>. [43](#), [44](#), [81](#)
- M.-O. Bristeau and B. Coussin. Boundary conditions for the shallow water equations solved by kinetic schemes. Technical Report RR-4282, INRIA, 2001. [1](#), [25](#)
- O. Delestre. *Simulation du ruissellement d'eau de pluie sur des surfaces agricoles*. PhD thesis, Université d'Orléans, Orléans, France, July 2010. URL <http://tel.archives-ouvertes.fr/tel-00531377>. [138](#), [184](#)
- R. F. Dressler. Hydraulic resistance effect upon the dam-break functions. *Journal of Research of the National Bureau of Standards*, 49(3):217–225, Sept. 1952. [66](#)
- T. Gallouët, J.-M. Hérard, and N. Seguin. Some approximate godunov schemes to compute shallow-water equations with topography. *Computers & Fluids*, 32:479–513, 2003. [51](#)
- N. Goutal and F. Maurel. Proceedings of the 2nd workshop on dam-break wave simulation. Technical Report HE-43/97/016/B, Electricité de France, Direction des études et recherches, 1997. [67](#), [69](#), [129](#)
- 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:[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. [43](#)
- 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. [43](#)
- 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](#), [25](#)
- 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:[DOI: 10.1016/0021-9991\(88\)90177-5](#). URL <http://www.sciencedirect.com/science/article/B6WHY-4DD1T6W-MM/2/bef99e4b67bd7132c8af1984c34ce57e>. [43](#)

- 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](https://doi.org/10.1177/0309133307081289). 55, 58
- 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](https://doi.org/10.1017/S0022112081001882). URL <http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=389055&fulltextType=RA&fileId=S0022112081001882>. 68, 132
- 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:[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>. 81

Index

- ~Bc_imp_discharge
 - Bc_imp_discharge, 16
- ~Bc_imp_height
 - Bc_imp_height, 18
- ~Bc_neumann
 - Bc_neumann, 21
- ~Bc_periodic
 - Bc_periodic, 22
- ~Bc_wall
 - Bc_wall, 24
- ~Boundary_condition
 - Boundary_condition, 25
- ~Choice_condition
 - Choice_condition, 27
- ~Choice_flux
 - Choice_flux, 29
- ~Choice_friction
 - Choice_friction, 31
- ~Choice_infiltration
 - Choice_infiltration, 32
- ~Choice_init_hu
 - Choice_init_hu, 34
- ~Choice_init_topo
 - Choice_init_topo, 35
- ~Choice_limiter
 - Choice_limiter, 36
- ~Choice_output
 - Choice_output, 37
- ~Choice_rain
 - Choice_rain, 39
- ~Choice_reconstruction
 - Choice_reconstruction, 40
- ~Choice_scheme
 - Choice_scheme, 41
- ~ENO
 - ENO, 43
- ~ENO_mod
 - ENO_mod, 44
- ~F_HLL
 - F_HLL, 45
- ~F_HLL2
 - F_HLL2, 47
- ~F_Kinetic
 - F_Kinetic, 48
- ~F_Rusanov
 - F_Rusanov, 50
- ~F_VFRoe
 - F_VFRoe, 51
- ~Flux
 - Flux, 53
- ~Fr_Darcy_Weisbach
 - Fr_Darcy_Weisbach, 55
- ~Fr_Laminar
 - Fr_Laminar, 56
- ~Fr_Manning
 - Fr_Manning, 58
- ~Friction
 - Friction, 59
- ~Gnuplot
 - Gnuplot, 61
- ~GreenAmpt
 - GreenAmpt, 63
- ~Hu_generated
 - Hu_generated, 65
- ~Hu_generated_Dressler_Dam_break
 - Hu_generated_Dressler_Dam_break, 66
- ~Hu_generated_Dry_Dam_break
 - Hu_generated_Dry_Dam_break, 67
- ~Hu_generated_Thacker
 - Hu_generated_Thacker, 68
- ~Hu_generated_Wet_Dam_break
 - Hu_generated_Wet_Dam_break, 69
- ~Hu_read
 - Hu_read, 70
- ~Hydrostatic
 - Hydrostatic, 72
- ~Infiltration
 - Infiltration, 74
- ~Initialization_hu
 - Initialization_hu, 75
- ~Initialization_topo
 - Initialization_topo, 77
- ~Limiter
 - Limiter, 78
- ~MUSCL
 - MUSCL, 81
- ~Minmod
 - Minmod, 80
- ~No_Friction
 - No_Friction, 82
- ~No_Infiltration
 - No_Infiltration, 84

- ~No_Rain
 - No_Rain, 85
- ~Order1
 - Order1, 86
- ~Order2
 - Order2, 87
- ~Output
 - Output, 89
- ~Parameters
 - Parameters, 97
- ~Parser
 - Parser, 114
- ~Rain
 - Rain, 115
- ~Rain_generated
 - Rain_generated, 116
- ~Rain_read
 - Rain_read, 118
- ~Reconstruction
 - Reconstruction, 119
- ~Scheme
 - Scheme, 122
- ~Topo_generated_Thacker
 - Topo_generated_Thacker, 132
- ~Topo_generated_bump
 - Topo_generated_bump, 129
- ~Topo_generated_flat
 - Topo_generated_flat, 130
- ~Topo_read
 - Topo_read, 133
- ~VanAlbada
 - VanAlbada, 134
- ~VanLeer
 - VanLeer, 135
- allocation
 - Scheme, 122
- amortENO
 - Parameters, 107
- BC_IMP_DISCHARGE_HPP
 - bc_imp_discharge.hpp, 138
- Bc_imp_discharge, 15
 - ~Bc_imp_discharge, 16
 - Bc_imp_discharge, 15
 - calc, 16
 - NewtonSolver, 16
 - ValDerPoly, 17
 - ValPoly, 17
- bc_imp_discharge.hpp
 - BC_IMP_DISCHARGE_HPP, 138
- Bc_imp_height, 18
 - ~Bc_imp_height, 18
 - Bc_imp_height, 18
- calc, 18
- Bc_neumann, 19
 - ~Bc_neumann, 21
 - Bc_neumann, 20
 - calc, 21
- Bc_periodic, 21
 - ~Bc_periodic, 22
 - Bc_periodic, 22
 - calc, 22
- Bc_wall, 23
 - ~Bc_wall, 24
 - Bc_wall, 23
 - calc, 24
- bound_flux
 - Choice_output, 37
 - Output, 89
- boundary
 - Scheme, 122
- Boundary_condition, 24
 - ~Boundary_condition, 25
 - Boundary_condition, 25
 - calc, 25
 - get_hbound, 25
 - get_ubound, 26
 - hbound, 26
 - NXCELL, 26
 - ubound, 26
 - ufix, 26
- c
 - Friction, 60
- CFL
 - Scheme, 124
- CHOICE_CONDITION_HPP
 - choice_condition.hpp, 142
- CHOICE_FLUX_HPP
 - choice_flux.hpp, 143
- CHOICE_FRICTION_HPP
 - choice_friction.hpp, 148
- CHOICE_INFILTRATION_HPP
 - choice_infiltration.hpp, 169
- CHOICE_INIT_HU_HPP
 - choice_init_hu.hpp, 152
- CHOICE_INIT_TOPO_HPP
 - choice_init_topo.hpp, 153
- CHOICE_LIMITER_HPP
 - choice_limiter.hpp, 162
- CHOICE_OUTPUT_HPP
 - choice_output.hpp, 178
- CHOICE_RAIN_HPP
 - choice_rain.hpp, 169
- CHOICE_RECONSTRUCTION_HPP
 - choice_reconstruction.hpp, 174
- CHOICE_SCHEME_HPP

- choice_scheme.hpp, 180
- calc
 - Bc_imp_discharge, 16
 - Bc_imp_height, 18
 - Bc_neumann, 21
 - Bc_periodic, 22
 - Bc_wall, 24
 - Boundary_condition, 25
 - Choice_condition, 27
 - Choice_flux, 29
 - Choice_friction, 31
 - Choice_infiltration, 32
 - Choice_limiter, 36
 - Choice_reconstruction, 40
 - Choice_scheme, 42
 - ENO, 43
 - ENO_mod, 44
 - F_HLL, 46
 - F_HLL2, 47
 - F_Kinetic, 49
 - F_Rusanov, 50
 - F_VFRoe, 51
 - Flux, 53
 - Fr_Darcy_Weisbach, 55
 - Fr_Laminar, 57
 - Fr_Manning, 58
 - Friction, 60
 - GreenAmpt, 63
 - Hydrostatic, 72
 - Infiltration, 74
 - Limiter, 79
 - MUSCL, 81
 - Minmod, 80
 - No_Friction, 82
 - No_Infiltration, 84
 - Order1, 86
 - Order2, 88
 - Reconstruction, 119
 - Scheme, 123
 - VanAlbada, 134
 - VanLeer, 135
- capacity
 - GreenAmpt, 63
- cfl
 - Flux, 54
 - Parameters, 107
- check_vol
 - Choice_output, 37
 - Output, 90
- Choice_condition, 26
 - ~Choice_condition, 27
 - calc, 27
 - Choice_condition, 27
 - get_hbound, 28
 - get_ubound, 28
- choice_condition.hpp
 - CHOICE_CONDITION_HPP, 142
- Choice_flux, 28
 - ~Choice_flux, 29
 - calc, 29
 - Choice_flux, 29
 - get_cfl, 29
 - get_f1, 29
 - get_f2, 29
 - set_tx, 30
- choice_flux.hpp
 - CHOICE_FLUX_HPP, 143
- Choice_friction, 30
 - ~Choice_friction, 31
 - calc, 31
 - Choice_friction, 30
 - get_qmod, 31
 - set_c, 31
 - set_dt, 31
- choice_friction.hpp
 - CHOICE_FRICTION_HPP, 148
- Choice_infiltration, 32
 - ~Choice_infiltration, 32
 - calc, 32
 - Choice_infiltration, 32
 - get_Vin, 33
 - get_hmod, 33
- choice_infiltration.hpp
 - CHOICE_INFILTRATION_HPP, 169
- Choice_init_hu, 33
 - ~Choice_init_hu, 34
 - Choice_init_hu, 33
 - initialization, 34
- choice_init_hu.hpp
 - CHOICE_INIT_HU_HPP, 152
- Choice_init_topo, 34
 - ~Choice_init_topo, 35
 - Choice_init_topo, 34
 - initialization, 35
- choice_init_topo.hpp
 - CHOICE_INIT_TOPO_HPP, 153
- Choice_limiter, 35
 - ~Choice_limiter, 36
 - calc, 36
 - Choice_limiter, 35
 - get_rec, 36
- choice_limiter.hpp
 - CHOICE_LIMITER_HPP, 162
- Choice_output, 36
 - ~Choice_output, 37
 - bound_flux, 37

- check_vol, 37
- Choice_output, 37
- initial, 38
- result, 38
- save, 38
- write, 38
- choice_output.hpp
 - CHOICE_OUTPUT_HPP, 178
- Choice_rain, 39
 - ~Choice_rain, 39
 - Choice_rain, 39
 - rain_func, 39
- choice_rain.hpp
 - CHOICE_RAIN_HPP, 169
- Choice_reconstruction, 40
 - ~Choice_reconstruction, 40
 - calc, 40
 - Choice_reconstruction, 40
- choice_reconstruction.hpp
 - CHOICE_RECONSTRUCTION_HPP, 174
- Choice_scheme, 41
 - ~Choice_scheme, 41
 - calc, 42
 - Choice_scheme, 41
- choice_scheme.hpp
 - CHOICE_SCHEME_HPP, 180
- cpu_time
 - Scheme, 124
- cum_flux_l
 - Scheme, 124
- cum_flux_r
 - Scheme, 124
- DT
 - Output, 91
 - Scheme, 124
- DX
 - Infiltration, 74
 - Initialization_hu, 76
 - Initialization_topo, 77
 - Output, 91
 - Rain, 115
 - Scheme, 124
- deallocation
 - Scheme, 123
- delta0_z
 - Reconstruction, 120
- delta_z
 - Scheme, 124
- dt
 - Friction, 60
 - Parameters, 108
- dtheta_NF
 - Parameters, 108
- dtheta_coef
 - Parameters, 108
- dtheta_init
 - Parameters, 108
- dtheta_namefile
 - Parameters, 108
- dx
 - Parameters, 108
- dzi
 - Scheme, 124
- ENO, 42
 - ~ENO, 43
 - calc, 43
 - ENO, 42
- ENO_mod, 43
 - ~ENO_mod, 44
 - calc, 44
 - ENO_mod, 44
- EPSILON
 - misc.hpp, 165
- elapsed_time
 - Scheme, 124
- end
 - Scheme, 125
- f1
 - Flux, 54
 - Scheme, 125
- f2
 - Flux, 54
 - Scheme, 125
- F_HLL, 45
 - ~F_HLL, 45
 - calc, 46
 - F_HLL, 45
- F_HLL2, 46
 - ~F_HLL2, 47
 - calc, 47
 - F_HLL2, 47
- F_Kinetic, 48
 - ~F_Kinetic, 48
 - calc, 49
 - F_Kinetic, 48
- F_Rusanov, 49
 - ~F_Rusanov, 50
 - calc, 50
 - F_Rusanov, 50
- F_VFRoe, 51
 - ~F_VFRoe, 51
 - calc, 51
 - F_VFRoe, 51
- fill_array
 - Parameters, 97

- Flux, [52](#)
 - ~Flux, [53](#)
 - calc, [53](#)
 - cfl, [54](#)
 - f1, [54](#)
 - f2, [54](#)
 - Flux, [53](#)
 - get_cfl, [53](#)
 - get_f1, [53](#)
 - get_f2, [53](#)
 - set_tx, [54](#)
 - tx, [54](#)
- flux
 - Parameters, [108](#)
- flux_l
 - Scheme, [125](#)
- flux_r
 - Scheme, [125](#)
- Fr
 - Scheme, [125](#)
- Fr_Darcy_Weisbach, [54](#)
 - ~Fr_Darcy_Weisbach, [55](#)
 - calc, [55](#)
 - Fr_Darcy_Weisbach, [55](#)
- Fr_Laminar, [56](#)
 - ~Fr_Laminar, [56](#)
 - calc, [57](#)
 - Fr_Laminar, [56](#)
- Fr_Manning, [57](#)
 - ~Fr_Manning, [58](#)
 - calc, [58](#)
 - Fr_Manning, [58](#)
- fric
 - Parameters, [108](#)
- friccoef
 - Parameters, [108](#)
- Friction, [59](#)
 - ~Friction, [59](#)
 - c, [60](#)
 - calc, [60](#)
 - dt, [60](#)
 - Friction, [59](#)
 - get_qmod, [60](#)
 - qmod, [60](#)
 - set_c, [60](#)
 - set_dt, [60](#)
- froude_number
 - Scheme, [123](#)
- FullSWOF_1D.cpp
 - main, [183](#)
- GRAV
 - misc.hpp, [165](#)
- GRAV_DEM
 - misc.hpp, [165](#)
- get_Kc_coef
 - Parameters, [101](#)
- get_Kc_init
 - Parameters, [101](#)
- get_KcNameFile
 - Parameters, [101](#)
- get_KcNameFileS
 - Parameters, [101](#)
- get_Ks_coef
 - Parameters, [101](#)
- get_Ks_init
 - Parameters, [101](#)
- get_KsNameFile
 - Parameters, [102](#)
- get_KsNameFileS
 - Parameters, [102](#)
- get_L_imp_h
 - Parameters, [102](#)
- get_L_imp_q
 - Parameters, [102](#)
- get_Lbound
 - Parameters, [102](#)
- get_Nxcell
 - Parameters, [103](#)
- get_Psi_coef
 - Parameters, [104](#)
- get_Psi_init
 - Parameters, [104](#)
- get_PsiNameFile
 - Parameters, [104](#)
- get_PsiNameFileS
 - Parameters, [104](#)
- get_R_imp_h
 - Parameters, [104](#)
- get_R_imp_q
 - Parameters, [104](#)
- get_Rbound
 - Parameters, [105](#)
- get_Vin
 - Choice_infiltration, [33](#)
 - Infiltration, [74](#)
- get_amortENO
 - Parameters, [97](#)
- get_cfl
 - Choice_flux, [29](#)
 - Flux, [53](#)
 - Parameters, [98](#)
- get_dt
 - Parameters, [98](#)
- get_dtheta_coef
 - Parameters, [98](#)
- get_dtheta_init

- Parameters, 98
- get_dthetaNameFile
 - Parameters, 98
- get_dthetaNameFileS
 - Parameters, 98
- get_dx
 - Parameters, 99
- get_f1
 - Choice_flux, 29
 - Flux, 53
- get_f2
 - Choice_flux, 29
 - Flux, 53
- get_flux
 - Parameters, 99
- get_fric
 - Parameters, 99
- get_friccoef
 - Parameters, 99
- get_hbound
 - Boundary_condition, 25
 - Choice_condition, 28
- get_hhydro_l
 - Hydrostatic, 72
- get_hhydro_r
 - Hydrostatic, 72
- get_hmod
 - Choice_infiltration, 33
 - Infiltration, 74
- get_hu
 - Parameters, 99
- get_huNameFile
 - Parameters, 99
- get_huNameFileS
 - Parameters, 100
- get_imax_coef
 - Parameters, 100
- get_imax_init
 - Parameters, 100
- get_imaxNameFile
 - Parameters, 100
- get_imaxNameFileS
 - Parameters, 100
- get_inf
 - Parameters, 100
- get_lim
 - Parameters, 102
- get_m
 - Parameters, 103
- get_modifENO
 - Parameters, 103
- get_nbtimes
 - Parameters, 103
- get_order
 - Parameters, 103
- get_outputDirectory
 - Parameters, 103
- get_qmod
 - Choice_friction, 31
 - Friction, 60
- get_rain
 - Parameters, 105
- get_rainNameFile
 - Parameters, 105
- get_rainNameFileS
 - Parameters, 105
- get_rec
 - Choice_limiter, 36
 - Limiter, 79
 - Parameters, 105
- get_suffix
 - Parameters, 105
- get_topo
 - Parameters, 106
- get_topographyNameFile
 - Parameters, 106
- get_topographyNameFileS
 - Parameters, 106
- get_tx
 - Parameters, 106
- get_ubound
 - Boundary_condition, 26
 - Choice_condition, 28
- get_zcrust_coef
 - Parameters, 106
- get_zcrust_init
 - Parameters, 106
- get_zcrustNameFile
 - Parameters, 107
- get_zcrustNameFileS
 - Parameters, 107
- GetValue
 - Parser, 114
- Gnuplot, 61
 - ~Gnuplot, 61
 - Gnuplot, 61
 - write, 62
- GreenAmpt, 62
 - ~GreenAmpt, 63
 - calc, 63
 - capacity, 63
 - GreenAmpt, 63
- h
 - Scheme, 125
- HE_CA
 - misc.hpp, 166

- hbound
 - Boundary_condition, 26
- Headers/libboundaryconditions/bc_imp_discharge.↔
hpp, 137
- Headers/libboundaryconditions/bc_imp_height.hpp,
138
- Headers/libboundaryconditions/bc_neumann.hpp, 138
- Headers/libboundaryconditions/bc_periodic.hpp, 139
- Headers/libboundaryconditions/bc_wall.hpp, 140
- Headers/libboundaryconditions/boundary_condition.↔
hpp, 140
- Headers/libboundaryconditions/choice_condition.hpp,
141
- Headers/libflux/choice_flux.hpp, 142
- Headers/libflux/f_hll.hpp, 143
- Headers/libflux/f_hll2.hpp, 144
- Headers/libflux/f_kinetic.hpp, 144
- Headers/libflux/f_rusanov.hpp, 145
- Headers/libflux/f_vfroe.hpp, 146
- Headers/libflux/flux.hpp, 146
- Headers/libfrictions/choice_friction.hpp, 147
- Headers/libfrictions/fr_darcy_weisbach.hpp, 148
- Headers/libfrictions/fr_laminar.hpp, 149
- Headers/libfrictions/fr_manning.hpp, 149
- Headers/libfrictions/friction.hpp, 150
- Headers/libfrictions/no_friction.hpp, 151
- Headers/libinitializations/choice_init_hu.hpp, 151
- Headers/libinitializations/choice_init_topo.hpp, 152
- Headers/libinitializations/hu_generated.hpp, 153
- Headers/libinitializations/hu_generated_dressler_↔
dam_break.hpp, 154
- Headers/libinitializations/hu_generated_dry_dam_↔
break.hpp, 154
- Headers/libinitializations/hu_generated_thacker.hpp,
155
- Headers/libinitializations/hu_generated_wet_dam_↔
break.hpp, 156
- Headers/libinitializations/hu_read.hpp, 156
- Headers/libinitializations/initialization_hu.hpp, 157
- Headers/libinitializations/initialization_topo.hpp, 157
- Headers/libinitializations/topo_generated_bump.hpp,
158
- Headers/libinitializations/topo_generated_flat.hpp, 159
- Headers/libinitializations/topo_generated_thacker.hpp,
159
- Headers/libinitializations/topo_read.hpp, 160
- Headers/liblimitations/choice_limiter.hpp, 161
- Headers/liblimitations/limiter.hpp, 162
- Headers/liblimitations/minmod.hpp, 162
- Headers/liblimitations/vanalbada.hpp, 163
- Headers/liblimitations/vanleer.hpp, 164
- Headers/libparameters/misc.hpp, 164
- Headers/libparameters/parameters.hpp, 167
- Headers/libparser/parser.hpp, 167
- Headers/librain_infiltration/choice_infiltration.hpp, 168
- Headers/librain_infiltration/choice_rain.hpp, 169
- Headers/librain_infiltration/greenampt.hpp, 169
- Headers/librain_infiltration/infiltration.hpp, 170
- Headers/librain_infiltration/no_infiltration.hpp, 171
- Headers/librain_infiltration/no_rain.hpp, 171
- Headers/librain_infiltration/rain.hpp, 172
- Headers/librain_infiltration/rain_generated.hpp, 172
- Headers/librain_infiltration/rain_read.hpp, 173
- Headers/libreconstructions/choice_reconstruction.↔
hpp, 174
- Headers/libreconstructions/eno.hpp, 174
- Headers/libreconstructions/eno_mod.hpp, 175
- Headers/libreconstructions/hydrostatic.hpp, 176
- Headers/libreconstructions/muscl.hpp, 176
- Headers/libreconstructions/reconstruction.hpp, 177
- Headers/libsave/choice_output.hpp, 178
- Headers/libsave/gnuplot.hpp, 178
- Headers/libsave/output.hpp, 179
- Headers/libschemas/choice_scheme.hpp, 180
- Headers/libschemas/order1.hpp, 180
- Headers/libschemas/order2.hpp, 181
- Headers/libschemas/scheme.hpp, 182
- height_inf_tot
 - Scheme, 125
- height_of_tot
 - Scheme, 125
- hl
 - Scheme, 125
- hl_rec
 - Hydrostatic, 73
- hleft
 - Scheme, 125
- hmod
 - Infiltration, 74
- hr
 - Scheme, 126
- hr_rec
 - Hydrostatic, 73
- hright
 - Scheme, 126
- hs
 - Scheme, 126
- hu_NF
 - Parameters, 109
- Hu_generated, 64
 - ~Hu_generated, 65
 - Hu_generated, 64
 - initialization, 65
- Hu_generated_Dressler_Dam_break, 65
 - ~Hu_generated_Dressler_Dam_break, 66
 - Hu_generated_Dressler_Dam_break, 66

- initialization, 66
- Hu_generated_Dry_Dam_break, 66
 - ~Hu_generated_Dry_Dam_break, 67
 - Hu_generated_Dry_Dam_break, 67
 - initialization, 67
- Hu_generated_Thacker, 67
 - ~Hu_generated_Thacker, 68
 - Hu_generated_Thacker, 68
 - initialization, 68
- Hu_generated_Wet_Dam_break, 69
 - ~Hu_generated_Wet_Dam_break, 69
 - Hu_generated_Wet_Dam_break, 69
 - initialization, 69
- hu_init
 - Parameters, 108
- hu_namefile
 - Parameters, 108
- Hu_read, 70
 - ~Hu_read, 70
 - Hu_read, 70
 - initialization, 71
- Hydrostatic, 71
 - ~Hydrostatic, 72
 - calc, 72
 - get_hhydro_l, 72
 - get_hhydro_r, 72
 - hl_rec, 73
 - hr_rec, 73
 - Hydrostatic, 72
- IE_CA
 - misc.hpp, 166
- imax_NF
 - Parameters, 109
- imax_coef
 - Parameters, 109
- imax_init
 - Parameters, 109
- imax_namefile
 - Parameters, 109
- inf
 - Parameters, 109
- Infiltration, 73
 - ~Infiltration, 74
 - calc, 74
 - DX, 74
 - get_Vin, 74
 - get_hmod, 74
 - hmod, 74
 - Infiltration, 74
 - NXCELL, 74
 - Vin, 74
- initial
 - Choice_output, 38
 - Output, 90
- initialization
 - Choice_init_hu, 34
 - Choice_init_topo, 35
 - Hu_generated, 65
 - Hu_generated_Dressler_Dam_break, 66
 - Hu_generated_Dry_Dam_break, 67
 - Hu_generated_Thacker, 68
 - Hu_generated_Wet_Dam_break, 69
 - Hu_read, 71
 - Initialization_hu, 76
 - Initialization_topo, 77
 - Topo_generated_Thacker, 132
 - Topo_generated_bump, 129
 - Topo_generated_flat, 131
 - Topo_read, 133
- Initialization_hu, 75
 - ~Initialization_hu, 75
 - DX, 76
 - initialization, 76
 - Initialization_hu, 75
 - M, 76
 - NXCELL, 76
- Initialization_topo, 76
 - ~Initialization_topo, 77
 - DX, 77
 - initialization, 77
 - Initialization_topo, 77
 - M, 77
 - NXCELL, 77
 - z, 77
- Kc_NF
 - Parameters, 109
- Kc_coef
 - Parameters, 109
- Kc_init
 - Parameters, 109
- Kc_namefile
 - Parameters, 109
- Ks_NF
 - Parameters, 110
- Ks_coef
 - Parameters, 109
- Ks_init
 - Parameters, 110
- Ks_namefile
 - Parameters, 110
- L
 - Parameters, 110
- L_IMP_H
 - Scheme, 126
- L_IMP_Q

- Scheme, 126
- L_imp_h
 - Parameters, 110
- L_imp_q
 - Parameters, 110
- Lbound
 - Parameters, 110
 - Scheme, 126
- lim
 - Parameters, 110
- Limiter, 78
 - ~Limiter, 78
 - calc, 79
 - get_rec, 79
 - Limiter, 78
 - rec, 79
- limiter
 - Reconstruction, 120
- M
 - Initialization_hu, 76
 - Initialization_topo, 77
 - Output, 91
 - Scheme, 126
- m
 - Parameters, 110
- MAX
 - misc.hpp, 166
- MAX_SCAL
 - misc.hpp, 166
- MIN
 - misc.hpp, 166
- MUSCL, 80
 - ~MUSCL, 81
 - calc, 81
 - MUSCL, 81
- main
 - FullSWOF_1D.cpp, 183
- maincalc
 - Scheme, 123
- Minmod, 79
 - ~Minmod, 80
 - calc, 80
 - Minmod, 80
- misc.hpp
 - EPSILON, 165
 - GRAV, 165
 - GRAV_DEM, 165
 - HE_CA, 166
 - IE_CA, 166
 - MAX, 166
 - MAX_SCAL, 166
 - MIN, 166
 - NB_CHAR, 166
 - RATIO_CLOSE_CELL, 166
 - SCALAR, 166
 - VE_CA, 166
 - VECT, 166
 - VERSION, 166
 - ZERO, 166
- modifENO
 - Parameters, 110
- NB_CHAR
 - misc.hpp, 166
- NBTIMES
 - Output, 92
 - Scheme, 126
- NXCELL
 - Boundary_condition, 26
 - Infiltration, 74
 - Initialization_hu, 76
 - Initialization_topo, 77
 - Output, 92
 - Rain, 115
 - Reconstruction, 120
 - Scheme, 126
- namefile_check_volume
 - Output, 91
- namefile_end
 - Output, 92
- namefile_flux
 - Output, 92
- namefile_init
 - Output, 92
- namefile_res
 - Output, 92
- nbt
 - Output, 92
 - Scheme, 126
- nbtimes
 - Parameters, 110
- NewtonSolver
 - Bc_imp_discharge, 16
- No_Friction, 82
 - ~No_Friction, 82
 - calc, 82
 - No_Friction, 82
- No_Infiltration, 83
 - ~No_Infiltration, 84
 - calc, 84
 - No_Infiltration, 83
- No_Rain, 84
 - ~No_Rain, 85
 - No_Rain, 85
 - rain_func, 85
- Nxcell
 - Parameters, 111

- ORDER
 - Scheme, 126
- order
 - Parameters, 111
- Order1, 85
 - ~Order1, 86
 - calc, 86
 - Order1, 86
- Order2, 87
 - ~Order2, 87
 - calc, 88
 - Order2, 87
- out
 - Scheme, 127
- Output, 88
 - ~Output, 89
 - bound_flux, 89
 - check_vol, 90
 - DT, 91
 - DX, 91
 - initial, 90
 - M, 91
 - NBTIMES, 92
 - NXCELL, 92
 - namefile_check_volume, 91
 - namefile_end, 92
 - namefile_flux, 92
 - namefile_init, 92
 - namefile_res, 92
 - nbt, 92
 - Output, 89
 - outputDirectory, 92
 - result, 90
 - save, 91
 - write, 91
- output_directory
 - Parameters, 111
- outputDirectory
 - Output, 92
- Parameters, 92
 - ~Parameters, 97
 - amortENO, 107
 - cfl, 107
 - dt, 108
 - dtheta_NF, 108
 - dtheta_coef, 108
 - dtheta_init, 108
 - dtheta_namefile, 108
 - dx, 108
 - fill_array, 97
 - flux, 108
 - fric, 108
 - friccoef, 108
 - get_Kc_coef, 101
 - get_Kc_init, 101
 - get_KcNameFile, 101
 - get_KcNameFileS, 101
 - get_Ks_coef, 101
 - get_Ks_init, 101
 - get_KsNameFile, 102
 - get_KsNameFileS, 102
 - get_L_imp_h, 102
 - get_L_imp_q, 102
 - get_Lbound, 102
 - get_Nxcell, 103
 - get_Psi_coef, 104
 - get_Psi_init, 104
 - get_PsiNameFile, 104
 - get_PsiNameFileS, 104
 - get_R_imp_h, 104
 - get_R_imp_q, 104
 - get_Rbound, 105
 - get_amortENO, 97
 - get_cfl, 98
 - get_dt, 98
 - get_dtheta_coef, 98
 - get_dtheta_init, 98
 - get_dthetaNameFile, 98
 - get_dthetaNameFileS, 98
 - get_dx, 99
 - get_flux, 99
 - get_fric, 99
 - get_friccoef, 99
 - get_hu, 99
 - get_huNameFile, 99
 - get_huNameFileS, 100
 - get_imax_coef, 100
 - get_imax_init, 100
 - get_imaxNameFile, 100
 - get_imaxNameFileS, 100
 - get_inf, 100
 - get_lim, 102
 - get_m, 103
 - get_modifENO, 103
 - get_nbtimes, 103
 - get_order, 103
 - get_outputDirectory, 103
 - get_rain, 105
 - get_rainNameFile, 105
 - get_rainNameFileS, 105
 - get_rec, 105
 - get_suffix, 105
 - get_topo, 106
 - get_topographyNameFile, 106
 - get_topographyNameFileS, 106
 - get_tx, 106

- get_zcrust_coef, 106
- get_zcrust_init, 106
- get_zcrustNameFile, 107
- get_zcrustNameFileS, 107
- hu_NF, 109
- hu_init, 108
- hu_namefile, 108
- imax_NF, 109
- imax_coef, 109
- imax_init, 109
- imax_namefile, 109
- inf, 109
- Kc_NF, 109
- Kc_coef, 109
- Kc_init, 109
- Kc_namefile, 109
- Ks_NF, 110
- Ks_coef, 109
- Ks_init, 110
- Ks_namefile, 110
- L, 110
- L_imp_h, 110
- L_imp_q, 110
- Lbound, 110
- lim, 110
- m, 110
- modifENO, 110
- nbtimes, 110
- Nxcell, 111
- order, 111
- output_directory, 111
- Parameters, 97
- Psi_NF, 111
- Psi_coef, 111
- Psi_init, 111
- Psi_namefile, 111
- R_imp_h, 111
- R_imp_q, 111
- rain, 111
- rain_NF, 112
- rain_namefile, 111
- Rbound, 112
- rec, 112
- setparameters, 107
- suffix_o, 112
- T, 112
- topo, 112
- topo_NF, 112
- topography_namefile, 112
- tx, 112
- zcrust_NF, 113
- zcrust_coef, 112
- zcrust_init, 112
- zcrust_namefile, 113
- Parser, 113
 - ~Parser, 114
 - GetValue, 114
 - Parser, 113
- Psi_NF
 - Parameters, 111
- Psi_coef
 - Parameters, 111
- Psi_init
 - Parameters, 111
- Psi_namefile
 - Parameters, 111
- q
 - Scheme, 127
- qmod
 - Friction, 60
- qs
 - Scheme, 127
- R_IMP_H
 - Scheme, 127
- R_IMP_Q
 - Scheme, 127
- R_imp_h
 - Parameters, 111
- R_imp_q
 - Parameters, 111
- RATIO_CLOSE_CELL
 - misc.hpp, 166
- Rain, 114
 - ~Rain, 115
 - DX, 115
 - NXCELL, 115
 - Rain, 115
 - rain_func, 115
- rain
 - Parameters, 111
 - Scheme, 127
- rain_NF
 - Parameters, 112
- rain_func
 - Choice_rain, 39
 - No_Rain, 85
 - Rain, 115
 - Rain_generated, 117
 - Rain_read, 118
- Rain_generated, 116
 - ~Rain_generated, 116
 - rain_func, 117
 - Rain_generated, 116
- rain_namefile
 - Parameters, 111

- Rain_read, 117
 - ~Rain_read, 118
 - rain_func, 118
 - Rain_read, 117
- Rbound
 - Parameters, 112
 - Scheme, 127
- rec
 - Limiter, 79
 - Parameters, 112
- Reconstruction, 118
 - ~Reconstruction, 119
 - calc, 119
 - delta0_z, 120
 - limiter, 120
 - NXCELL, 120
 - Reconstruction, 119
 - zl, 120
 - zr, 120
- result
 - Choice_output, 38
 - Output, 90
- SCALAR
 - misc.hpp, 166
- save
 - Choice_output, 38
 - Output, 91
- Scheme, 120
 - ~Scheme, 122
 - allocation, 122
 - boundary, 122
 - CFL, 124
 - calc, 123
 - cpu_time, 124
 - cum_flux_l, 124
 - cum_flux_r, 124
 - DT, 124
 - DX, 124
 - deallocation, 123
 - delta_z, 124
 - dzi, 124
 - elapsed_time, 124
 - end, 125
 - f1, 125
 - f2, 125
 - flux_l, 125
 - flux_r, 125
 - Fr, 125
 - froude_number, 123
 - h, 125
 - height_inf_tot, 125
 - height_of_tot, 125
 - hl, 125
 - hleft, 125
 - hr, 126
 - hright, 126
 - hs, 126
 - L_IMP_H, 126
 - L_IMP_Q, 126
 - Lbound, 126
 - M, 126
 - maincalc, 123
 - NBTIMES, 126
 - NXCELL, 126
 - nbt, 126
 - ORDER, 126
 - out, 127
 - q, 127
 - qs, 127
 - R_IMP_H, 127
 - R_IMP_Q, 127
 - rain, 127
 - Rbound, 127
 - Scheme, 122
 - start, 127
 - TX, 127
 - time_initial, 127
 - Total_volume_outflow, 127
 - u, 128
 - ul, 128
 - ur, 128
 - us, 128
 - Vin_tot, 128
 - Vol_inf_tot, 128
 - Vol_of_tot, 128
 - Vol_rain_tot, 128
 - z, 128
- set_c
 - Choice_friction, 31
 - Friction, 60
- set_dt
 - Choice_friction, 31
 - Friction, 60
- set_tx
 - Choice_flux, 30
 - Flux, 54
- setparameters
 - Parameters, 107
- Sources/FullSWOF_1D.cpp, 183
- Sources/libboundaryconditions/bc_imp_discharge.cpp, 183
- Sources/libboundaryconditions/bc_imp_height.cpp, 184
- Sources/libboundaryconditions/bc_neumann.cpp, 185
- Sources/libboundaryconditions/bc_periodic.cpp, 185
- Sources/libboundaryconditions/bc_wall.cpp, 186

- Sources/libboundaryconditions/boundary_condition.↔
cpp, 186
- Sources/libboundaryconditions/choice_condition.cpp,
187
- Sources/libflux/choice_flux.cpp, 187
- Sources/libflux/f_hll.cpp, 188
- Sources/libflux/f_hll2.cpp, 188
- Sources/libflux/f_kinetic.cpp, 189
- Sources/libflux/f_rusanov.cpp, 189
- Sources/libflux/f_vfroe.cpp, 190
- Sources/libflux/flux.cpp, 190
- Sources/libfrictions/choice_friction.cpp, 191
- Sources/libfrictions/fr_darcy_weisbach.cpp, 191
- Sources/libfrictions/fr_laminar.cpp, 192
- Sources/libfrictions/fr_manning.cpp, 192
- Sources/libfrictions/friction.cpp, 193
- Sources/libfrictions/no_friction.cpp, 193
- Sources/libinitializations/choice_init_hu.cpp, 194
- Sources/libinitializations/choice_init_topo.cpp, 194
- Sources/libinitializations/hu_generated.cpp, 195
- Sources/libinitializations/hu_generated_dressler_↔
dam_break.cpp, 195
- Sources/libinitializations/hu_generated_dry_dam_↔
break.cpp, 196
- Sources/libinitializations/hu_generated_thacker.cpp,
196
- Sources/libinitializations/hu_generated_wet_dam_↔
break.cpp, 197
- Sources/libinitializations/hu_read.cpp, 197
- Sources/libinitializations/initialization_hu.cpp, 198
- Sources/libinitializations/initialization_topo.cpp, 198
- Sources/libinitializations/topo_generated_bump.cpp,
199
- Sources/libinitializations/topo_generated_flat.cpp, 199
- Sources/libinitializations/topo_generated_thacker.cpp,
200
- Sources/libinitializations/topo_read.cpp, 200
- Sources/liblimitations/choice_limiter.cpp, 201
- Sources/liblimitations/limiter.cpp, 201
- Sources/liblimitations/minmod.cpp, 202
- Sources/liblimitations/vanalbada.cpp, 202
- Sources/liblimitations/vanleer.cpp, 203
- Sources/libparameters/parameters.cpp, 203
- Sources/libparser/parser.cpp, 204
- Sources/librain_infiltration/choice_infiltration.cpp, 204
- Sources/librain_infiltration/choice_rain.cpp, 205
- Sources/librain_infiltration/greenampt.cpp, 205
- Sources/librain_infiltration/infiltration.cpp, 206
- Sources/librain_infiltration/no_infiltration.cpp, 206
- Sources/librain_infiltration/no_rain.cpp, 207
- Sources/librain_infiltration/rain.cpp, 207
- Sources/librain_infiltration/rain_generated.cpp, 208
- Sources/librain_infiltration/rain_read.cpp, 208
- Sources/libreconstructions/choice_reconstruction.cpp,
209
- Sources/libreconstructions/eno.cpp, 209
- Sources/libreconstructions/eno_mod.cpp, 210
- Sources/libreconstructions/hydrostatic.cpp, 210
- Sources/libreconstructions/muscl.cpp, 211
- Sources/libreconstructions/reconstruction.cpp, 211
- Sources/libsave/choice_output.cpp, 212
- Sources/libsave/gnuplot.cpp, 212
- Sources/libsave/output.cpp, 213
- Sources/libschemas/choice_scheme.cpp, 213
- Sources/libschemas/order1.cpp, 214
- Sources/libschemas/order2.cpp, 214
- Sources/libschemas/scheme.cpp, 215
- start
 - Scheme, 127
- suffix_o
 - Parameters, 112
- T
 - Parameters, 112
- TX
 - Scheme, 127
- time_initial
 - Scheme, 127
- topo
 - Parameters, 112
- topo_NF
 - Parameters, 112
- Topo_generated_Thacker, 131
 - ~Topo_generated_Thacker, 132
 - initialization, 132
 - Topo_generated_Thacker, 131
- Topo_generated_bump, 129
 - ~Topo_generated_bump, 129
 - initialization, 129
 - Topo_generated_bump, 129
- Topo_generated_flat, 130
 - ~Topo_generated_flat, 130
 - initialization, 131
 - Topo_generated_flat, 130
- Topo_read, 132
 - ~Topo_read, 133
 - initialization, 133
 - Topo_read, 133
- topography_namefile
 - Parameters, 112
- Total_volume_outflow
 - Scheme, 127
- tx
 - Flux, 54
 - Parameters, 112
- u

- Scheme, [128](#)
- ubound
 - Boundary_condition, [26](#)
- ufix
 - Boundary_condition, [26](#)
- ul
 - Scheme, [128](#)
- ur
 - Scheme, [128](#)
- us
 - Scheme, [128](#)
- VE_CA
 - misc.hpp, [166](#)
- VECT
 - misc.hpp, [166](#)
- VERSION
 - misc.hpp, [166](#)
- ValDerPoly
 - Bc_imp_discharge, [17](#)
- ValPoly
 - Bc_imp_discharge, [17](#)
- VanAlbada, [133](#)
 - ~VanAlbada, [134](#)
 - calc, [134](#)
 - VanAlbada, [134](#)
- VanLeer, [135](#)
 - ~VanLeer, [135](#)
 - calc, [135](#)
 - VanLeer, [135](#)
- Vin
 - Infiltration, [74](#)
- Vin_tot
 - Scheme, [128](#)
- Vol_inf_tot
 - Scheme, [128](#)
- Vol_of_tot
 - Scheme, [128](#)
- Vol_rain_tot
 - Scheme, [128](#)
- write
 - Choice_output, [38](#)
 - Gnuplot, [62](#)
 - Output, [91](#)
- z
 - Initialization_topo, [77](#)
 - Scheme, [128](#)
- ZERO
 - misc.hpp, [166](#)
- zcrust_NF
 - Parameters, [113](#)
- zcrust_coef
 - Parameters, [112](#)
- zcrust_init
 - Parameters, [112](#)
- zcrust_namefile
 - Parameters, [113](#)
- zl
 - Reconstruction, [120](#)
- zr
 - Reconstruction, [120](#)